Selvamuthu, Dharmaraja; Kumar, Vineet; Mishra, Abhishek

Article

Indian stock market prediction using artificial neural networks on tick data

Financial Innovation

Provided in Cooperation with:
Springer Nature

Suggested Citation: Selvamuthu, Dharmaraja; Kumar, Vineet; Mishra, Abhishek (2019) : Indian stock market prediction using artificial neural networks on tick data, Financial Innovation, ISSN 2199-4730, Springer, Heidelberg, Vol. 5, Iss. 1, pp. 1-12, https://doi.org/10.1186/s40854-019-0131-7

This Version is available at:
https://hdl.handle.net/10419/237162

Selvamuthu et al. Financial Innovation (2019) 5:16
https://doi.org/10.1186/s40854-019-0131-7

Financial Innovation

CASE STUDY Open Access

Indian stock market prediction using artificial neural networks on tick data

Dharmaraja Selvamuthu*, Vineet Kumar and Abhishek Mishra

* Correspondence:
dharmar@maths.iitd.ac.in
Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India

Abstract

Introduction: Nowadays, the most significant challenges in the stock market is to predict the stock prices. The stock price data represents a financial time series data which becomes more difficult to predict due to its characteristics and dynamic nature.

Case description: Support Vector Machines (SVM) and Artificial Neural Networks (ANN) are widely used for prediction of stock prices and its movements. Every algorithm has its way of learning patterns and then predicting. Artificial Neural Network (ANN) is a popular method which also incorporate technical analysis for making predictions in financial markets.

Discussion and evaluation: Most common techniques used in the forecasting of financial time series are Support Vector Machine (SVM), Support Vector Regression (SVR) and Back Propagation Neural Network (BPNN). In this article, we use neural networks based on three different learning algorithms, i.e., Levenberg-Marquardt, Scaled Conjugate Gradient and Bayesian Regularization for stock market prediction based on tick data as well as 15-min data of an Indian company and their results compared.

Conclusion: All three algorithms provide an accuracy of 99.9% using tick data. The accuracy over 15-min dataset drops to 96.2%, 97.0% and 98.9% for LM, SCG and Bayesian Regularization respectively which is significantly poor in comparison with that of results obtained using tick data.

Keywords: Neural Networks, Indian Stock Market Prediction, Levenberg-Marquardt, Scale Conjugate Gradient, Bayesian Regularization, Tick by tick data

Introduction

A stock market is a platform for trading of a company's stocks and derivatives at an agreed price. Supply and demand of shares drive the stock market. In any country stock market is one of the most emerging sectors. Nowadays, many people are indirectly or directly related to this sector. Therefore, it becomes essential to know about market trends. Thus, with the development of the stock market, people are interested in forecasting stock price. But, due to dynamic nature and liable to quick changes in stock price, prediction of the stock price becomes a challenging task. Stock markets are mostly a non-parametric, non-linear, noisy and deterministic chaotic system (Ahangar et al. 2010).

As the technology is increasing, stock traders are moving towards to use Intelligent Trading Systems rather than fundamental analysis for predicting prices of stocks, which helps them to take immediate investment decisions. One of the main aims of a trader is to predict the stock price such that he can sell it before its value decline, or

buy the stock before the price rises. The efficient market hypothesis states that it is not possible to predict stock prices and that stock behaves in the random walk. It seems to be very difficult to replace the professionalism of an experienced trader for predicting the stock price. But because of the availability of a remarkable amount of data and technological advancements we can now formulate an appropriate algorithm for prediction whose results can increase the profits for traders or investment firms. Thus, the accuracy of an algorithm is directly proportional to gains made by using the algorithm.

Case description

There are three conventional approaches for stock price prediction: technical analysis, traditional time series forecasting, and machine learning method. Earlier classical regression methods such as linear regression, polynomial regression, etc. were used to predict stock trends. Also, traditional statistical models which include exponential smoothing, moving average, and ARIMA makes their prediction linearly. Nowadays, Support Vector Machines (Cortes & Vapnik, 1995) (SVM) and Artificial Neural Networks (ANN) are widely used for the prediction of stock price movements. Every algorithm has its way of learning patterns and then predicting. Artificial Neural Network (ANN) is a popular and more recent method which also incorporate technical analysis for making predictions in financial markets. ANN includes a set of threshold functions. These functions trained on historical data after connecting each other with adaptive weights and they are used to make future predictions. (Trippi & Turban, 1992; Walczak, 2001; Shadbolt & Taylor, 2002) (Kuan & Liu, 1995) investigated the out-of-sample forecasting ability of recurrent and feedforward neural networks based on empirical foreign exchange rate data (Kuan & Liu, 1995). In 2017, Mehdi Khashei and Zahra Haji Rahimi evaluated the performance of series and parallel strategies to determine a more accurate one using ARIMA and MLP (Multilayer Perceptron) (Mehdi & Zahra, 2017).

