International Business Research

www.ccsenet.org/journal.html

Vol. 2, No. 1

January 2009

Optimal Combination of Trading Rules Using Neural Networks

Subrata Kumar Mitra

Professor, Institute of Management Technology

35 Km Milestone, Katol Road

Nagpur – 441 502, India

Tel: 91-712-280-5000 E-mail: skmitra@imtnag.ac.in

Abstract

A large number of trading rules based on technical analysis of prices are being used by investing community for generating trading signals for short term investments. As profitability of these trading rules vary, it is not easy to judge which particular rule really 'works'. Instead of a single trading rule, combination of rules are likely to offer the portfolio benefits of better risk adjusted return and hence, an experiment is carried out to combine signals generated from of moving averages of different window size using an artificial neural network. It is observed that the risk adjusted performance measure of the artificial neural network based trading model is better than that of simple 'Buy and Hold' strategy.

Keywords: Trading Rules, Technical Analysis, Neural Networks

1. Introduction

The investors not only look for long term capital gains from the market, but also like to maximize returns exploiting opportunities from short term price movements. A large number of trading rules based on technical analysis of prices are used for generating buy and sell signal for short term investments. From the numerous trading rules being used by the trading community, it is not easy to judge which particular rule really 'works'.

Investment professionals with long years of experience may take very good trading decision form their expertise. But there is no guarantee that such decisions will always work and therefore the use of a systematic procedure to generate trading decisions becomes important. A systematic decision making approach can also help to overcome various limitations that are inherent in human professionals. Further, as different practitioners have different views on the same information set, systematic evaluation methods will reduce personal bias. In the area of investment management, where decisions involve very large amounts of money, sometimes the lifelong savings of the clients, reassurance of the soundness of the investment decision-making process is necessary.

Technical trading rules are increasingly being used in financial markets for over a century ever since it was popularized by Charles Dow in 1900. But analysis of trading rules have started drawing more attention in 1990s and several authors have expressed that financial prices and returns are predictable to some extent, either from their own past or from some other publicly available information. For example, Bessembinder & Chan (1995), Blume, Easley & O.hara (1994), Brock, Lakonishok & Lebaron (1992), Ramazan (1998), Jegadeesh & Titman (1993, 2000), Lo & MacKinlay (1988), Neftci (1991), Ready (1997) test various trading rules based on technical analysis and reported that technical analysis provides information beyond that already incorporated in the current price. However, there cannot be a fixed trading rule as excessive usage of a particular trading rule will reduce efficacy of that rule. If everybody starts using a particular rule, that rule will not work any more and hence, the trading rules need to be continuously upgraded based on changing market dynamics.

Analyses made the literature are based on performance of a specific trading rule used in isolation. Instead of relying on a single trading rule, traders often use a variety of trading rules and sometimes a combination. Combining of rules are likely to offer the portfolio benefits of better risk adjusted return. Markowitz (1952, 1959) has shown that in portfolio context unsystematic risk can be reduced by diversification; possibly similar benefit arises when multiple trading rules are combined for taking a trading decision.

2. Using Moving Averages for Generating Trading Signal

The moving average (MA) method is one of the most widely used methods of generating trading rules. It includes numeral versions and different levels of complexity. A moving average is an average of observations from several

consecutive time periods. To compute a moving average sequence, we compute successive averages of a given number of consecutive observations. The objective underlying the MA method is to smooth out seasonal variation in the data.

The most widely used moving average (MA) is the *n*-day simple MA given by: $SMA_t = \frac{1}{n}\sum_{i=t-n+1}^{t} P_i$, where SMA_t is the simple *n*-day moving average at period *t* and P_i is the closing price for period *i*. In the simple MA procedure, a buy signal is generated when the closing price rises above the MA and a sell signal is generated when the closing price falls below the MA. If there is a clear trend, this method will work well. If, however, the market move sideways or if there is excessive volatility, there will be a lot of false signals.

