Multimed Tools Appl (2017) 76:18569–18584
DOI 10.1007/s11042-016-4159-7

Stock prediction using deep learning

Ritika Singh[1] · **Shashi Srivastava**[2]

Received: 16 July 2016 / Revised: 24 October 2016 / Accepted: 14 November 2016 /
Published online: 17 December 2016

Abstract Stock market is considered chaotic, complex, volatile and dynamic. Undoubtedly, its prediction is one of the most challenging tasks in time series forecasting. Moreover existing Artificial Neural Network (ANN) approaches fail to provide encouraging results. Meanwhile advances in machine learning have presented favourable results for speech recognition, image classification and language processing. Methods applied in digital signal processing can be applied to stock data as both are time series. Similarly, learning outcome of this paper can be applied to speech time series data. Deep learning for stock prediction has been introduced in this paper and its performance is evaluated on Google stock price multimedia data (chart) from NASDAQ. The objective of this paper is to demonstrate that deep learning can improve stock market forecasting accuracy. For this, $(2D)^2$PCA + Deep Neural Network (DNN) method is compared with state of the art method 2-Directional 2-Dimensional Principal Component Analysis $(2D)^2$PCA + Radial Basis Function Neural Network (RBFNN). It is found that the proposed method is performing better than the existing method RBFNN with an improved accuracy of 4.8% for Hit Rate with a window size of 20. Also the results of the proposed model are compared with the Recurrent Neural Network (RNN) and it is found that the accuracy for Hit Rate is improved by 15.6%. The correlation coefficient between the actual and predicted return for DNN is 17.1% more than RBFNN and it is 43.4% better than RNN.

Keywords Deep Learning · Stock Prediction · Neural Network · $(2D)^2$PCA · Radial Basis Function Neural Network · Regularization · Multimedia

✉ Ritika Singh
ritikasingh.ism@gmail.com

Shashi Srivastava
shashi_cheshta@rediffmail.com

[1] Indian School of Mines, Dhanbad, India

[2] Faculty of Management Studies, Banaras Hindu University, Varanasi, India

1 Introduction

Accurate stock market multimedia (chart) prediction is considered impossible by the old school of thought. The Efficient Market hypothesis states that stock prices reflect all current information and new information leads to unpredictable stock prices. Random walk concluded that stock prices cannot be accurately predicted using historical values [17].

However, the attraction of good returns has led to myriad methods for price prediction. In the last three decades abundant research has been done in this area. But still researchers are of the view that prediction of stocks on non-linear non-stationary financial time series is one of the most challenging tasks. Several mathematical models have been developed but the results are still dissatisfying [21]. Studies focusing on forecasting the stock markets have been mostly preoccupied with forecasting volatilities [6].

Professional Traders use fundamental and technical analysis for price prediction. There is plethora of literature which suggests different methods for stock price and indices prediction. Fundamental approach is the traditional approach using company parameters [19]. Technical analysis is based on Dow Theory [19] and uses price history for prediction. It ranges from traditional statistical modelling to methods based on artificial intelligence and machine learning [28]. In literature many ANN models are evaluated against the statistical models for stock prediction. ANN has also been compared with different data mining classification algorithms [9, 28] and the comparison suggests that ANN models give better results [6]. Literature suggests that the first neural network in stock market prediction was given by White [29]. Classical ANNs were mostly used in stock in the later part of last century. However, Ten years ago researchers focused on applying Multi layer perceptron (MLP) for stock prediction [20, 24]. Of late, different variants of ANN in hybrid models have been applied to stock market prediction. Atsalakis et al. [1] have surveyed ANNs applied for stock prediction but they have not pointed out the feature extraction methods used for stock prediction. The literature includes Genetic Algorithm to optimise a RNN for stock forecasting [13]. Artificial Fish Swarm Algorithm (AFSA) was used to optimise RBFNN for stock prediction [23]. Extreme learning machine has also been used to construct Decision Support System for stock prices prediction and trading strategies [25]. Moreover, different feature extraction methods were used with ANN such as Curvilinear Component Analysis along with RBFNN was used for Bel20 Stock Market Index [16]. Fusion technique was used in Indian context to model stock data [22]. Very recently, combination of feature extraction using 2-Directional 2-Dimensional Principal Component Analysis ($(2D)^2PCA$) with RBFNN has been applied for stock price prediction [4].