Artificial neural networks have been used widely to solve many problems due to its versatile nature. (Samek & Varachha, 2013) (Yodele et al., 2012), presented a hybridized approach, i.e., a combination of the variables of fundamental and technical analysis of stock market indicators to predict future stock prices to improve the existing methods, (Yodele et al., 2012) (Y Kara & A Boyacioglu, 2011) discussed stock price index movement using two models based on Artificial Neural Network (ANN) and Support Vector Machine (SVM). They compared the performances of both the models and concluded that the average performance of the ANN model was significantly better than the SVM model. (Y Kara & A Boyacioglu, 2011) (Qi & Zhang, 2008) investigated the best modeling of trend time series using Neural Network. They used four different approaches, i.e., raw data, raw data with a time index, de-trending and differencing for modeling various trend patterns and concluded Neural Network gives better results (Qi & Zhang, 2008). H.K. Cigizoglu, (2003) discussed the application of ANN forecasting, estimation and extrapolation of the daily flow data belonging to the rivers in the East Mediterranean region of Turkey. In their study, they found that ANN provides a better fit to the data than conventional methods (Cigizoglu, 2003). ANN can consider as a computation or a mathematical model which is inspired by the functional or structural characteristics of biological neural networks. These neural networks are developed in such a way that it can extract patterns from noisy data. ANN first train a system using

a large sample of data known as training phase then it introduces the network to the data which was not included in the training phase, this phase known as validation or prediction phase. The sole motive of this procedure is to predict new outcomes. (Bishop, 1995) This idea of learning from training and then predicting outcomes in ANN comes from the human brain which can learn and respond. Thus ANN has been used in many applications and is proven successful in executing complex functions in a variety of fields (Fausett, 1994).

The data used in this case study is tick data of Reliance Private Limited from period 30 NOV 2017 to 11 JAN 2018 (excluding holidays). There are roughly 15,000 data points per day. The dataset used contains approximately 430,000 data points. The data obtained from Thomson Reuter Eikon database[1] (This database is a paid product of Thomson Reuter). Each tick refers to the change in the price of the stock from trade to trade. The stock price at the start of every 15 min extracted from the tick data. This represents the secondary dataset on which same algorithms have run. In this study, we have made predictions on Tick Data, and 15-min Data using the same neural networks and their results are compared.

Discussion and evaluation

In this study, we have used variations of ANN to predict the stock price. But the efficiency of forecasting by ANNs depends upon the learning algorithm used to train the ANN. This paper compares three algorithms, i.e., Levenberg-Marquardt (LM), Scaled Conjugate Gradient and Bayesian Regularization. As shown in Fig. 1, neural networks with 20 hidden layers and a delay of 50 data points are used. Thus, each prediction is made using the last 50 values.

Theory of Levenberg-Marquardt

The Levenberg-Marquardt algorithm was developed to approximate the second-order training speed to avoid the computation of the Hessian matrix, and used for solving a non-linear least square problem. The Hessian matrix can be estimated if the performance function is in the form of a sum of squares by

$$H = J^T J \tag{1}$$

Equation **(1)** is used to avoid heavy computation of hessian matrix as it can be calculated using Jacobian matrix.

The gradient is calculated in **(2)**, which is first order derivative of total error function and used for updating weights in **(4)**

$$g = J^T e \tag{2}$$

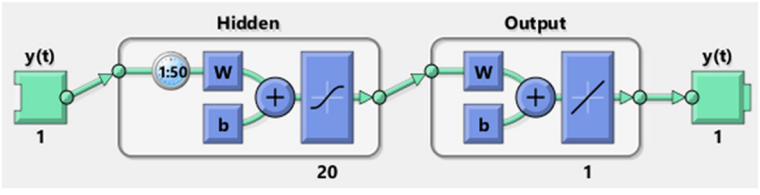

Fig. 1 Neural Network Structure

where $\boldsymbol{J}$ is the Jacobian matrix and $\mathbf{e}$ is a vector of network errors. All the first derivatives which correspond to the network errors with respect to biases and weights contained in $\boldsymbol{J}$. Keeping in mind the end goal to ensure that the approximated Hessian matric $\boldsymbol{J^T J}$ is invertible, Levenberg– Marquardt calculation acquaints another approximation of Hessian matrix:

$$\boldsymbol{H = J^T J + \mu I} \tag{3}$$

where $\boldsymbol{\mu}$ is scaler and $\boldsymbol{I}$ is identity matrix. By combining **(2)** and **(3)** update rule for the Levenberg-Marquardt algorithm is as the Newton-like update:

$$\boldsymbol{x_{k+1} = x_k - \left[J^T J + \mu I\right]^{-1} J^T e} \tag{4}$$

If the value of the scalar $\boldsymbol{\mu}$ is zero, this algorithm will be similar to Newton's method which uses Hessian matrix approximation. If the scalar $\boldsymbol{\mu}$ becomes large, this algorithm will be similar to gradient descent with small step size. But, Newton's method is much closer and faster near an error minimum. So, the primary objective is to shift toward Newton's method as fast as possible. Thus, decreasing $\boldsymbol{\mu}$ after each successful step leads to trimming of the performance function. $\boldsymbol{\mu}$ will increase only when there is any improvement in performance function at any tentative step as shown in Fig. 2. Therefore, at each iteration, the performance function is always reduced.