A modification of simple moving average is exponential moving average (*EMA*) that gives more weight to the most recent time periods. It is described recursively as: $EMA_t = \alpha.P_t + (1-\alpha).EMA_{t-1}$, where α is a value between 0 and 1.

For example, if α = 0.5, the most recent value P_t is given 50% weight and all other past values are given remaining 50% weight. When the computation begins, the current price is set to *EMA* and as more prices are available, the averaging process is continued.

$EMA_1 = P_1$

$EMA_2 = \alpha.P_2 + (1-\alpha).EMA_1$

$EMA_3 = \alpha.P_3 + (1-\alpha).EMA_2$

The exponential moving average performs well for many business applications, usually producing results superior to the moving average. (Krantardzic, 2001)

A moving average summarizes the recent past data, further; spotting the change in the trend of data may additionally improve forecasting performances. Some of the measures that compare current price with the moving averages are

$P_t - MA_t$: the difference between the current price and its moving average;

$MA_t - MA_{(t-k)}$: the differences between two moving averages, of same window size;

$MA_{(t,n)} - MA_{(t,m)}$: the differences between two moving averages, of different window size; and

$\frac{P_t}{MA_t}$: the ratio between the current value and its moving average.

In the present study the current price will be compared with its moving average in ratio format. When current price P_t is be more than its Moving average MA_t, it is considered as an indication of uptrend and a buy signal is generated and vice versa a sell signal is generated when current price is less than its moving average. Numerically, the ratio $\frac{P_t}{MA_t} > 1$ will generate a buy signal and the ratio $\frac{P_t}{MA_t} < 1$ will generate a sell signal.

The value of the ratio $\frac{P_t}{MA_t}$ will nevertheless depend on the window size of the moving average. A moving average having less window size, say 3 days, will follow current price closely and the ratio $\frac{P_t}{MA_t}$ will change from 'more than 1' to 'less than 1' more frequently generating a large number of buy and sell trading signals. Whereas a moving average having a large window size, 'say 200 days' will generate trading signal less frequently. More trading arising out of less window size may capture the minor movements of prices well, but consequently the transaction costs will also increase. The success of a moving average based trading method clearly depends on selection of a proper window size, but there is no known method to determine the window size. Therefore, an experiment is carried out to combine signals generated from of moving averages of different window size using an artificial neural network.

3. An Introduction to Artificial Neural Network

The artificial neural network based techniques are an information processing model derived from functioning of human brain. This is a simple information processing device that accepts many inputs, combines them, and produces an output. The basic element of a neural network is a neuron. The output of one neuron becomes input to other neurons. A neural network is a structure of many such neurons connected in a systematic way. In the study, the neural networks used are feed-forward neural networks, where information processing moves only in forward direction as shown in figure -1.

The neurons in the network are arranged in layers. Typically, there is one layer for input neurons (the input layer), one or more layers of internal processing units (the hidden layers), and one layer for output neurons (the output layer). Each layer is either partly or fully interconnected to the preceding layer and the following layer.

The connections between neurons have weights associated with them, which determine the strength of influence of a neuron to other neurons. Information flows from the input layer through the hidden processing layer(s) to the final output layer to generate predictions. The connection weights are determined by a training process, wherein known input and known output data is fed to the network. The network adjusts connected weights so that a relationship between inputs and outputs can be established with certain degree of accuracy.

3.1 Designing of Network Structure

There are many parameters to design a feed forward neural network. Decisions regarding number of inputs in the input layer, number of hidden layers and number of neurons in the hidden layers, interconnection of neurons among layers etc. are to be taken. Though some techniques are mentioned in literature for determining these parameters, there is no uniformity. Structure of the network largely remains a design issue and leaves ample scope of innovation to the analyst.

3.1.1 Input Layer

The input layer to the neural network is the medium through which the inputs are presented to the neural network. When a set of input is presented to the input later of the neural network, the inputs are processed and resultant information is passed to the subsequent layer(s). Every input neuron should represent some known variable that has an influence over the output of the neural network. As final output will depend on inputs introduced to the network, the quality and relevance inputs are very important.