Some of the studies, however, have shown that ANN has few drawbacks and it is not suitable for stock prediction because stock market data has enormous noise and complex dimensionality. ANN exhibits inconsistent and unpredictable behaviour on this data [23]. Most of the neural networks have shallow architecture and are thus designed with one hidden layer. One of the reasons may be unsuccessful training strategy for multi-layer. Another problem with neural networks is that many of them tend to fall into a local optimum solution thus over-fitting [14]. However, deep architectures can overcome these problems [14] and they have already yielded promising performance in many fields including language [30], speech [18] and image [8, 10, 34]. Hinton first proposed the

idea of deep learning and harnessed the power of model beyond three-level nets [7]. According to some recent papers DNNs can give better approximation to nonlinear functions than shallow models [15, 26]. DNNs have been applied to some time series forecasting and they have shown good results [11, 12].

Recent literature suggests that researchers are attempting to use deep learning for stock prediction. Successful application in speech domain [18] has led to the idea that since speech is a time series data and stock data is also time-series so this method can be used. However, the DNN techniques are not fully harnessed. A paper which uses autoencoders to extract features from the input variables has been proposed for stock trading strategies [27].

In this paper DNN has been introduced as a classifier to stock multimedia (chart) trend prediction and when compared with state of the art technologies [23] applied to stock this perform better. Section 2 describes the research methodology suggested for stock prediction. Section 3 provides the framework for the proposed model. It depicts the working of the model through block diagram. Along with this different features of the deep learning which are used in the model have been described. Section 4 implements the framework and describes the experimental setup. Section 5 is a results and analysis section which shows that DNN performs better than some of the latest techniques applied in this domain. Finally, Section 6 carries the conclusion and the future direction.

2 Research methodology

The proposed methodology uses $(2D)^2PCA$ for dimensionality reduction [33] and thereafter it uses DNN as a predictor. For $(2D)^2PCA$, N samples in I consists of $\{I_{11}, I_{12}, I_{1n}, I_{m1}, I_{m2}, I_{mn}\}$ where $N = m*n$ and the covariance can be defined as

$$Cov(I) = \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(I_{ij}-\overline{I}\right)*\left(I_{ij}-\overline{I}\right)^{T} \tag{1}$$

Where, $\overline{I} = \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}$ is the mean of all samples. Here single value decomposition (SVD) is used to compute projecting subspace V_d for first d largest Eigen value. The feature matrix Y for (2D)PCA is obtained by (2)

$$Y_i = I^T * V_d \tag{2}$$

Next, Y^T isused as the new training sample in place of I and the process is repeated to compute the output of $(2D)^2PCA$ as Z_i. The output of $(2D)^2PCA$ is fed to the DNN.

DNN is a multi-layer feed forward neural network and it uses supervised learning as shown in Fig. 1. Here X_i are nodes in the input layer and Y_j represent neurons in the 1^{st} hidden layer and it uses hyperbolic tangent function for computation. Z_k represent neurons in the 2^{nd} layer and it again uses hyperbolic tangent function for computation. Finally the output layer has two nodes P_l which uses the softmax function for classification and linear function for regression. The hyperbolic tangent represents the activation function for the network. U_{ij} are the weights connecting the input and 1^{st} hidden layer and b_j are the biases for 1^{st} hidden layer. V_{jk} are the weights connecting the 1^{st} hidden layer and the 2^{nd} hidden layer and c_k are the biases for 2^{nd} hidden layer.

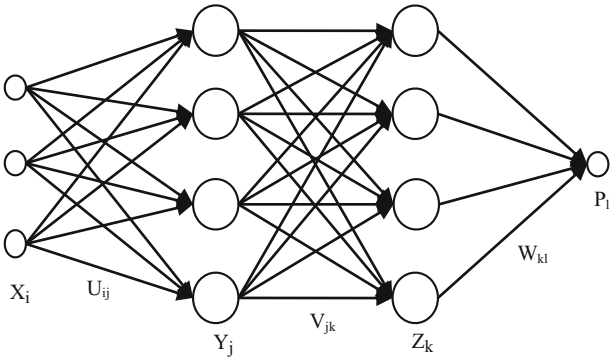

Fig. 1 Forward Propagation of a 4 layered DNN

Finally W_{kl} are the weights connecting the 2^{nd} hidden layer and the output layer and d_l are the biases for output layer.