One of the significant merits of the LM approach is that it performs similarly to gradient search and Newton method for large values of $\boldsymbol{\mu}$ and small values of $\boldsymbol{\mu}$ respectively. The LM algorithm merges the best attributes of the steepest-descent algorithm

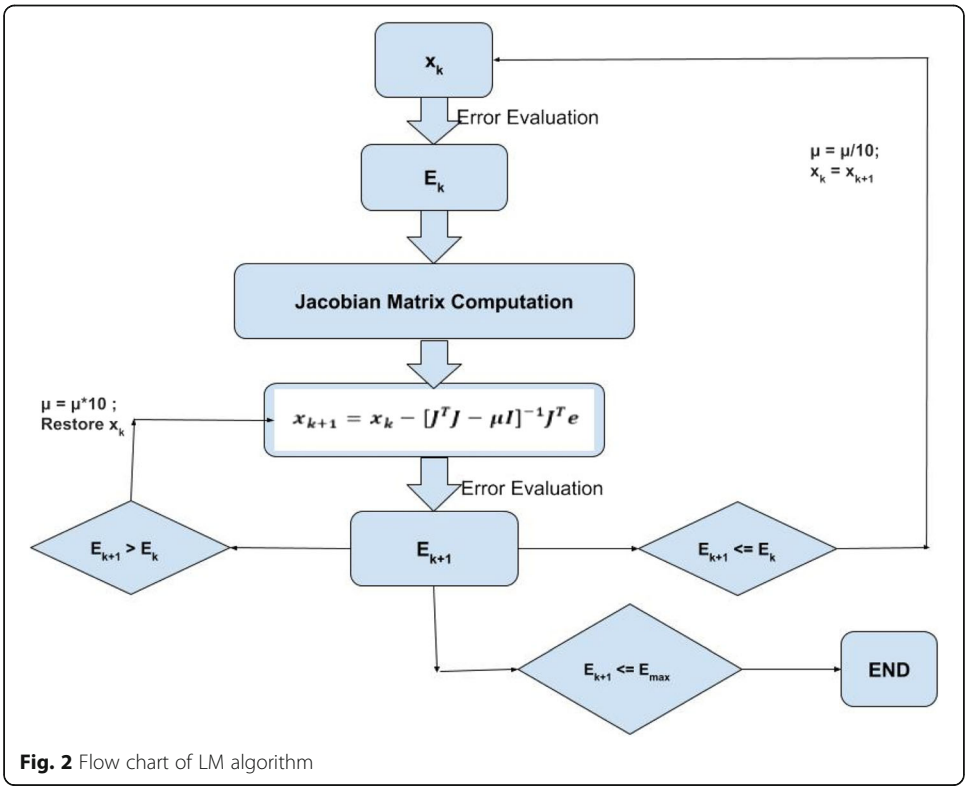

Fig. 2 Flow chart of LM algorithm

and the Gauss-Newton technique. Also, many of their limitations avoided. Specifically, this algorithm handles the problem of slow convergence efficiently (Hagan & Menhaj, 1994).

Theory of scaled conjugate gradient

In the backpropagation algorithm, the weights are adjusted in the steepest descent direction (negative of the gradient) because the performance function decreases rapidly in this direction. But, the rapid reduction of performance function in this direction does not imply the fastest convergence always. The search is done along the conjugate directions in the conjugate gradient algorithms thus, generally producing speedier convergence than the steepest-descent direction. To find the length of the updated weight step size, most of the algorithms use a learning rate. But, the step size is modified in each iteration in most of the conjugate gradient algorithms. Therefore, to reduce the performance function, the search is done along the conjugate gradient direction to find the step size.

A key advantage of the scaled conjugate gradient is that it does not line search at each iteration as compared to all other conjugate gradient algorithms. In the line search, the network responses of all training inputs are computed some times for every search which is computationally expensive. Thus, to avoid time-consuming line searches the scaled conjugate gradient algorithm (SCG) was designed by Moller, (1993). The Scaled Conjugate Gradient (SCG) algorithm is a supervised algorithm which is fully-automated. It does not include any critical user-dependent parameters, and it is much faster than the Levenberg-Marquardt Backpropagation. We can use this algorithm on any dataset, if net input, weight and transfer functions of the given dataset have a derivative function. Derivatives of performance concerning bias variable $\mathbf{X}$ and weight are calculated using backpropagation. So, it avoids line search at every iteration to approximate scale step size by using the LM algorithm (Hagan et al., 1996).