3.1.2 Hidden Layers

There are really two decisions to be made with regards to the hidden layers. The first is how many hidden layers to have in the neural network and then how many neurons will be in each of these layers. Neural networks with two or more hidden layers can represent functions with any kind and hence there is no theoretical reason to use neural networks with any more than two hidden layers.

Deciding the number of hidden neurons in layers is an important part of deciding the overall neural network structure. Hidden layers do not directly interact with the external environment but influences the final output. Hence, both the number of hidden layers and number of neurons in each of these hidden layers must be carefully designed. Using lesser number of neurons in the hidden layers will result in under-fitting. Under-fitting occurs when few neurons in the hidden layers are unable detect relationships is complex scenario. On the contrary, using too many neurons in the hidden layers may result in over-fitting. Over-fitting occurs when the neural network has so much information processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers. Another problem can occur even when training dataset is very large. A large number of neurons in the hidden layers can increase the training time of the network. Over-fitting with large training data may fit past data very well. The objective of neural network model is to extract general relationship in the data which can be used in new environment. Thus generalization of network relations is more important than over-fitting.

There are few rule-of-thumb methods for deciding structure of hidden layer. These rules-of-thumb are only starting points to consider the initial structure. Ultimately the selection of the architecture of the neural network has to be finalized by experimentation.

- The number of neurons should be in the range between the size of the input layer and the size of the output layer.
- The number of neurons may be 2/3 of the input layer size, plus the size of the output layer.
- The number of neurons should be less than twice the input layer size.

In the present study, a three layered network with: one input layer having three input nodes, one hidden layer having three processing nodes and finally one output layer producing a single output is used, as shown in figure 1. The structure of the network can indeed be varied as per requirement of the analysis.

3.1.3 Output Layer

The output layer of the neural network presents output to the external environment. The output is derived from inputs via complex relationships inbuilt in the neural network structure.

3.2 Input-Output Relationships

Data ranges of real-world input parameters vary widely. For example, one variable may have data that ranges between 0 and 1, while another variable can be a five digit value. If both of the variables are used in their natural scale, the second variable is likely to be given much more weight in the model than the first variable, simply because of its original values (and therefore the differences between records). To compensate for this effect of scale, range fields are usually

transformed so that they all have the same scale. In the study, range fields are made uniform to have values between -0.50 and +0.5 by using a rescaled sigmoid function.

The activation of each neuron from input later to hidden layer and again from hidden layer to output layer is calculated as $a_j = f(\sum_i w_{ij} x_i)$, where a_j is the activation of neuron j, i is the set of neurons in the preceding layer, w_{ij} is the weight of the connection between neuron i and neuron j, x_i is the output of neuron i, $f(x)$ is a transfer function used to scale the summation values from -0.5 to +0.5. In the study, we used sigmoid or logistic transfer function to scale the numerical values. The original formula for sigmoid conversion is $f_{SIGMOID}(x) = \frac{1}{1+e^{-x}}$ which converts any x value within a range of 0 to 1. The sigmoid values are further rescaled by a deducting 0.5 so that values remain within -0.5 to + 0.5. Thus the conversion function is $f(x) = \frac{1}{1+e^{-x}} - 0.5$.

In the network used (figure -1), x_1, x_2 and x_3 are the input nodes, y_1, y_2 and y_3 are nodes in the hidden layer and z is the final output. The activation of each node in hidden layer is calculated by using following rescaled sigmoid function of weighted inputs.

$y_i = \frac{1}{1+e^{-\sum(w_{ij}.x_j)_i}} - 0.5$, where w_{ij} is the connection weights between y_i and x_j nodes. Similarly, the final output z can be estimated using the transformation: $z = \frac{1}{1+e^{-\sum(w_{ij}.y_j)_i}} - 0.5$.

