

Prediction of Stock Market Index Using Neural Networks: An Empirical Study of BSE

R. Lakshman Naik*[1], B. Manjula[1], D. Ramesh[1], B. Sridhara Murthy[2], Prof. SSVN Sarma[3]
1. Department of Informatics, Kakatiya University, Warangal, A.P, India
2. Department of MCA, KITS, Warangal, Andhra Pradesh, India.
3. Department of CSE, Vaagdevi College of Engineering,Warangal, A.P, India
* E-mail of the corresponding author:lakshman.ramavathu@gmail.com

Abstract
Predicting stock data with traditional time series analysis has become one popular research issue. An artificial neural network may be more suitable for the task, because no assumption about a suitable mathematical model has to be made prior to forecasting. Furthermore, a neural network has the ability to extract useful information from large sets of data, which often is required for a satisfying description of a financial time series. Subsequently an Error Correction Network is defined and implemented for an empirical study. Technical as well as fundamental data are used as input to the network. One-step returns of the BSE stock index and two major stocks of the BSE are predicted using two separate network structures. Daily predictions are performed on a standard Error Correction Network whereas an extension of the Error Correction Network is used for weekly predictions. The results on the stocks are less convincing; nevertheless the network outperforms the naive strategy.
Keywords: - Prediction of stock, ECN, Backpropagation, Feedforward Neural Networks, Dynamic system.

1. Introduction

Different techniques are being used in the trading community for prediction tasks. In recent years the concept of neural networks has emerged as one of them. A neural network is able to work parallel with input variables and consequently handle large sets of data swiftly. The principal strength with the network is its ability to find patterns and irregularities as well as detecting multi-dimensional non-linear connections in data. The latter quality is extremely useful for modeling dynamical systems, e.g. the stock market. Apart from that, neural networks are frequently used for pattern recognition tasks and non-linear regression. An Error Correction Network (ECN) is built and implemented for an empirical study. Standard benchmarks are used to evaluate the network's ability to make forecasts. The objective of this study is to conclude whether an ECN could be successfully used as decision support in a real trading situation. The matters of buy-sell signals, transaction costs and other trading issues are not considered.

Neural networks can be applied to all sorts of financial problems, not only stock prediction tasks, also used for forecasts of yield curves, exchange rates, bond rates etc. The principal motivation for the neural network approach in stock prediction is twofold:

- Stock data is highly complex and hard to model; therefore a non-linear model is beneficial
- A large set of interacting input series is often required to explain a specific stock, which suites neural networks

This approach is also applied for the prediction task from the angle of economics. Such as stock forecasting to predict returns rather than actual stock prices. Another reason is that it facilitates stabilization of the model over a long period of time [1], prior to model.

Documentation is made ignoring needed data and importance is given to the data irrelevant for the problem. Frequently one has to handle missing data at recreant intervals or even data emerging discontinuous time series. A convenient way to work around these difficulties is by letting the network accept missing data.

2. Error Correction Networks

Most dynamical systems contain both an autonomous part and a part governed by external forces. Many times relevant external forces may be hard to identify or data could be noisy. As a consequence a correct description of the dynamics may be impossible to get. A remedy for a better model description is to use the previous model error as additional information to the system will take care of all the above data.

2.1. *Mathematical Description*

Following set of equations consisting of a state and output equations is a recurrent description of dynamic system is very general form for discrete time grids. A basic dynamical recurrent system depicted in Fig. 1, at time t, it

can be expressed as follows:

$$s_t = f(s_{t-1}, u_t) \quad \text{state transition} \qquad (1)$$

$$y_t = g(s_t) \quad \text{output equation} \qquad (2)$$

The state transition is a mapping from the pervious internal hidden state of the system s_{t-1} and the influence of external inputs u_t to the new state s_t. The output equation computes the observable output vector y_t.