Training phase stops when any of the following conditions appear:

- If the maximum number of repetitions achieved.
- If maximum time is overshot.
- The performance reduced to the target.
- If the gradient of the performance is lower than the minimum gradient.
- If the validation performance has crossed the maximum fail times since the last time it decreased (when using validation).

Theory of Bayesian regularization

Bayesian regularized artificial neural networks (BRANNs) eliminate or reduce the requirement for lengthy cross-validation. Hence perform more robustly than standard backpropagation. Bayesian regularization is a mathematical process that converts a nonlinear regression into a “well-posed” statistical problem in the manner of ridge regression. A key advantage of this algorithm is that it considers the probabilistic nature of the weights in the network related to the given data set. The probability of overfitting increases dramatically as more hidden layer of neurons added in the neural network. Thus for a stopping point, it requires a validation set. In this algorithm all

unreasonable complex models penalized by pushing extra linkage weights to zero. The network will train and calculate the non-trivial weights. As the network grows some parameters will converge to a constant. Also, the volatility and noise in stock markets lead to the probability of overtraining for basic backpropagation networks. But, Bayesian networks are more parsimonious and tend to reduce the probability of overfitting and eliminate the need for a validation step. Therefore, the available data for training is increased (Jonathon, 2013).

Bayesian regularization has the same usage criteria as the Scale Conjugate Gradient Backpropagation algorithm. This algorithm minimizes the weights and a linear combination of squared errors. For good generalization qualities of the network, this algorithm modifies the linear combinations (Guresen et al., 2011; Hagan & Menhaj, 1999). This Bayesian regularization takes place within the Levenberg-Marquardt algorithm.

Results

Performance plots

The performance plots help us to identify the number of iterations (epochs) at which the mean squared error become least or stops changing. The number of iterations does not represent time as we can see that Scaled Conjugate Gradient gives the best validation in 103 (54) iterations and Levenberg-Marquardt gives in 10 (13) iterations on tick dataset (15-min dataset) but the time taken by Scaled Conjugate Gradient is less than Levenberg-Marquardt in both datasets. From Fig. 3. We see that Bayesian Regularization is giving least mean squared error compared to Levenberg-Marquardt, followed by Scaled Conjugate Gradient when overall performance over all datasets. But, when only the performance on test dataset is compared, the Scaled Conjugate Gradient gives the best performance.

For all three algorithms, the same dataset is used. The training is done on 60% of the dataset, 15% of the dataset is used for validations, and the rest 25% of the dataset is used for testing (Since 25% of dataset is used for testing, value of K in K-fold validation is 4). Since the Bayesian Regularized Artificial Neural Network uses both the training and validation parts for training, it uses a total of 75% of the dataset for training. The testing dataset is chosen at random from the dataset.

Regression plots

The network performance is validated through regression plots. Thus, the network output regarding targets for training, validation, testing, and overall datasets are displayed by the regression plots. The Bayesian Regularization uses the whole validation dataset for training as well. From Fig. 4. We can see that the fit is very good for all tick data sets as the R values in each case of 0.99 or above, but the accuracy drops when predicting over the 15-min dataset. Only Bayesian Regularization gives R-value of almost 0.99 on the 15-min dataset. The accuracy of SCG and LM drop to 0.97 and 0.96 respectively over the 15-min dataset. Here also Bayesian Regularization is outperforming LM and SCG incomplete dataset. But, when only the regression plots on test dataset are compared, the Scaled Conjugate Gradient gives best results. These plots portray that