The final value (z) is used for generating trading signal. Values of these rescaled sigmoid functions range between -0.5 and +0.5 with mean value of 0. If the value of z is found positive, it is considered as a signal of uptrend and conversely, when the value of z is negative it is taken as a signal of down trend. If buy, hold and sell decisions are represented by +1, 0 and -1 respectively, then final buy and sell decisions are determined using the value of Sign(z). Sign(.) determines the sign of a number: returns 1 if the number is positive, zero (0) if the number is 0, and -1 if the number is negative.

3.3 Training

The output value z can be calculated from the inputs (x_i) and connection weights w_{ij} using relationships mentioned in previous section. The values of x_i (inputs) are known to us but the values of connection weights (w_{ij}) are not known. The training the network is carried out to find out values of w_{ij}, so that these values can be used for generating future signals. The objective is to forecast output $z_{(t+1)}$, which will match with future actual return $r_{(t+1)}$. However the future returns can never be accurately predicted and any prediction will always have some error. The purpose is to minimize these errors as much as possible so that the forecast is of some practical use. The total error can be measured by adding absolute errors of each observation $ABS(r_{(t+1)} - z_{(t+1)})$. A more acceptable form is based on minimization of Total Squared Error (TSE): Minimize: $\Sigma (r_{(t+1)} - z_{(t+1)})^2$

The minimization can be done using any commercially available software. In the study, the optimization was carried out using "Solver" add-in available in Microsoft Excel. Solver uses the Generalized Reduced Gradient (GRG2) nonlinear optimization code developed by Leon Lasdon, University of Texas at Austin, and Allan Waren, Cleveland State University. When Solver reaches an acceptable solution, it has minimized the total squared error term TSE by changing value of specified cells (these cells are weights w_{ij} of the network). The values of changed cells are the optimized weights and can be used in predicting trading signals in future.

4. Empirical Testing

4.1 Data

The study examines the profitability of technical trading rules applied to three Indian Stock Indices for the period 1st April 1998 to 31st December 2007, covering a period of 10 years. The daily closing values of following indices are analyzed in the study (details on these indices can be obtained from www.nseindia.co.in).

- S&P CNX Nifty
- CNX Nifty Junior
- CNX Defty

4.2 Converting Indices data to Network Inputs

Inputs to the network must contain information pertaining to output (to predict price movements). A large number of academic studies support usefulness of moving averages for determining trends in stock price series. The following inputs selected in the study compares current price with past moving averages.

- The first input compares closing price of the security with its 3 day moving average :

- $x_1 = (p_t$ / Moving average of past 3 days)
- The second input compares closing price of the security with its 7 day moving average:
- $x_2 = (p_t$ / Moving average of past 7 days)
- The third input compares closing price of the security with its 30 day moving average:
- $x_3 = (p_t$ / Moving average of past 30 days)

4.3 Finding Value of Network Weights

Training of network refers to a method of determining the value of connecting weights of the network based on a certain performance measure, such as cumulative profit. The performance of the trading systems is usually determined by optimizing over past known data, but there is no consensus on how much past data to be used. A common procedure to assess the profitability of technical trading is to choose the optimal parameter using the first part of the available data and then test the parameter upon the remaining data for out-of-sample verification. Out-of-sample verification is an important factor in testing the performance of technical trading strategies due to the danger of data snooping biases.

For each financial series, the training procedure is carried out using the past one year's data. The network weights (w_{ij}), that have shown the best performance over a year, are used for the out-of-sample trading in the next year. At the end of the next year, new optimal weights for the year are again calculated, and this procedure is repeated during the rest of the sample period. For example, the connection weights used for the year 2000-01 are trained weights that generated the highest cumulative return in the year 1999-00. The new connection weights for 2001-02 are selected using the data for the year 2000-01, and so forth. This procedure ensures that the entire neural network model is adaptive and all the trading results are out-of-sample.

