Bayesian Neural Network for Stock Market Forecasting

Bennett Breese
Dept. of Aerospace Engineering
University of Cincinnati
Cincinnati, USA
breesebj@mail.uc.edu

Kaushik Palani
Dept. of Mechanical Engineering
University of Cincinnati
Cincinnati, USA
palanikk@mail.uc.edu

***Abstract*— Economists and investors have always sought to predict future stock prices using predictive indicators such as exponential moving averages (EMA), relative strength indices (RSI), stochastic K%, etc. Recent developments in AI and neural networks, however, have greatly contributed to the accuracy of economic forecasting. Neural networks allow the model to adapt to market changes, however they generally suffer from overtraining and overfitting. The objective of this project is to prevent overfitting and generalize the network. To do this, the sum square error of the network weights will be added to the cost function, and Bayesian techniques will be used to determine the contributions of the weights and error in the cost function. Various predictive indicators will be used for the inputs of the network. The output will be the predicted next day closing price. The network will be trained and verified on historical stock data.**

Keywords—Neural Network, Bayesian Regulation, Economics

I. INTRODUCTION

The presented work attempts to generalize a stock forecasting neural network through the use of Bayesian regulation. This work relies heavily on the process presented by Ticknor (2013) [1] and uses Bayesian regulation techniques of Forsee and Hagan (1997) [2].

The accuracy of the NN presented in [1] is of great interest because of recent unanticipated market volatility resulting from recessions and U.S. economic stimulus packages. This paper seeks to test the robustness of the forecasting NN when faced with highly uncertain systems such as the stock market.

The work closely follows the process in [1] in an attempt to duplicate its results to learn the basic methodology.

II. PROCESS

A. Neural Network Structue

The network consists of five layers: and one input layer, one hidden layer, and one output layer.

The NN has nine total inputs, each of which are commonly used predictive indicators. These indicators were calculated for each trading day. The table below lists the NN's inputs and their respective formulas.

Network Inputs	Formula
Low Price	L_t
High Price	H_t
Opening Price	O_t
5-day Exponential Moving Average	$EMA_t = C_t\left(\frac{\alpha}{1+5}\right) + EMA_{t-1}\left(1-\left(\frac{\alpha}{1+5}\right)\right)$
10-day Exponential Moving Average	$EMA_t = C_t\left(\frac{\alpha}{1+10}\right) + EMA_{t-1}\left(1-\left(\frac{\alpha}{1+10}\right)\right)$
Relative Strength Index (RSI)	$RSI_{14day} = 100 - \left\lfloor \frac{100}{1+\frac{g/14}{l/14}} \right\rfloor$
Williams %R	$W_t = \frac{H_{t-14:t} - C_t}{H_{t-14:t} - L_{t-14:t}} * 100\%$
Stochastic Oscillator (Fast, K)	$\%K_t = \frac{C_t - L_{t-14:t}}{H_{t-14:t} - L_{t-14:t}} * 100\%$
Stochastic Oscillator (Slow, D)	$\%D_t = \frac{\sum_{i=0}^{2} \%K_{t-i}}{3}$

Table 1:
C_t is the closing price at time t.
α is a smoothing factor ($\alpha = 2$).
g and l are the averages of the gains and losses over 14 days.

The hidden layer consists of only five neurons. Lastly, the NN has one output: the next day's closing price.

It should be noted that all the inputs and outputs are normalized between -1 and 1 to assist in NN training. The output then must be rescaled to its original units to obtain a meaningful prediction.

B. Bayesian Regulation and Cost Function

Neural networks have been used in the past for market forecasting. A typical NN uses the sum of squared errors (E_D) as the cost function (the function we seek to minimize). However, these previous networks suffered from overfitting and overtraining. In essence, the network memorized the training data, instead of learning.