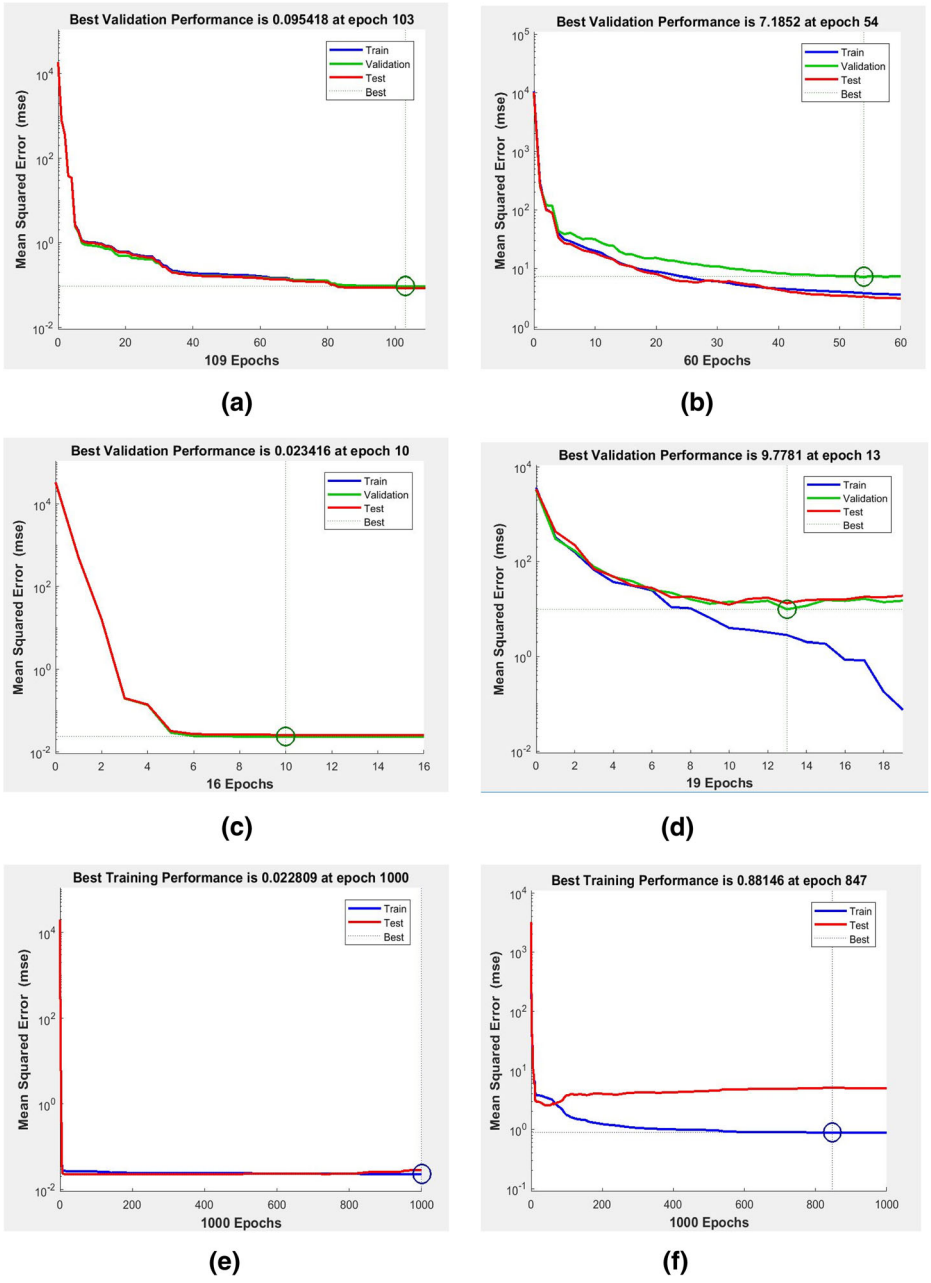

Fig. 3 Validation Performance plots. (**a**) SCG on tick data: (**b**) SCG on 15-min data: (**c**) Levenberg-Marquardt on tick data: (**d**) Levenberg-Marquardt on 15-min data: (**e**) Bayesian Regularization on tick data: (**f**) Bayesian Regularization on 15-min data

prediction over tick dataset gives better predictions than prediction over the 15-min dataset.

In Table 1 prediction accuracy using general validation and K-fold validation have been compared using mean square error (MSE) metric. We can see that there is not significant change in accuracy for all the algorithms. Similar comparison has been done in Table 2 using 15-min tick data. In Tables 3 and 4 general validation has been used in order to compare results using MSE and mean absolute percentage error (MAPE) metrics. In both the table prediction accuracy using MAPE and MSE indicators has