4.4 Estimation of Profit (Loss)

Profitability of a trading position depends on the change of market price of the traded security and the position of the trader (either long or short). If trader has taken a long position, he will be benefited by a price rise of the security but will incur loss by a price decrease. Similarly the trader can make profit in a declining market by taking a short position. The trader is presumed to take a long position whenever the network output $z_{(t+1)}$ gives a positive value. Likewise, a short position is taken whenever the value of $z_{(t+1)}$ is negative. The trading decision can be represented by the following dummy variable.

$$d_{(t+1)} = \begin{cases} 1, & \text{if } z_{(t+1)} > 0 \\ d_{(t)}, & \text{if } z_{(t+1)} = 0 \\ -1, & \text{if } z_{(t+1)} < 0 \end{cases}$$

The dummy variable $d_{(t+1)}$ is equal to one (negative one) when the trader goes long (short) in traded asset. The return of the trader on a particular day can be estimated as follows:

$r_{(t+1)} = d_{(t+1)} \ (P_{(t+1)} - P_{(t)})$, where, $r_{(t+1)}$ denotes the return of the trader resulting from the decision taken ($d_{(t+1)}$) at the close of period t, which depends on value of $d_{(t+1)}$ and change in the asset value within period t and (t+1).

When $d_{(t+1)} = d_{(t)}$, the existing position (long/short position in the asset) is maintained and no new transaction need to be carried out. Hence transaction cost is not applicable. If $d_{(t+1)} \neq d_{(t)}$, then the position held is reversed at the close of period (t+1), necessitating two single transaction (closing existing position and opening a new position in opposite direction). Taking transaction cost into account, daily gross profit becomes:

$$r_{(t+1)} = d_{(t+1)} \ (P_{(t+1)} - P_{(t)}) - c \ | d_{(t+1)} - d_{(t)} | P_{(t),}$$

where c is transaction cost (in fraction of asset value) of a single transaction and $| d_{(t+1)} - d_{(t)} |$ denotes absolute value of the difference $d_{(t+1)} - d_{(t)}$. Total cumulative profit after transaction costs can be obtained adding daily profits.

About 15 years ago, transaction costs in Indian markets used to be very high. But the scenario is changed now. Brokerage rates on Indian bourses have crashed to historic lows due to competition. Apart from competition, sustained reforms in the financial markets have led to lower transaction cost. Brokers can no longer justify higher transaction cost after introduction screen based trading, electronic transfer of shares through depositories, launch of internet driven trading and substantial increase of trading volume. The brokerage rates in the market during the period have plummeted from around 2% for delivery bases transactions to less than 0.05% of turnover for future trades. All profitability calculation in the study is carried out at 0.05% transaction cost analogous to cost applicable in futures market.

4.5 Trading Profits

The trading results of using Neural Network model is calculated for these three financial series and the same are compared with the profitability of Buy and Hold strategy. In Buy and Hold Strategy, the security is bought at the start of the study period and sold at the end of the period. No transaction is carried out during the period and no transaction cost is incurred. Whereas in trading model using neural networks, transactions were many causing high transaction cost. The

profitability of Neural Network model is compared with Buy & Hold strategy for the selected three series. The results are given in tables 1, 2 & 3.

It may be seen from the tables that total net profit using Neural Network Model is generally higher than that of Buy and Hold Strategy even after transaction costs indicating usefulness of technical trading rules.

4.6 Statistical tests

The most widely used risk adjusted investment performance measure is developed by Prof. William F. Sharpe; his measure is not only widely used in academia but also by market practitioners.

The Sharpe Ratio (SR) can be calculated as follows.

$$SR = \frac{\mu(r_i) - r_f}{\sigma_i}$$

Where r_i is the return over period i, is the mean and is the sample standard deviation over the n periods observed and r_f is the risk-free rate of interest. Originally, the benchmark for the Sharpe Ratio was taken to be a risk-less security, where the differential return is equal to the excess return of the fund over a one-period risk-less rate of interest. The usefulness of the Sharpe Ratio is based on the premise that a differential return represents the result of a zero-investment strategy. But in case of trading in a forward or future contract, one need not finance the asset by making full payment, often such contracts can be purchased by providing a small margin payment or providing some short of guarantee. Therefore traded contracts of stock index futures can be considered as zero-investment strategies.