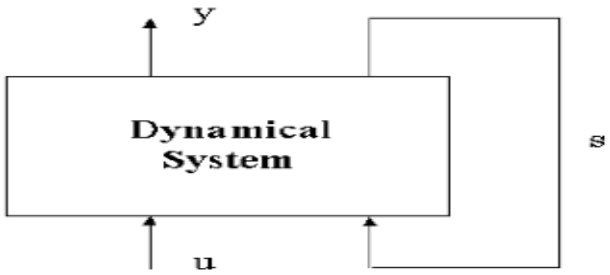

Figure 1: A dynamical system with input u, output y and internal state s. The identification of dynamical system using a discrete time description: Input $u_t \in R^k$ hidden states $s_t \in R^d$ and output $y_t \in R^n$

The system can be viewed as a partially observable autoregressive dynamic state s_t which also driven by external disturbances u_t. Without the external inputs the systems is called autonomous system[7].The task of indentify the dynamic system of equations 1 & 2, can then be stated as task to find (parameterized) functions f, g such that an average distanced measurement (Eq. 3) between the observed data y_t^d, t=1,2,……T and the computed data y_t of the model is minimal[5]

$$\min_{f,g} \leftarrow \frac{1}{T} \sum_{t=1}^{T} (y_t - y_t^d)^2 \qquad (3)$$

Where T is the number of patterns (Patterns are data points in a time series). If we ignore the dependency on s_{t-1} in the state transition equation by assuming $s_t = f(u_t)$ and $y_t = g(s_t) = g(f(u_t))$, we are back in the frame work where the neural network approach without recurrence i.e. a feedforward neural network, can solve the system identification task.

The Eq. 1 and 2 without external inputs u_t would represent an autonomous system. Let the observed model error at the previous time t-1 act as an additional input to the system. We get (y_t^d denotes observed data, y_t is the computed output and s_t describes the state.)

$$s_t = f(s_{t-1}, u_t, y_{t-1} - y_{t-1}^d) \qquad (4)$$

$$y_t = g(s_t) \quad (5)$$

The identification task of Eq. 1, 2 & 3, can be implemented as a neural network (denoted $\aleph$) then we get

$$s_t = \aleph(s_{t-1}, u_t, y_{t-1} - y^d_{t-1}; v) \quad (6)$$

$$y_t = \aleph(s_t; w) \quad (7)$$

By specifying the functions f and g as neural networks with parameter vectors v and w, we have transformed the system identification task of Eq. 3 into a parameter optimization problem

$$\min_{v,w} \leftarrow \frac{1}{T} \sum_{t=1}^{T} (y_t - y^d_t)^2 \quad (8)$$

The dynamic system described in Eq. 9 & 10, can be model by Error Correction neural network architecture as shown in fig 2. The weights are v = {A, B} and w = {C}. in general one may think of a matrix D instead of the identity matrix (id) between the hidden layer s_t this is not necessary because for linear output layer s_t can combines matrix A and D to a new matrix A' between the input s_{t-1} and the hidden layer. The output equation $\aleph(s_t; w)$ is realized as a linear function. It is straight forward to show by using an augmented inner state vector that this is nota functional restriction.

$$s_t = \tanh(As_{t-1} + Bu_t + D(Cs_{t-1} - y^d_{t-1})) \quad (9)$$

$$y_t = Cs_t \quad (10)$$

The term Cs_{t-1} recomputed the last output y_{t-1} and compares to the observed data y^d_{t-1}. The matrix transformation D is necessary in order to adjust different dimensionalities in the state transition equation.

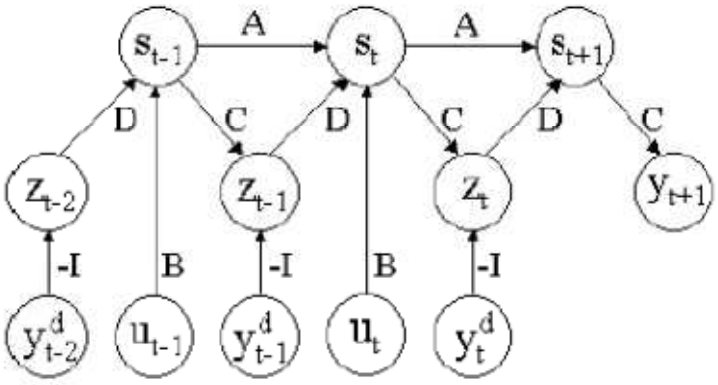

Figure 2: The error correction neural network.