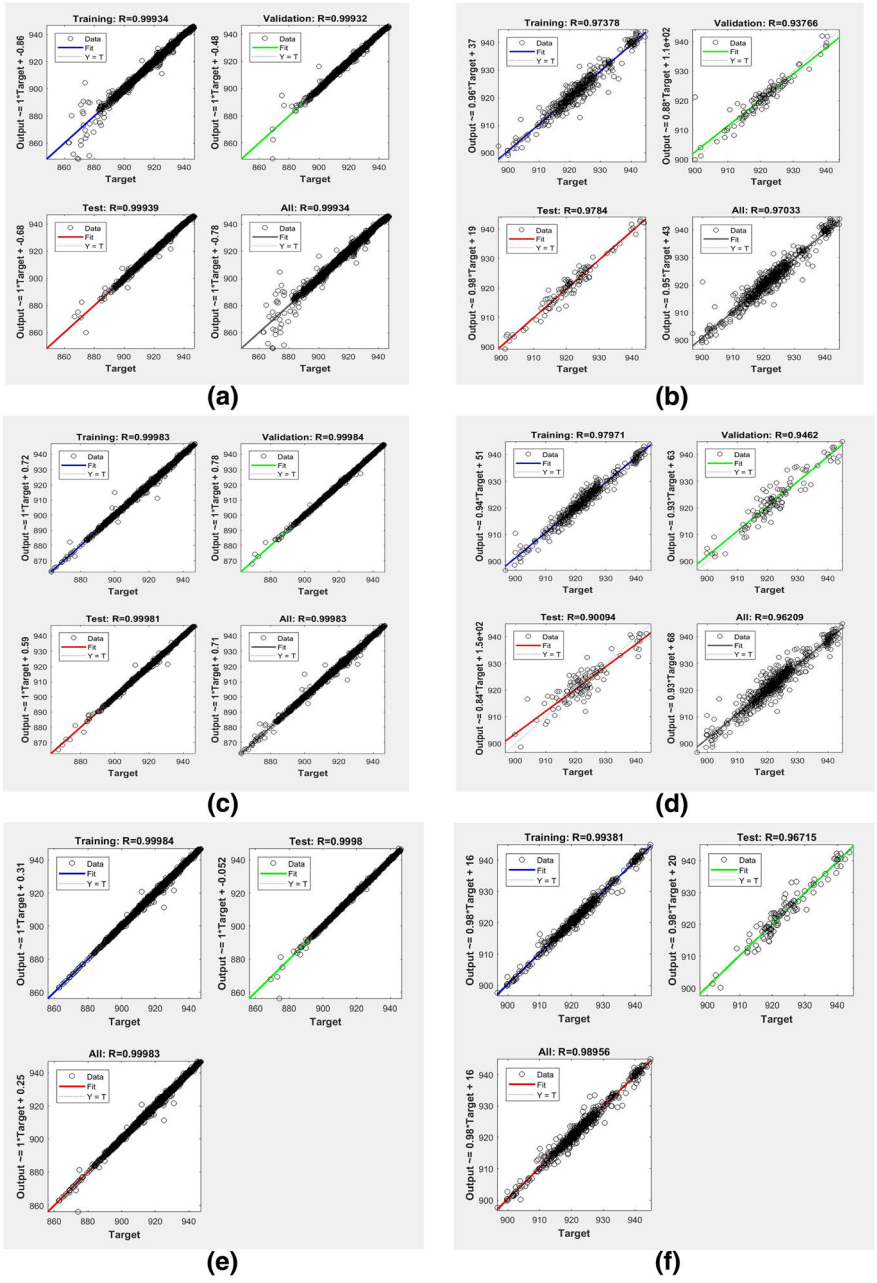

Fig. 4 Regression plots. (**a**) SCG on tick data: (**b**) SCG on 15-min data: (**c**) Levenberg-Marquardt on tick data: (**d**) Levenberg-Marquardt on 15-min data: (**e**) Bayesian Regularization on tick data: (**f**) Bayesian Regularization on 15-min data

Table 1 Accuracy (in %) comparison between K-fold and General validation for tick data

No.	Algorithms	K-fold validation				General validation			
		Train	Valid	Test	All	Train	Valid	Test	All
1	SCG	99.943	99.938	99.947	99.943	99.934	99.932	99.939	99.934
2	LM	99.983	99.974	99.83	99.982	99.983	99.984	99.981	99.983
3	BR	99.984	–	99.971	99.981	99.984	–	99.98	99.983

Table 2 Accuracy (in %) comparison between K-fold and General validation for 15-min tick data

No.	Algorithms	K-fold validation				General validation			
		Train	Valid	Test	All	Train	Valid	Test	All
1	SCG	97.478	97.116	95.5	96.934	97.378	93.766	97.84	97.033
2	LM	98.762	95.461	94.853	97.304	97.971	94.62	90.94	96.209
3	BR	99.51	–	93.422	98.245	99.381	–	96.715	98.956

been compared. From Tables 1, 2, 3 and 4, we can say that prediction using tick data gives us better accuracy rather than using 15-min tick data.

Error histograms

In Fig. 5, red bars represent the testing data, green bars represent validation data, and blue bars represent training data. The error range (maximum negative error to maximum positive error) is divided into 20 bins, and the plots are drawn. The outliers can be identified in the histogram. Thus, the data points where the fit is notably worse than the majority of data termed as outliers. In this case, we can see that all three methods give better results on tick data as compared to the 15-min data. The error on tick data is majorly in the smallest bin over null error whereas the error in 15-min data is distributed over a few bins. From the error histograms, it's visible that Bayesian Regularization outperforms both Scaled Conjugate Gradient and Levenberg-Marquardt regarding accuracy over both the datasets.

Conclusion and future work

This study compares the performance of three Neural Network learning algorithms, i.e., Levenberg-Marquardt, Scaled Conjugate Gradient and Bayesian Regularization by predicting over tick by tick dataset and 15-min dataset. The study shows that prediction using tick by tick data for the stock market gives much better results than prediction using 15-min dataset. The first algorithm is based on Levenberg-Marquardt optimization which uses an approximation to Hessian matrix to approach second-order training speed. This gives excellent results and takes a few hours to train. The second algorithm Scaled Conjugate Gradient (SCG), based on conjugate directions uses a step size scaling mechanism to avoid time-consuming line search per learning iteration, which makes this algorithm much faster than the second order algorithms like Levenberg-Marquardt. Training using SCG takes a few minutes which is a significant improvement over Levenberg-Marquardt, but the error also increases in the tick data prediction. The third algorithm Bayesian Regularization takes a few days to train over a large dataset but gives better results than both Levenberg-Marquardt and SCG. All three algorithms provide an accuracy of 99.9% using tick data. The accuracy over