Sharpe ratio in zero-investment strategies can be calculated omitting risk free rate as follows.

$$SR = \frac{\mu(r_i)}{\sigma_i}$$

The Sharpe ratio of an investment can be compared with any benchmark by computing the Sharpe Ratio for the benchmark and compare with the investment model. The Sharpe Ratio of Neural Network model and Buy and Hold Strategy are compared for the three financial series in Tables 4, 5 and 6.

In the tables, it can be found that Sharpe Ratio of Neural Network model is higher than that of Buy and Hold Strategy in most of the cases indicating that the model has given better risk adjusted return. The Sharpe Ratio is also directly related to the t-statistic for measuring the statistical significance of the return. The Sharpe Ratio, when multiplied by the square root of 'n' (the number of returns used for the calculation) is equivalent to t-statistic.

$$t_{value} = \frac{\mu(r_i)}{\sigma_i}.\sqrt{n}$$

$$t_{value} = SR.\sqrt{n}$$

The t-statistic as defined above for the full financial series is calculated in table -7. Since t-statistics of all the three financial series are not only positive in but also statistically significant at 1% level, use of the artificial neural network based trading model may be considered as a better alternative to Buy and Hold Strategy.

5. Conclusion

In the study, investment decisions were taken using technical analysis based trading rules and tested on three stock index series in Indian stock market. Instead of relying on a single trading rule, the rules are combined using an artificial neural network model. The theoretical profits from the model are estimated and compared with profits obtainable from "Buy and Hold" strategy. It is observed that the risk-adjusted performance of the Neural Network based trading model is generally better than Buy and Hold strategy.

Like many previous studies, the present study also demonstrates that it is possible to earn positive return by using technical trading rules. However one of the major impediments of trading profit is transaction cost. The study is carried out taking a relatively low transaction cost of 0.05%, usually applicable for futures trading where trades are squared off without delivery. Wherever the delivery is involved, the brokerage fees are significantly higher. Thus investor has to pay more attention in minimizing transaction cost for trading success. In the study, only moving averages are used as inputs to the network. Many other indicators; both technical analysis indicators and fundamental analysis ratios can also be used as inputs to the artificial neural network model to improve the investment performance. Configuration of the neural network model and node relationships can also be altered for further development.

References

Bessembinder, H., Chan, K. (1995). The Profitability of Technical Trading Rules in the Asian Stock Markets. *Pacific-Basin Finance Journal*, 3 (2/3), 257-284.

Blume, Lawrence, David Easley & Maureen O.hara. (1994). Market Statistics and Technical Analysis: The Role of Volume, *Journal of Finance*, 49, 153-181.

Brock, W., Lakonishok, J. & Lebaron, B. (1992). Simple Technical trading rules and the stochastic properties of stock returns. *Journal of Finance*, 47, 1731-1764.

Gencay, Ramazan. (1998). The Predictability of Security Returns with Simple Technical Trading Rules. *Journal of Empirical Finance*, 5, 347-59.

Jegadeesh, Narasimhan & Sheridan Titman. (1993). Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency. *Journal of Finance*, 48, 65-91.

Jegadeesh, Narasimhan. (2000). Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation: Discussion. *Journal of Finance*, 55, 1765-70.

Kantardzic M. (2001). *Data Mining Concepts, Models, Methods, and Algorithms*. Wiley-Interscience.

Lo, Andrew W. & A. Craig MacKinlay. (1988). Stock Market Prices Do Not Follow Random Walks: Evidence from A Simple Specification Test. *Review of Financial Studies*, 1, 41-66.

Markowitz, Harry M. (1952). Portfolio selection, *Journal of Finance*, 7, 77-91.