The identification of above model is ambiguously creating the numerical problem, because autoregressive structures i.e. the dependency of s_t on s_{t-1}, could either be coded in matrix A or DC. On this problem, one may

argue to transform Eq. 9, well define form (Eq. 11) utilizing $\dot{A} = A + DC$

$$s_t = \tanh(\dot{A} s_{t-1} + Bu_t + Dy^d_{t-1})) \qquad (11)$$

Adding a non-linearity is a measure to avoid this problem [2]. This yield

$$s_t = \tanh(As_{t-1} + Bu_t) + D\tanh(Cs_{t-1} - y^d_{t-1})) \qquad (12)$$

$$y_t = Cs_t \qquad (13)$$

Distinguishing external inputs u_t affecting the state transition s_t from target inputs y^d_t is important. I denote the fixed identity matrix. As a consequence the target values of output clusters $z_{t-\tau}$ $\tau = 0,1,2$, are zero. Only the difference between y_t and y^d_t influences s_t. The ECN offers forecasts based on the modelling of the recursive structure (matrix A), the external forces (matrix B) and the error correcting part (matrices C and D). The error correcting part can also be viewed as an external input similar to u_t.

2.2. Variants and Invariants

Predicting a high-dimensional dynamical system is difficult. A way of reducing the complexity of the task is to separate the dynamics into time variant and invariant structures. Let the system forecasts the variants then eventually this will combine this forecast with the unchanged invariants. This can be done by connecting the standard ECN (Fig. 2) to a compression-decompression network shown in Fig. 3. Matrix E separates variants from invariants while matrix F reconstructs the dynamics. The actual forecasting is coded in G, [3].

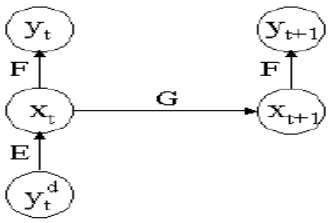

Figure 3: Separation of variants and invariants

3. Training The Proposed Model

3.1. Backpropagation

After the description of the network structure, training set-up has to be settled. As previously mentioned the overall objective in training is to minimise the discrepancy between real data and the output of the network. This principle is referred to as supervised learning (The other basic class of learning paradigms is unsupervised or self-organized learning. One well known learning method in this class is the Self Organizing Map (SOM)). In a step-by-step manner the error guides the network in the direction towards the target data. The backpropagation algorithm belongs to this class and can be described as "an efficient way to calculate the partial derivatives of the network error function with respect to the weights" [2].

3.2. Learning Algorithm

The backpropagation algorithm supplies information about the gradients. However, a learning rule that uses this information to update the weights efficiently is also needed. A weight update from iteration k to k + 1 may look like

$$w_{k+1} = w_k + \eta . d_k \qquad (14)$$

Where d_k describes the search direction and η the learning rate (or step length). Issues that have to be

addressed are how to determine (i) the search direction, (ii) the learning rate and (iii) which patterns to include. A familiar way of determining the search direction d_k is to apply the gradient descent which is a relatively simple rule [4]. The major drawback though is that learning easily is caught in a local minimum. To avoid this hazard the vario-eta algorithm can be chosen as learning rule [5, 6]. Basically it is a stochastic approximation of a Quasi-Newton method. In the vario-eta algorithm a weight-specific factor, β, is related to each weight. For an arbitrary weight, the j^{th} weight-specific factor β is defined according to Eq. 15.

$$\beta^{j} = \frac{1}{\sqrt{\sum_{t=1}^{N} \left(\frac{\partial E_t}{\partial w^{j}} - \overline{E} \right)^2}} \qquad (15)$$

$$\overline{E} = \frac{1}{N} \sum_{t=1}^{N} \frac{\partial E_t}{\partial w^{j}}$$ Where $\overline{E}$ denotes the E average and E is error function

Assuming there are p weights in the network. The search direction is determined by multiplying each component of the negative gradient with its weight-specific factor,

$$d_k = \begin{bmatrix} \beta^{j} & 0 & 0 \\ 0 & . & 0 \\ 0 & 0 & \beta^{p} \end{bmatrix} . \nabla E \qquad (16)$$

Above E denotes the error function and N the number of patterns. A benefit with the vario-eta rule is that weight increments $\eta . d_k$ become non-static. This property implies a potentially fast learning phase in [2]. Concerning a reasonable value of the learning rate η. The learning rate is many times determined on an ad hoc basis.