Table 3 Accuracy (in %) comparison between MAPE and MSE for tick data

No.	Algorithms	MAPE				MSE			
		Train	Valid	Test	All	Train	Valid	Test	All
1	SCG	99.909	99.914	99.902	99.908	99.934	99.932	99.939	99.934
2	LM	99.984	99.972	99.981	99.981	99.983	99.984	99.981	99.983
3	BR	99.984	–	99.938	99.973	99.984	–	99.98	99.983

Table 4 Accuracy (in %) comparison between MAPE and MSE for 15-min tick data

No.	Algorithms	MAPE				MSE			
		Train	Valid	Test	All	Train	Valid	Test	All
1	SCG	97.127	95.926	96.919	96.907	99.934	99.932	99.939	99.934
2	LM	95.588	88.622	87.903	92.553	99.983	99.984	99.981	99.983
3	BR	99.769	–	92.908	98.147	99.984	–	99.98	99.983

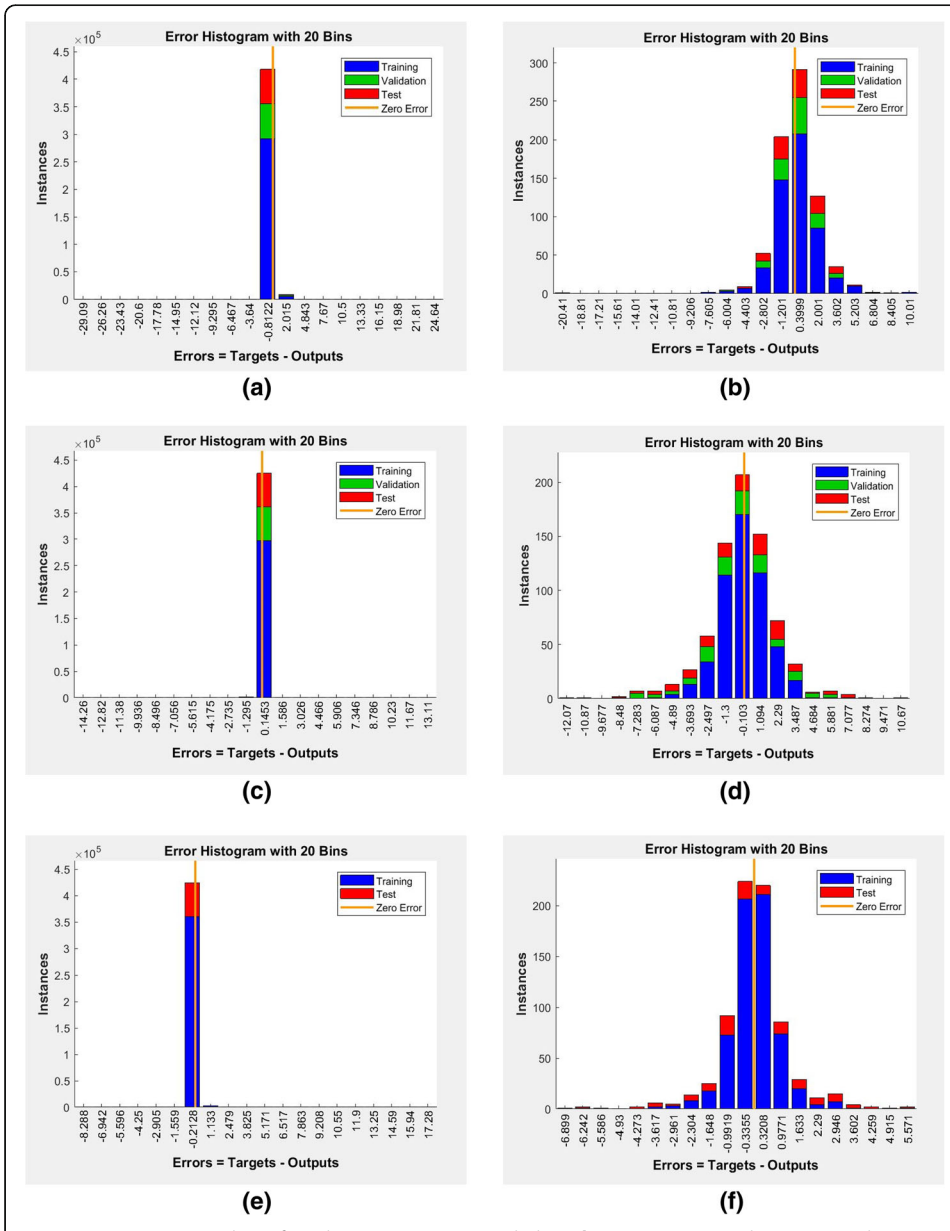

Fig. 5 Error Histogram plots of predictions. (**a**) SCG on tick data: (**b**) SCG on 15-min data: (**c**) Levenberg-Marquardt on tick data: (**d**) Levenberg-Marquardt on 15-min data: (**e**) Bayesian Regularization on tick data: (**f**) Bayesian Regularization on 15-min data

15-min test dataset changes completely. SCG takes least time and gives best results compared to Levenberg-Marquardt and Bayesian Regularization. But the resut obtained on 15 min dataset is significantly poor in comparison with that of results obtained using tick data.