Markowitz, Harry M. (1999). The early history of portfolio theory: 1600-1960, *Financial Analysts Journal*, 55, 5-16.

Neftci, Salih N. (1991). Naive Trading Rules in Financial Markets and Wiener-Kolmogorov Prediction Theory: A Study of .Technical Analysis. *Journal of Business*, 64, 549-71.

Ready, M. (1997). Profits from Technical Trading Rules, Working paper. *University of Wisconsin-Madison, Madison, WI.*

Table 1. Comparing profits of Neural Network model and Buy and hold strategy (For Nifty)

Year	No of Trading Days	Gross Profit	Transaction per Year	Transaction Cost	Net Profit	Profit from Buy-Hold Strategy
1998-1999	251	253	88	42	211	-87
1999-2000	254	-237	114	77	-314	471
2000-2001	251	1011	68	46	965	-397
2001-2002	247	308	80	44	265	1
2002-2003	251	243	96	50	192	-155
2003-2004	254	646	70	55	591	835
2004-2005	253	924	56	48	876	248
2005-2006	251	1124	60	75	1050	1406
2006-2007	249	826	68	118	708	160
2007-2008	250	3071	72	180	2890	1101
1999-2008	2511	8170	772	735	7435	3584

Table 2. Comparing profits of Neural Network model and Buy and hold strategy (For Junior-Nifty)

Year	No of Trading Days	Gross Profit	Transaction per Year	Transaction Cost	Net Profit	Profit from Buy-Hold Strategy
1998-1999	251	565	92	70	495	671
1999-2000	254	2445	86	125	2320	1433
2000-2001	251	2474	76	96	2378	-1970
2001-2002	247	748	70	49	699	64
2002-2003	251	509	82	60	449	-329
2003-2004	254	2231	66	79	2152	2205
2004-2005	253	2989	68	122	2867	868
2005-2006	251	2903	68	173	2730	2202
2006-2007	249	2351	88	287	2065	38
2007-2008	250	8252	68	331	7921	1394
1999-2008	2511	25468	764	1392	24076	6575

Table 3. Comparing profits of Neural Network model and Buy and hold strategy (For Defty)

Year	No of Trading Days	Gross Profit	Transaction per Year	Transaction Cost	Net Profit	Profit from Buy-Hold Strategy
1998-1999	251	394	158	61	332	-141
1999-2000	254	-147	190	104	-251	346
2000-2001	251	753	140	71	682	-361
2001-2002	247	269	152	60	209	-51
2002-2003	251	192	160	60	132	-89
2003-2004	254	650	118	66	585	754
2004-2005	253	152	148	101	51	168
2005-2006	251	487	140	134	353	1068
2006-2007	249	1745	138	188	1557	265
2007-2008	250	4707	140	286	4420	1142
1999-2008	2511	9201	1484	1130	8071	3101

Table 4. Comparison of Sharpe Ratio (For Nifty)

	Neural Network Model			Buy & Hold Strategy		
Year	Average Profit per Day	Standard Deviation of Daily Profit	Sharpe Ratio	Average Profit per Day	Standard Deviation of Daily Profit	Sharpe Ratio
1998-1999	0.84	17.51	0.048	-0.35	17.53	-0.020
1999-2000	-1.24	25.93	-0.048	1.86	25.86	0.072
2000-2001	3.85	26.34	0.146	-1.58	26.66	-0.059
2001-2002	1.07	14.62	0.073	0.00	14.66	0.000
2002-2003	0.77	10.25	0.075	-0.62	10.31	-0.060
2003-2004	2.33	22.46	0.104	3.29	22.38	0.147
2004-2005	3.46	26.87	0.129	0.98	27.11	0.036
2005-2006	4.18	26.66	0.157	5.60	26.42	0.212
2006-2007	2.84	60.53	0.047	0.64	60.68	0.011
2007-2008	11.56	101.36	0.114	4.40	102.24	0.043
1999-2008	2.96	42.25	0.070	1.43	42.43	0.034