Regarding pattern selection, a stochastic procedure can be used. This simply means that the gradient ∇E^{M} of a subset M of all patterns are used as an approximation of the true gradient ∇E according to

$$\nabla E^{M} = \frac{1}{|M|} \sum_{t \varepsilon M} \nabla E_t \qquad (3.17)$$

Where |M | denotes the number of elements of M. M is subset network

M can be composed in several ways. In our empirical study selection of M is made with equal probability, a predefined number of patterns (less than 10 percent of the training data) to represent M. The gradient ∇E^{M} of Eq. 17 was computed and used as input to Eq. 16. Once all weights were updated, out of the remaining training patterns a new subset was picked for the next iteration. It is also noted that recent patterns are considered significant one, may prefer a non-uniform probability distribution, where it is more likely to choose a recent pattern [6].

3.3. Cleaning

The dilemma of overfitting is deeply rooted in neural networks. One way to suppress overfitting is to assume input data not to be exact (the case in the field of financial analysis). The total error of pattern t can be split into two components associated with the weights and the erroneous input respectively. The corrected input data, $\tilde{x}_t$, can be expressed as

$$\tilde{x}_t = x_t + \Delta x_t \qquad (18)$$

Where x_t is the original data and Δx_t a correction vector. During training the correction vector must be updated in parallel with the weights. To this end, the output target difference i.e. the difference in output from using original and corrected input data has to be known which is only true for training data. Accordingly, the

model might be optimised for training data but not for generalisation data because the latter has a different noise distribution and an unknown output target difference. To work around this disadvantage the model $\widetilde{x}_t$ is composed according to

$$\widetilde{x}_t = x_t + \Delta x_t - \delta \qquad (19)$$

δ is exactly one element drawn at random from $\{\Delta x_i$ i = 1, . . . , T $\}$ (memorised correction vectors from training data). A composition with an additional noise term (Eq. 19) benefits from distribution properties which is desirable for generalisation, [5].

The input modification "cleaning with noise" described above helps the network to concentrate on broader structures in data. With this extent the model is prevented from establishing false causalities with considerable effect.

3.4. Stopping Criteria

It is also underling to study the number of epochs (An epoch is completed when all training patterns have been read in exactly once) are needed for a network to be trained. Mainly two paradigms exist, late and early stopping. Late stopping means that the network is trained until a minimum error on the training set is reached, i.e. the network is clearly overfitted. Then different techniques are used to exterminate nodes in the network (known as pruning). By doing so eventually good generalisation ability is reached.

The concept of early stopping is a way of avoiding overfitting. During learning the progression is monitored and training is terminated as soon as signs of overfitting appear. A clear advantage with early stopping is that the time of training is relatively short. On the downside it is hard to know when to stop.

3.5. Error Function

When modelling one also has to be aware of outliers in data. Outliers typically appear when the economic or political climate is unstable or unexpected information enters the market. By picking an appropriate error function the impact of outliers can be restrained.

$$\frac{1}{a}\ln(\cosh(a(o_i - t_i))) \qquad (20)$$

The ln(cosh(·)) error function is a smooth approximation of the absolute error function $|o_i - t_i|$. o_i denotes the response from output neuron i and t_i the corresponding target. a ∈ [3, 4] has proven to be suitable for financial applications [5].

Compared to a quadratic error function the ln(cosh) error function has a reminiscent behaviour in the region around zero but not for large positive or negative values as we see in Fig. 4. The advantage with this function when modelling financial data is that a large difference in output and target yields a limited and more reasonable error.