Large networks, containing many hidden layer neurons, are prone to overfitting. Bayesian regulation prevents overfitting by driving unnecessary weights to zero. To accomplish this, an additional term is introduced into the cost function, which is the sum square of the network weights (E_w). The equation is as follows:

$$F = \beta E_D + \alpha E_w$$

Where α and β are the cost function parameters. The optimal cost function parameters must be found. This process is clearly outlined in [2]. First, the probability density function (PDF) for the weights is obtained using Bayes rule.

$$P(\boldsymbol{w}|D,\alpha,\beta,M)=\frac{P(D|\boldsymbol{w},\beta,M)P(\boldsymbol{w}|\alpha,M)}{P(D|\alpha,\beta,M)}$$

D is the data set, M is the NN model, and $\boldsymbol{w}$ is the vector of network weights. If the noise the in the training data and prior distribution for the weights is assumed to be Gaussian, then the following is true:

$$P(\boldsymbol{w}|D,\alpha,\beta,M)=\frac{\frac{1}{Z_w(\alpha)Z_D(\beta)}e^{-F(\boldsymbol{w})}}{Normalization}=\frac{1}{Z_F(\alpha,\beta)}e^{-F(\boldsymbol{w})}$$

Where $Z_w(\alpha)=\left(\frac{\pi}{\alpha}\right)^{N/2}$ and $Z_D(\beta)=\left(\frac{\pi}{\beta}\right)^{n/2}$; with N being the number of network weights, and n being the number of data points. Maximizing the above PDF is the same as minimizing the cost function $F(\boldsymbol{w})$. The optimal cost function parameters must still be obtained. Using Bayes rule again, a PDF for the optimization parameters is obtained.

$$P(\alpha,\beta|D,M)=\frac{P(D|\alpha,\beta,M)P(\alpha,\beta|M)}{P(D|M)}$$

If $P(\alpha,\beta|M)$ is assumed to be uniform PDF, then a MAP estimate can be obtained from by maximizing $P(D|\alpha,\beta,M)$, which is the normalization factor for $P(\boldsymbol{w}|D,\alpha,\beta,M)$. Thus…

$$P(D|\alpha,\beta,M)=\frac{P(D|\boldsymbol{w},\beta,M)P(\boldsymbol{w}|\alpha,M)}{P(\boldsymbol{w}|D,\alpha,\beta,M)}=\frac{Z_F(\alpha,\beta)}{Z_D(\beta)Z_W(\alpha)}$$

$Z_F(\alpha,\beta)$ is then estimated by a Taylor series expansion, and because the cost function $F(\boldsymbol{w})$ is quadratic in a small area around the minimum point $\boldsymbol{w}^{MP}$, the following approximation can be made.

$$Z_F(\alpha,\beta)=(2\pi)^{\mathrm{N}/2}\left(\det((\boldsymbol{H}^{MP})^{-1})\right)^{1/2}e^{-F(\boldsymbol{w}^{MP})}$$

$$\boldsymbol{H}=\beta\nabla^2E_D+\alpha\nabla^2E_w$$

$\boldsymbol{H}$ is the Hessian matrix of the cost function. The cost function parameters can then be obtained.

$$\alpha^{MP}=\frac{\gamma}{2E_w(\boldsymbol{w}^{MP})}$$

$$\beta^{MP}=\frac{n-\gamma}{2E_D(\boldsymbol{w}^{MP})}$$

Where γ is the effective number of parameters:

$$\gamma=N-2\alpha^{MP}tr(\boldsymbol{H}^{MP})^{-1}$$

All of this is outlined by Forsee and Hagan in [2]. Please refer to it for further reference.

The Levenburg-Marquardt algorithm is used to optimize the cost function, primarily because it readily provides a Gauss-Newton approximation to the Hessian matrix ($\boldsymbol{H}=\nabla^2F(\boldsymbol{w})\approx 2\beta\boldsymbol{J}^T\boldsymbol{J}+2\alpha\boldsymbol{I}_N$). One iteration of the algorithm attempts to minimizes the cost function for an initial guess of α and β. The Hessian matrix is then used to compute the new values of α and β. The process is repeated until convergence.