Table 5. Comparison of Sharpe Ratio (For Junior-Nifty)

	Neural Network Model			Buy & Hold Strategy		
Year	Average Profit per Day	Standard Deviation of Daily Profit	Sharpe Ratio	Average Profit per Day	Standard Deviation of Daily Profit	Sharpe Ratio
1998-1999	1.97	32.37	0.061	2.67	32.42	0.082
1999-2000	9.14	82.69	0.110	5.64	82.93	0.068
2000-2001	9.48	70.59	0.134	-7.85	70.89	-0.111
2001-2002	2.83	21.55	0.131	0.26	21.75	0.012
2002-2003	1.79	17.80	0.100	-1.31	17.89	-0.073
2003-2004	8.47	42.86	0.198	8.68	43.04	0.202
2004-2005	11.33	61.42	0.185	3.43	62.47	0.055
2005-2006	10.88	58.00	0.188	8.77	58.61	0.150
2006-2007	8.29	122.72	0.068	0.15	123.24	0.001
2007-2008	31.68	233.00	0.136	5.57	235.40	0.024
1999-2008	9.59	95.95	0.100	2.62	96.53	0.027

Table 6. Comparison of Sharpe Ratio (For Defty)

	Neural Network Model			Buy & Hold Strategy		
Year	Average Profit per Day	Standard Deviation of Daily Profit	Sharpe Ratio	Average Profit per Day	Standard Deviation of Daily Profit	Sharpe Ratio
1998-1999	1.32	14.78	0.090	-0.56	14.84	-0.038
1999-2000	-0.99	20.75	-0.048	1.36	20.72	0.066
2000-2001	2.72	20.35	0.134	-1.44	20.56	-0.070
2001-2002	0.85	10.74	0.079	-0.21	10.78	-0.019
2002-2003	0.53	7.31	0.072	-0.36	7.37	-0.048
2003-2004	2.30	17.32	0.133	2.97	17.26	0.172
2004-2005	0.20	21.80	0.009	0.66	21.83	0.030
2005-2006	1.41	21.47	0.065	4.25	21.12	0.201
2006-2007	6.25	48.96	0.128	1.07	49.54	0.022
2007-2008	17.68	94.56	0.187	4.57	96.24	0.047
1999-2008	3.21	37.38	0.086	1.23	37.57	0.033

Table 7. Sharpe Ratio and t-statistic of Financial Series

Sl. No.	Financial Series	Average Profit per Day	Standard Deviation of Daily Profit	Sharpe Ratio	t-statistic
1	Nifty	2.96	42.25	0.070	3.51*
2	Junior-Nifty	9.59	95.95	0.100	5.01*
3	Defty	3.21	37.38	0.086	4.30*

* Significant at 1%.

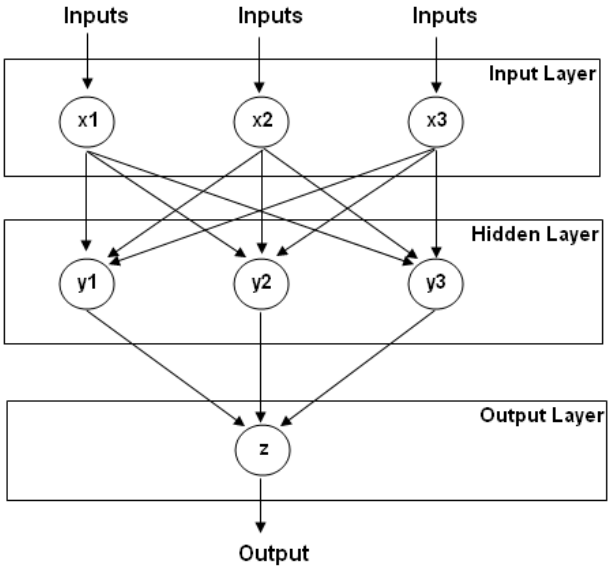

Figure 1. Schematic diagram of an Artificial Neural Network