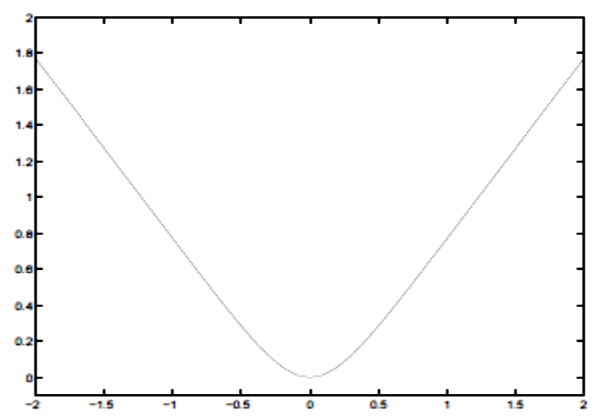

Figure 4: The lncosh error function of Eq. 20, a = 3.

4. Evaluation of the Study

A crucial part of financial forecasting is the evaluation of the prediction algorithm. Several performance measures are widely used, but a performance measure in itself is not sufficient for a satisfying evaluation. Relevant benchmarks are also needed. A benchmark is basically a different prediction algorithm used for comparison. In this study, the quality of a prediction algorithm is judged with the help of existing offered data.

4.1. Benchmarks

Any prediction algorithm claiming to be successful should outperform the naive predictor defined as

$$\hat{y}_{t+1} = y_t \tag{21}$$

Where y_t is the current value of the stock and $\hat{y}_{t+1}$ the predicted value one time-step into the future. The Eq. 21 states that the most intelligent suggestion of tomorrow's price is today's price which is a direct consequence of the Efficient Market Hypothesis (EMH), (The EMH states that the current market price reflects the assimilation of all information available. Therefore no prediction of future changes in the price can be made given this information). In the empirical study a comparison to the naive prediction of returns was made. The definition is

$$\hat{R}_{t+1} = R_t \tag{22}$$

Where R_t is the last known return and $\hat{R}_{t+1}$ the predicted one-step return. For a prediction algorithm with incorporated buy and sell signals it could be useful to do a comparison with the buy-and-hold return R_b. This strategy expresses the profit made when making an investment at the start of a time period and selling n time-steps into the future, i.e.

$$R_b = 100.\frac{y_{t+n} - y_t}{y_t} \tag{23}$$

A comparison to the buy-and-hold strategy simply gives an indication of the quality of the signals. Is it more profitable to be a "passive" investor?

4.2. Performance Measures

Three frequently used measures, namely hit rate, return on investment and realised potential, and are defined below. Begin with the hit rate H_R.

$$H_R = \frac{\left|\{t \left| R_t^k \hat{R}_t^k > 0, t, ..., N\}\right.\right|}{\left|\{t \left| R_t^k \hat{R}_t^k \neq 0, t, ..., N\}\right.\right|} \tag{24}$$

Where $R_t^k \hat{R}_t^k$ is the actual (predicted) k-step return at time t. (According to the definition the k-step return $R_t^k = (y_t - y_{t-k})/(y_{t-k})$). The norm simply is the number of elements in the series. Eq. 24 indicates how often the algorithm produces a correct prediction. In this context a prediction is correct when the direction of the stock k time-steps into the future is successfully predicted. The return on investment ROI takes into account not only the correctness of the sign, but also the quantity of the actual return. The definition is

$$ROI = \sum_{t=1}^{T} R_t.sign(\hat{R}_t) \tag{25}$$

Finally, in Eq. 26 the definition of realised potential RP is given.

$$RP = \frac{\sum_{t=1}^{T} R_t .sign(\hat{R}_t)}{\sum_{t=1}^{T} |R_t|} \qquad (26)$$

The realised potential states how large part of the total movement (upwards and downwards) the prediction algorithm successfully identifies.

5. Empirical Study: Forecasting the INR/USD Fx-Rate

In the empirical study well traded stocks with a reasonable spread were considered. Mainly because a large spread may be fatal if one wishes to incorporate the predictions in real trading. Hence a decision is further to make one-day predictions (using daily data) of the Bombay stock exchange (BSE), AxisBank B and Cipla B with a standard ECN. Also one-week predictions (using weekly data) of AxisBank B and Cipla B were performed on an ECN separating variants and invariants. Basically, one assumes that certain time invariant structures can be identified and learned quickly. This means that the latter part of the training is performed with some weights frozen. The occurrence of invariant structures could prove to be more evident in a weekly compared to a daily model, because patterns originate from the same day of the week in a weekly model.