$$Y_j = f\left(X_i, U_{ij}, b_j\right) = tanh\left\{\left(\sum_{i=1}^{3} X_i{*}U_{ij}\right) + b_j\right\} = \frac{e^{\left\{\left(\sum_{i=1}^{3} X_i{*}U_{ij}\right)+b_j\right\}} - e^{-\left\{\left(\sum_{i=1}^{3} X_i{*}U_{ij}\right)+b_j\right\}}}{e^{\left\{\left(\sum_{i=1}^{3} X_i{*}U_{ij}\right)+b_j\right\}} + e^{-\left\{\left(\sum_{i=1}^{3} X_i{*}U_{ij}\right)+b_j\right\}}} \tag{3}$$

$$Z_k = f_1\left(Y_j, V_{jk}, c_k\right) = tanh\left\{\left(\sum_{j=1}^{4} Y_j{*}V_{jk}\right) + c_k\right\} = \frac{e^{\left\{\left(\sum_{j=1}^{4} Y_j{*}V_{jk}\right)+c_k\right\}} - e^{-\left\{\left(\sum_{j=1}^{4} Y_j{*}V_{jk}\right)+c_k\right\}}}{e^{\left\{\left(\sum_{j=1}^{4} Y_j{*}V_{jk}\right)+c_k\right\}} + e^{-\left\{\left(\sum_{j=1}^{4} Y_j{*}V_{jk}\right)+c_k\right\}}} \tag{4}$$

$$P_l = f_2(Z_k, W_{kl}, d_l) = softmax\left\{\left(\sum_{k=1}^{4} Z_k{*}W_{kl}\right) + d_l\right\} = \frac{e^{\left\{\left(\sum_{k=1}^{4} Z_k{*}W_{kl}\right)+d_l\right\}}}{\sum_{k=1}^{4} e^{\left\{\left(\sum_{k=1}^{4} Z_k{*}W_{kl}\right)+d_l\right\}}} \tag{5}$$

Learning occurs when these weights are adapted to minimize the error on labelled training data. The loss error function which is the objective function is minimized for the model

depending on whether the model terminates in a linear regression or classification. W is the collection $\{w_i\}_{1:N-1}$, where w_i denotes the weight matrix connecting layers i and $i+1$ for a network of N layers. B is the collection $\{b_i\}_{1:N-1}$, where b_i denotes the column vector of biases for layer $i+1$.

The model given in Fig. 1 is a regression problem. For regression the loss function is given below:

$$Mean\ \ Squared\ \ Error = L(W,\ \ B|j) = \frac{1}{2}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2 \tag{6}$$

Here, y_j is the actual output and $\hat{y}_j$ is the predicted output where j denotes number of training examples. The loss function for classification is given below:

$$Cross\ \ Entropy = L(W,B|j) = -\sum_{j=1}^{n} ln\left(\hat{y}_j\right) * y_j + ln\left(1-\hat{y}_j\right) * \left(1-y_j\right) \tag{7}$$

In order to update weights and biases of the network a supervised training algorithm Stochastic Gradient Descent (SGD) is used. Following process is iterated till the convergence criteria are reached. First, W and B are initialized and then updated according to the following equations.

$$w_{jm} = w_{jm} - \alpha * \frac{\partial L(W,B|j)}{\partial w_{jm}} \tag{8}$$

$$b_{jm} = b_{jm} - \alpha * \frac{\partial L(W,B|j)}{\partial b_{jm}} \tag{9}$$

Here, α is the learning rate and w_{jm} is the weight for m^{th} neuron connecting layer j and $j+1$. Similarly, b_{jm} is the bias for m^{th} neuron connecting layer j and $j+1$ whereas $\partial L(W,B|\ \frac{j)}{\partial w_{jm}}$ is computed using backward propagation. The chain rule is used to compute this function and for the last output layer the computation is shown below:

$$\frac{\partial L(W,B|j)}{\partial w_{jm}} = \frac{\partial L(W,B|j)}{\partial f_2(Z_k, w_{kl}, d_l)} * \frac{\partial f(Z_k, w_{kl}, d_l)^2}{\partial\left(\sum_{k=1}^{4} Z_k * w_{kl} + d_l\right)} * \frac{\partial\left(\sum_{k=1}^{4} Z_k * w_{kl} + d_l\right)}{\partial w_{jm}} \tag{10}$$

3 Proposed method

In this paper, we have two purposes: one is to introduce DNN as a proposed model in stock prediction and the other is to demonstrate that this method gives improved result compared to state of the art method. Figure 2 represents the proposed model.