III. Model Setup

With a preliminary optimization run to determine the optimal network size, the network is setup with a single layer of five hidden neurons. This value is chosen after an experiment of varying the number of neurons until the effective number of parameters reached a constant value. A tangent sigmoid function is used for the hidden layer which ensured the output in the range [-1,1].

$$\tanh(\mathrm{n})=\frac{2}{1+exp(-2n)}-1$$

IV. Simulation Experiments

The section of historical stock data was used to train and test the neural network. 80% of the selected data was used for training, and 20% was used for validation. The data included stocks of Microsoft, Amazon, and Google for the experiment. All of the available data was normalized and the parameters were stored as settings for later rescaling of the output values.

The performance of the network is measured by computing the mean absolute percentage error MAPE

$$\mathrm{MAPE}=\frac{\sum_{i=1}^{r}\left(\frac{abs(y_i-p_i)}{y_i}\right)}{r}\times 100\%$$

The network is trained for no more than 1000 epochs around where it converges.

V. Results

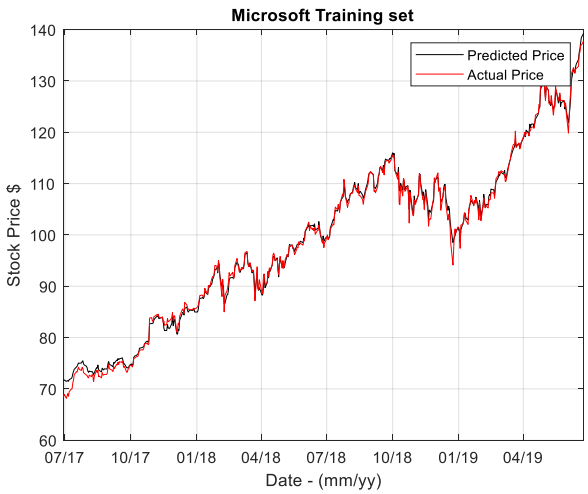

Fig 1b: One day future stock prediction for Microsoft test set

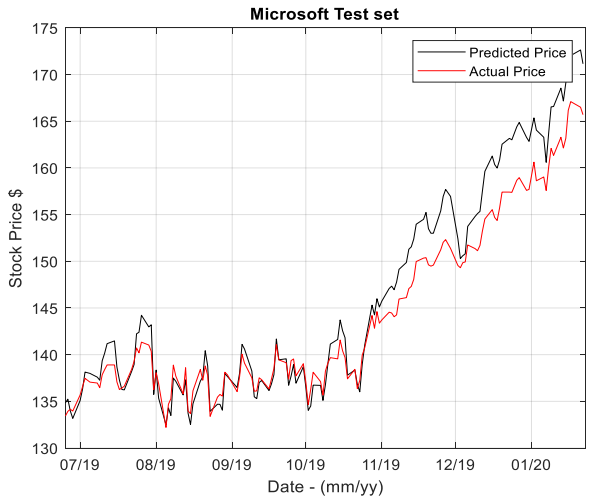

Fig 1b: One day future stock prediction for Microsoft training set

Fig 2a: One day future stock prediction for Amazon training set

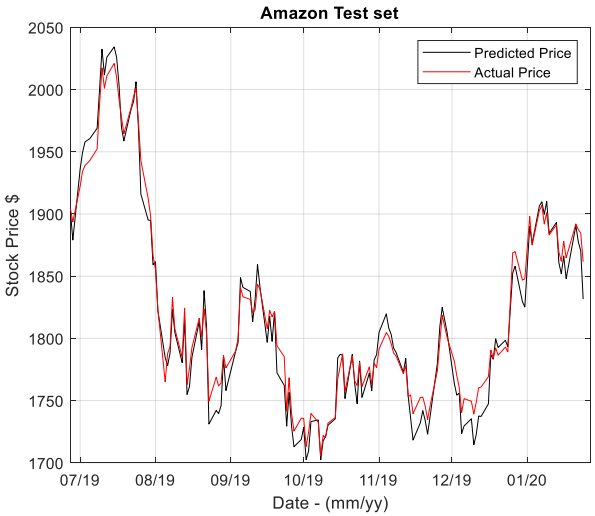

Fig 2b: One day future stock prediction for Amazon test set

Fig 3a: One day future stock prediction for Google training set

Fig 3b: One day future stock prediction for Google test set

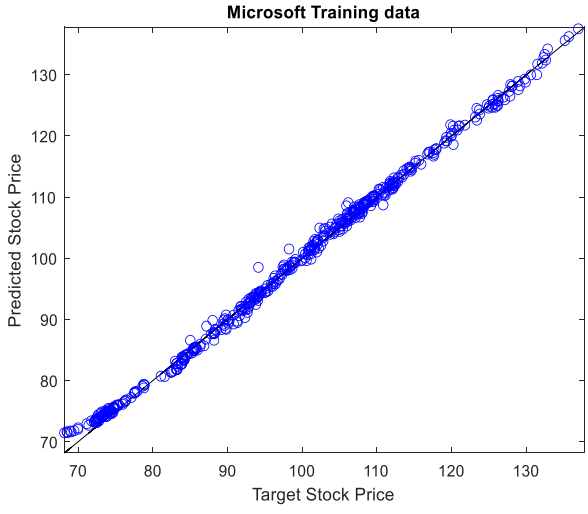

Fig 4a: Comparison of target and predicted one day future price – Microsoft training set

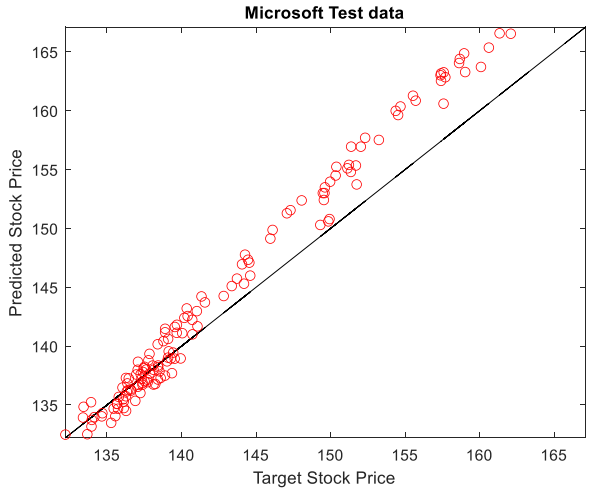

Fig 4b: Comparison of target and predicted one day future price – Microsoft test set

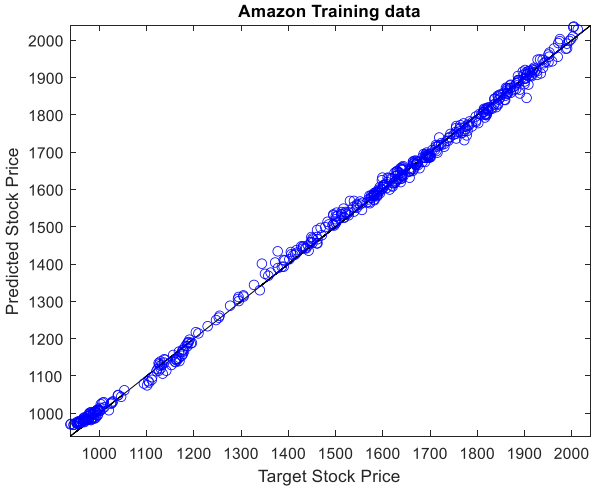

Fig 5a: Comparison of target and predicted one day future price – Amazon training set