5.1. Data Series

For all predictions the following four time series were used as raw input:

- Closing price y (price of the last fulfilled trade during the day)
- Highest price paid during the day, y_H
- Lowest price paid during the day, y_L
- Volume V (total amount of traded stocks during the day)

Additionally, external time series served as input. Table 1 gives a summary of time series considered to have a significant impact on the behaviour of the AxisBank B and Cipla B. Regarding the BSE prediction, another set of external inputs was used. Table 2 gives the full list. All data used in the modeling was acquired from the daily service provider E-Eighteen.com and yahoo finance

Table 1: external time series used predicting AxisBank B and Cipla B.

Stock Model Input	
SENSEX	NIFTY
6 months interest rate Bombay	1 year interest rate Bombay
Bombay INR/USD FX - rate	Bombay INR/USD FX - rate

Table 2: external time series used predicting the BSE.

Index Model Input	
BSE- IT (Infotech)	BSE- AUTO
BSE-BANK	BSE-OILGAS
6 months interest rate Bombay	1 year interest rate Bombay
SENSEX	NIFTY
Bombay INR/USD FX - rate	Bombay INR/USD FX - rate

5.2. Technical Considerations

Prior to each training session relevant data series were transformed and preprocessed in different ways. For all external time series (Tab. 1 and 2) we calculated the normalized one-step return R_t is calculated. The Gaussian volume was computed in a running window of 30 days and 12 weeks for daily and weekly predictions respectively. On inputs y, y_H and y_L, we was applied the log-return. The definition is given in Eq. 27. For small changes the log-return is similar to the one-step return R_t.

$$R(\log) = \frac{y_t}{y_{t+1}} \qquad (27)$$

Data were also divided into three subsets: a training set, a validation set and a generalisation set. Roughly, half of all patterns available were used for training and one-quarter each for validation and generalisation. The generalisation period ran over 12 months for both the daily and the weekly model. (In the daily model some training sessions were performed on a shorter generalisation set due to missing data.). As error function we used the ln(cosh) function of Eq. 20 with a=3 is used. The standard tanh(·) function served as activation function.

5.3. Training Procedure

Weights were initialized randomly (uniformly distributed in the interval [−1, 1]). The ECN was trained (on the training set) for 500 and 1000 epochs in the daily model and weekly model respectively. It is noticed that this was enough for the cleaning error (correction to the net inputs) to stabilize. After training, weights associated with the best performance (in terms of hit rate) on the validation set were selected and applied to the generalisation set to get the final results.

5.4. Implementation

Neuro Dimension provide a software for neural computing, NSNN (Neuro Solutions for Neurel Networks)is designed to build ANN, based on which mimic the learning process of the brain in order to extract patterns from historical data. Once these files are loaded into the software it is possible to start training.

6. Experimental Results

In the following sections, results from the empirical study are presented. Tables are based on generalisation data. The naive predictions of returns constitute benchmark for each prediction model. Values of hit rate (HR) and realised potential (RP) using the ECN and the naive strategy are given in tables below.

6.1. Daily Predictions

Table 3: Daily predictions of BSE. Hit rate and realised potential using the ECN and the naive predictor.

1-Day forecast (%)				
Period	HR ECN	RP ECN	HR navie	RP navie
Jan.05 to Dec. 05	52.4	10.8	54.7	22.3
Jan. 06 to Dec. 06	59.1	32.4	52.8	12.5
Jan. 07 to Dec. 07	54.5	11	51.4	8.9
Jan. 08 to Dec. 08	57.9	24.5	50	1.7
Jan. 09 to Dec. 09	60.1	26.1	53.9	12
Jan. 10 to Dec. 10	57.9	24.4	53.5	18.3
Jan. 11 to Dec. 11	56.3	17.8	52	15.2
Jan. 12 to Jun. 12	56.2	21	48.5	4.3
Mean	56.8	21	52.1	11.9

Table 4: Daily predictions of Axis Bank B. Hit rate and realized potential using the ECN and the naive predictor.