In this case study, the data of past 30 business days used. A more extensive dataset can be used to bring in seasonal and annual factors that affect the stock price movement. Also predicting the minute by minute data can reduce dataset size by 70% and may be able to give comparable results while allowing us to use historical data of a more significant period. Recurrent Neural Networks may provide better predictions than the neural networks used in this study, e.g., LSTM (Long Short-Term Memory). Since statements and opinions of renowned personalities are known to affect stock prices, some Sentiment Analysis can help in getting an extra edge in stock price prediction.

Endnotes

[1]https://eikon.thomsonreuters.com/index.html

Abbreviations

ANN: Artificial Neural Network; ARIMA: Autoregressive Integrated Moving Average; BPNN: Back Propagation Neural Network; BR: Bayesian Regularization; BRANN: Bayesian Regularized Artificial Neural Network; JAN: January; LM: Levenberg Marquardt; LSTM: Long Short-Term Memory; MAPE: Mean Absolute Percentage Error; MLP: Multilayer Perceptron; MSE: Mean Square Error; NOV: November; QSAR: Quantitative Structure-Activity Relationship; SCG: Scaled Conjugate Gradient; SVM: Support Vector Machine; SVR: Support Vector Regression

Acknowledgments

The authors are thankful to the Department of Mathematics, IIT Delhi for providing us the resources to perform this case study.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Availability of data and materials

The data used is included with the submission of manuscript.

Authors' contributions

We have no conflicts of interest to disclose. All the authors contributed equally to this work. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Received: 9 April 2018 Accepted: 11 March 2019
Published online: 21 March 2019

References

Ahangar RG, Yahyazadehfar M, Pournaghshband H (2010) The comparison of methods artificial neural network with linear regression using specific variables for prediction stock Price in Tehran stock exchange. Int J Comp Sci Informat Sec 7(2):38–46

Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, Oxford, UK

Cigizoglu HK (2003) Estimation, forecasting and extrapolation of river flows by artificial neural networks. Hydrol Sci J 48(3):349–361

Cortes C, Vapnik V (1995) Support vector networks. Mach Learn 20:273–297

Fausett L (1994) Fundamentals of neural networks. Prentice Hall, New York, NY, USA

Guresen E, Kayakutlu G, Daim TU (2011) Using artificial neural network models in stock market index prediction. Expert Syst Appl 38:10389–10397

Hagan MT, Demuth HB, Beale MH (1996) Neural Network Design. PWS Publishing, Boston, MA

Hagan MT, Menhaj M (1999) Training feedforward networks with the Marquardt algorithm. IEEE Trans Neural Netw 5(6):989–993

Hagan MT, Menhaj MB (1994) Training feedforward networks with the Marquardt algorithm. IEEE Trans Neural Netw 5(6):989–993

Jonathon TL (2013) A Bayesian regularized artificial neural network for stock market forecasting. Experts System with Application 40:5501–5506
Kuan CM, Liu T (1995) Forecasting exchange rates using feedforward and recurrent neural networks. J Appl Econ 10(4):347–364
Mehdi K, Zahra H (2017) Performance evaluation of series and parallel strategies for financial time series forecasting. Financial Innovation 3:24
Moller MF (1993) A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw 6:525–533
Qi M, Zhang GP (2008) Trend time series modeling and forecasting with neural networks. IEEE Trans Neural Netw 19(5):808–816
Samek D, Varachha P (2013) Time series prediction using artificial neural networks. International Journal of Mathematical Models and Methods in Applied Sciences 7(1):30–46
Shadbolt J, Taylor JG (eds) (2002) Neural networks and the financial markets: predicting, combining and portfolio optimization. Springer-Verlag, London
Trippi RR, Turban E (eds) (1992) Neural networks in finance and investing: using artificial intelligence to improve real world performance. McGraw-Hill, New York
Walczak S (2001) An empirical analysis of data requirements for financial forecasting with neural networks. J Manage Inf Syst 17(4):203–222
Y Kara, M, A Boyacioglu, O K, Baykan (2011) Predicting direction of stock Price index movement using artificial neural networks and support vector machines: the sample of the Istanbul stock exchange. Expert Syst Appl 38:5311–5319
Yodele et al (2012) Stock Price prediction using neural network with hybridized market indicators. Journal of Emerging Trends in Computing and Information Sciences 3(1):1–9