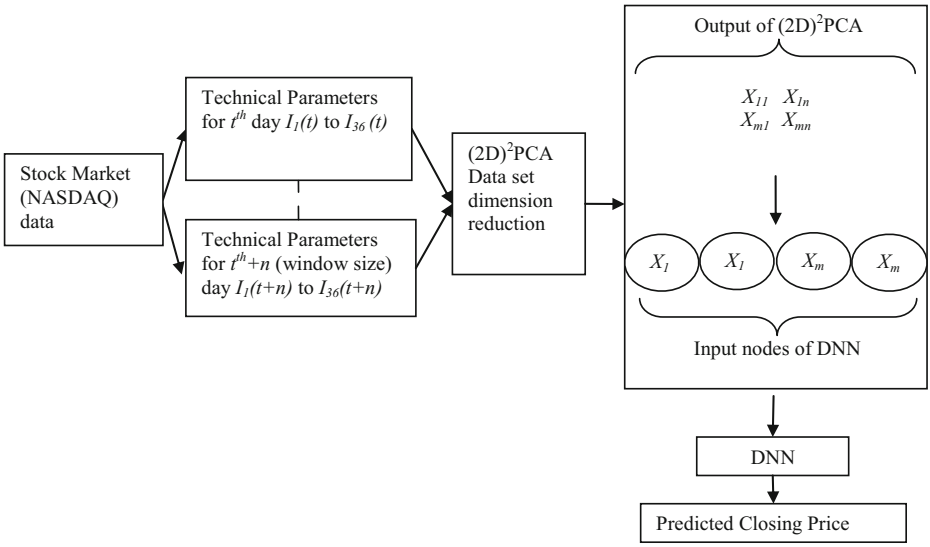

Fig. 2 Framework of the proposed model

3.1 Data Collection

The data is collected from NASDAQ and the prediction is done for individual stock. This is because the index data does not consider firms characteristics and company wise prediction is more useful for the investors [20]. Therefore the data is collected for Google stock multimedia (chart) which is an American multinational technology company specializing in Internet-related services and products. Our goal is to consider a time period long enough to capture a high diversity in price movements and also to avoid data snooping. The data set used for experiment is from August 19, 2004 to December 10, 2015. Hence, the model is built for working 2843 days. Further the data set is divided in training set and testing set. Training set consists of data from August 19 2004 to May 31 2011 and testing set from June 1 2011 to December 10 2015.

In the problem, each record of data set includes daily information which consists of the closing price, the highest price, the lowest price, and the opening price named at day t as $x(t)$, $x_h(t)$, $x_l(t)$ and $x_o(t)$ respectively. Other technical analysis parameters used as input include the leading, lagging and trend change indicators to get a composite result.

This paper uses 36 variables for forecasting as used in literature [23]. The variables I_1 to I_{36} are computed based on the equations in Table 1. However, two parameters have been replaced with Bollinger Bands which compare the volatility and the relative price levels [28]. These variables when used as input to the model provide the forecast for the closing price on the next day. The forecast is for short-term because data far from the forecasting date provides less and less information useful to forecasting value [16].

Table 1 Input variables for the stock market data set

Name of the Variable	Description and Formula
$I_1 = x_o(t)$	Open Price
$I_2 = x_h(t)$	High Price
$I_3 = x_l(t)$	Low Price
$I_4 = x(t)$	Close Price
$I_5 = MA5$, $I_6 = MA10$, $I_7 = MA20$	Moving Average
$I_8 = BIAS5$, $I_9 = BIAS10$	BIAS
$I_{10} = DIFF$	*EMA12-EMA26*
I_{11} = ,*BU*	$(x(t)\text{-}bollinger_{upper})/bollinger_{upper}$
$I_{12} = BL$	$(x(t)\text{-}bollinger_{lower})/bollinger_{lower}$
$I_{13} = K$, $I_{14} = D$	Stochastic Fast %K ,Fast %D
$I_{15} = ROC$	Price rate of change
$I_{16} = TR$	True range of price movements
$I_{17} = MTM6$, $I_{18} = MTM12$	Momentum
$I_{19} = WR\%10$, $I_{20} = WR\%5$	Williams index
$I_{21} = OSC6$, $I_{22} = OSC12$	Oscillator
$I_{23} = RSI6$, $I_{24} = RSI12$	Relative strength index
$I_{25} = PSY$	Psychological line
I_{26}	$K(t)\text{-}K(t\text{-}1)$
I_{27}	$D(t)\text{-}D(t\text{-}1)$
I_{28}	$(x(t)\text{-}x(t\text{-}1))/x(t\text{-}1)$
I_{29}	$(x(t)\text{-}x_o(t))/x_o(t)$
I_{30}	$(x(t)\text{-}x_l(t))/(x_h(t)\text{-}\ x_l(t))$
I_{31}	$(MA5(t)\text{-}MA5(t\text{-}1))/MA5(t\text{-}1)$
I_{32}	$(MA20(t)\text{-}MA20(t\text{-}1))/MA20(t\text{-}1)$
I_{33}	$(MA5(t)\text{-}MA20(t\text{-}1))/MA20(t\text{-}1)$
I_{34}	$(x(t)\text{-}MA20(t))/MA20(t)$
I_{35}	$(x(t)\text{-}min(x(t\text{-}1),x(t\text{-}2),\ldots,x(t\text{-}N)))/\ min(x(t),x(t\text{-}1),x(t\text{-}2),\ldots,x(t\text{-}N))$
I_{36}	$(x(t)\text{-}max(x(t\text{-}1),x(t\text{-}2),\ldots,x(t\text{-}N)))/\ max(x(t),x(t\text{-}1),x(t\text{-}2),\ldots,x(t\text{-}N))$