1-Day forecast (%)				
Period	HR ECN	RP ECN	HR navie	RP navie
Jan.05 to Dec. 05	50.7	5.1	42.4	-5.8
Jan. 06 to Dec. 06	51.2	9.3	41.7	2.1
Jan. 07 to Dec. 07	56.3	30.8	43.3	-7.3
Jan. 08 to Dec. 08	51.2	15.2	39.8	-6.4
Jan. 09 to Dec. 09	55.7	17.8	48.2	4.5
Jan. 10 to Dec. 10	48	0	46.5	-0.4
Jan. 11 to Dec. 11	53.5	3.7	40.7	-0.5
Jan. 12 to Jun. 12	47.6	10.1	48	22.6
Mean	51.8	11.5	43.8	1.1

Table 5: Daily predictions of Cipla B. Hit rate and realised potential using the ECN and the naive predictor.

1-Day forecast(%)				
Period	HR ECN	RP ECN	HR navie	RP navie
Jan.05 to Dec. 05	44.9	9.7	41.3	12.8
Jan. 06 to Dec. 06	44.1	-2.5	47.6	18.4
Jan. 07 to Dec. 07	52.4	21	45.7	22.3
Jan. 08 to Dec. 08	46.5	13.8	39	5.4
Jan. 09 to Dec. 09	44.5	1.3	42.5	5.4
Jan. 10 to Dec. 10	47.8	15	45.8	12.2
Jan. 11 to Dec. 11	44.9	7.8	42.9	10.6
Jan. 12 to Jun. 12	40.3	-2.5	37.9	1.5
Mean	45.7	8	42.8	11.1

Weekly Predictions

Table 6: Weekly predictions of Axis bank B. Hit rate and realised potential using the ECN and the naive predictor.

1-week forecast (%)				
Period	HR ECNN	RP ECNN	HR navie	RP
Jan. 06 to Dec. 06	60	25.9	44.4	-18.8
Jan. 07 to Dec. 07	48.9	22.1	44.4	0.3
Jan. 08 to Dec. 08	47.8	-8	50	18.9
Jan. 09 to Dec. 09	62.2	25.7	55.6	-19
Jan. 10 to Dec. 10	51.1	-0.9	60	2.9
Jan. 11 to Dec. 11	57.8	-6.5	55.6	-1
Jan. 12 to Jun. 12	42.2	-12.9	42.2	-34.1
Mean	52.9	6.5	50.3	-7.3

Table 7: Weekly predictions of Cipla B. Hit rate and realised potential using the ECN and the naive predictor.

1-week forecast (%)				
Period	HR ECN	RP ECN	HR navie	RP navie
Jan. 06 to Dec. 06	66.7	47.7	46.7	12.7
Jan. 07 to Dec. 07	60	32.2	44.4	-33
Jan. 08 to Dec. 08	51.1	21.3	40	-9.7
Jan. 09 to Dec. 09	45.7	13.3	45.7	-17.8
Jan. 10 to Dec. 10	57.8	33.4	42.4	-17.8
Jan. 11 to Dec. 11	60	18	48.9	-0.5
Jan. 12 to Jun. 12	55.6	15.7	60	12.5
Mean	56.7	25.9	46.8	-7.7

7. Conclusion

The most crucial issue of stock prediction is how to get stability over time in this respect neither the daily nor the weekly model is not optimized and refinements have to be made. Nevertheless, there is no doubt about neural networks potential in a trading environment. As already exhibited in previous section, the ECN occasionally shows good results.

Intuitively, one may think that the weight initialization phase is decisive for the outcome of the predictions. Theoretically, this should not be the case though due to the stochastisity in the selection of patterns during training. To confirm this notion a training scheme where was set up the net to initialized with the "best" (in terms of hit rate) generalisation weights from the previous training period. The results gave indications of an even negative impact (on the hit rate) using a biased initialization compared to a random one. This phenomenon typically illustrates the challenge faced when trying to validate a dynamical system for a longer period of time. Previously gained information about the system may be hard to utilize successfully in future time perspective.