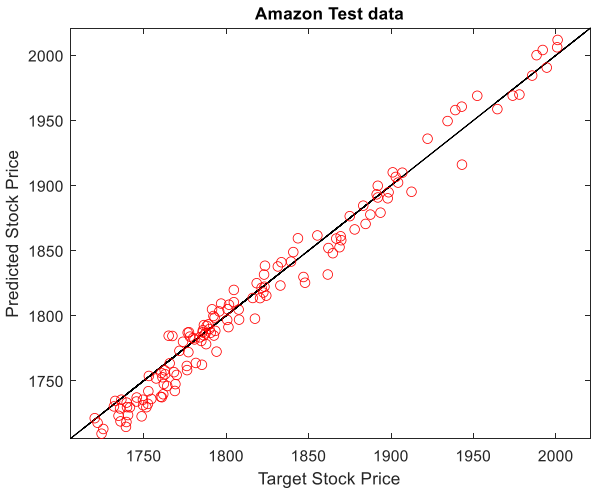

Fig 5b: Comparison of target and predicted one day future price – Amazon test set

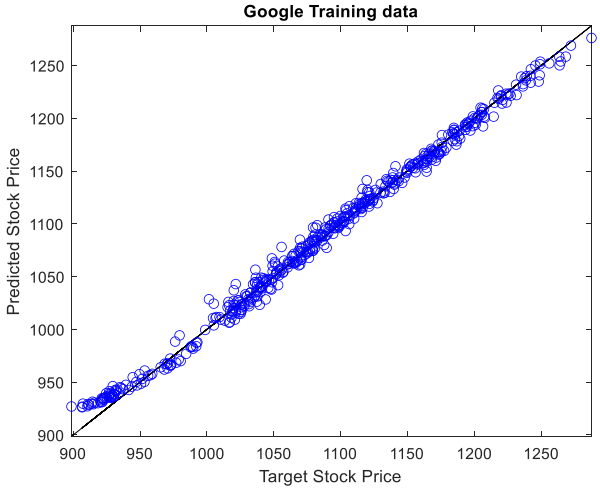

Fig 6a: Comparison of target and predicted one day future price – Google training set

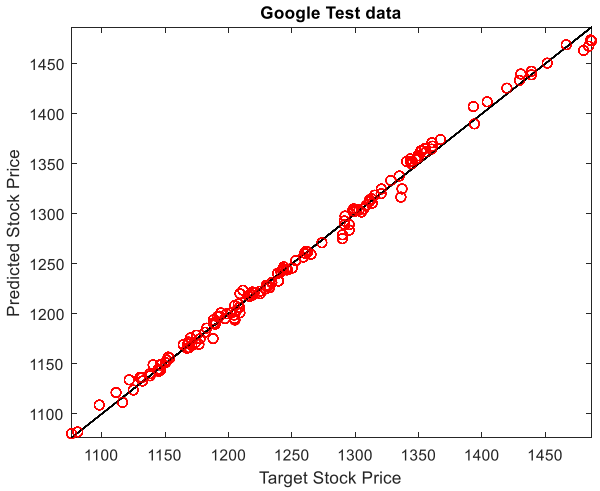

Fig 6b: Comparison of target and predicted one day future price – Google test set

STOCK NAME	MAPE TRAINING SET	MAPE TEST SET
MICROSOFT	1.4152%	1.8221%
AMAZON	1.6000%	1.0724%
GOOGLE	1.3085%	1.0041%

Table 2: Mean absolute percentage error MAPE for the training and test sets over different stocks

Conclusion

The Bayesian regularized neural network model eliminates the dependency on the validation set to prevent overfitting and overtraining of the network. Bayesian regularization ensures that the network learns the model instead of “memorizing” it, thus improving the accuracy of any new inputs it receives. In theory this should provide better results for atypical inputs, such as those experienced during economic declines.

References

[1] Ticknor, Jonathan L. "A Bayesian regularized artificial neural network for stock market forecasting." *Expert Systems with Applications* 40.14 (2013): 5501-5506.

[2] Foresee, F. Dan, and Martin T. Hagan. "Gauss-Newton approximation to Bayesian learning." *Proceedings of International Conference on Neural Networks (ICNN'97)*. Vol. 3. IEEE, 1997.