A further development is to combine one-day predictions with multi-day predictions. To estimate the relevance of a possible detected trend a weighted sum could be calculated, where weights are associated with the one-day forecast, two-day forecast and so on in descending order. Eventually, the aggregate is used to make an assessment of whether the trend id likely to turn out as predicted or not. Moreover, the forecasts of a committee of models could be averaged. This will bring about a reduction of the forecast variance, thus a more reliable output

References

[1] Hellström T. A Random Walk through the Stock Market, Licentiate Thesis, Department of Computing Science, Ume a University, Sweden, 1998.
[2] Grothmann R. Multi-Agent Market Modeli ng based on Neural Net works, Ph.D. Thesis, Faculty of Economics, University of Bremen, Germany, 2002.
[3] Zimmermann H.-G. System Identification & Forecasting by Neural Networks. Principles, Techniques, Applications, Compilation of material presented to a conference held at Mälardalen University, September 2003.
[4] Heath M.T. Scientific Computing- An Introductory Survey, McGraw Hill, New York, 1997.
[5] Neuneier R. and Zimmermann H.-G. How to Train Neural Networks. In Orr G.B. and Müller K.-R. (Eds.) Neural Networks: Tricks of the Trade, pp. 373-423, Springer, Berlin, 1998.
[6] NSNN User Manual. Version 6.1, Neuro Dimensions, NewYork 2012. http://www.neurosolutions.com/products/ns/
[7] Haykin S. Neural Networks. A Comprehensive Foundation, Macmillan College Publishing, New York, 1994.

Author:
R. Lakshman Naik received his B.Tech. (Electronics and Communication Engineering) from Sree Sarathi Institute of Engineering and Technology, Jawaharlal Nehru Technological University (JNTUH), Krishna, Andhra Pradesh, India and M.Tech.(Computer Science and Engineering) from Balaji Institute of Technology and Sciences, JNTUH, Warangal, A.P, India. He served as a Systems Engineer in Beteel Business Solution and Wipro Technologies, Hyderabad. Now he is working as Senior Systems Engineer in Unit4System Pvt. Ltd. He is a member of IAENG, IAEME, AIRCC and various organizations. He has Publications more than 10 papers in area of computer networks, Data mining and Artificial neural networks.

B. Manjula received her BCA (Computer Application) from Osmania University, Hyderabad, A.P, India and M.Sc. (Information System) from Osmania University, Hyderabad, A.P, India. She served as an Academic Consultant in Department of Computer Science, Osmania Univesrsity, Hyderabad, A.P., India. And now she is working as Assistant professor in Department of Informatics, Kakatiya University, Warangal, A.P, India. She is a Member of IAENG and IAEME. She has Publications more than 11 papers in area of computer networks, Data mining and neural networks.

D. Ramesh received his B.Tech. (Computer Science and Engineering) from Srineedhi Institute of Science and Technology, JNTU, Hyderabad, A.P, India and M.Tech. (Computer Science) from School of Information Technology, JNTU, Hyderabad, A.P, India. He served as an Academic Consultant in JNTU, Hyderabad, A.P, India. Now he is working as Assistant Professor in Department of Informatics, Kakatiya University, Warangal,

A.P, India. He has Publications more than 4 papers in area of Data mining and neural networks.

B. Sridhara Murthy received his MCA (Computer Application) from Osmania University, Hyderabad, A.P, India and M.Tech (Computer Science and Engineering) from Balaji Institute of Technology and Sciences, JNTUH, Warangal, A.P, India. He served as an Assistant professor in Department of CSE, Balaji Institute of Technology and Sciences, Warangal, A.P., India. And now he is working as Assistant professor in Department of MCA, KITS, Warangal, A.P, India. He is a Member of ISTE. His current research interests in area of computer networks and business data mining.

S.S.V.N. Sarma is a Professor and working as a Dean, Placement cell officer in Vaagdevi College of Engineering, Warangal (A.P), India. He has more than 45 years of experience in teaching area. He served as Head in Department of informatics, Chairman in Board of Studies, and Dean in Faculty of Science, Kakatiya University, Warangal, A.P, India. His current research interests in the area of Data mining, Distributed systems, Mobile computing, Computer networks, Computer security, and Performance evaluation. He has published and presented more than 120 articles. He has organized several workshops, delivered number of lecture at major and minor conferences, and serves as Editor of several International/National journals and magazines.