3.2 Dimension reduction

$(2D)^2PCA$ is used to reduce the dimensions of the data set [33] as it projects the original raw data matrix into a projection matrix. There is loss of information due to this method but the processing time and the convergence speed of the model increases many fold. On a large data set such loss of information would not cause much variation in the output. The output of $(2D)^2PCA$ is fed to the DNN input nodes as shown in Fig. 2.

3.3 Forecasting

The forecasting is done in two phases where in the first phase the training is done to compute weights W and biases B of the model. In the second phase testing is done where W and B are

used to compute the output. Before this the output of the $(2D)^2PCA$ is normalized [23] according to the (11) to bring it in a range [0,1] .

$$Z_{ij} = \frac{Z_{ij} - min(Z_i)}{max(Z_i) - min(Z_i)} \tag{11}$$

Here in (11), Z_i denotes the output of $(2D)^2PCA$ and Z_{ij} is the normalized output which is used as the input to the DNN

3.4 Regularization

DNN is a complicated network and uses large number of parameters and as the complexity of the model increases the bias decreases but variance increases. In order to balance the bias-variance trade-off regularization is used and it makesthe model simpler. Further it reduces the variance by limiting the biases and making few of them 0 thus reducing the generalization error (error rate observed on validation data) and avoiding model overfitting [3, 5, 31].

The model uses $\ell 1$ and $\ell 2$ norms for regularization as it modifies the loss function mentioned below:

$$L'(W, B|j) = L(W, B|j) + \lambda_1 R_1(W, B|j) + \lambda_2 R_2(W, B|j) \tag{12}$$

For $\ell 1$ regularization $R_1(W,B\ \mathrm{I}\ j)$ is sum of all absolute weights and biases. For $\ell 2$ regularization $R_2(W,B\ \mathrm{I}\ j)$ is sum of squares of all weights and biases. The constants λ_1 and λ_2 are chosen to be very small, for example 10^{-5}.

3.5 Advanced optimization

Adaptive learning rate algorithm ADADELTA [32] automatically combines the benefits of learning rate annealing and momentum training to avoid slow convergence. It predicts the stock prices very fast and gives more accurate result. Learning rate annealing is a heuristics approach and the drawback of this method is that it tends to slow down at local minima and move fast whenever suitable and moreover the learning rate is applied to all dimensions of the parameter [32].

Momentum is a per-dimension training method and it is an improvement over SGD. The gradients along the minima are much smaller but since they are in the same direction they keep accumulating, hence speeding up the training.

$$w_t + 1 = w_t + \Delta w_t \tag{13}$$

$$b_t + 1 = b_t + \Delta b_t \tag{14}$$

$$\Delta w_t = (\rho \Delta w_t - 1) - \eta \frac{\partial L(W, B|t)}{\partial w_t} \tag{15}$$

$$\Delta b_t = (\rho \Delta b_t - 1) - \eta \frac{\partial L(W, B|t)}{\partial b_t} \tag{16}$$

Here, ρ is a constant which is controlling the decay of previous parameter updates and η is global learning rate shared by all dimensions. ADAGRAD uses an update rule for bias b and for weight w which is given in (17)

$$\Delta w_t = -\frac{\eta}{\sqrt{\sum_{i=1}^{t} \left(\frac{\partial L(W, B|i)}{\partial w_i}\right)^2}} * \left(\frac{\partial L(W, B|t)}{\partial w_t}\right) \tag{17}$$

The ADAGRAD method is sensitive to the choice of learning rate η and since the denominator is continual accumulation of squared gradient η will continue to decay. As suggested by Zeiler [3] ADADELTA is used to overcome these limitations. ADADELTA accumulates the gradient for a certain window size and the denominator uses local estimate for recent gradients where at time t the running average is given by (18)

$$E\left[\left(\frac{\partial L(W, B|t)}{\partial w_t}\right)^2\right] = \rho E\left[\left(\frac{\partial L(W, B|t-1)}{\partial w_{t-1}}\right)^2\right] + (1-\rho) * \left(\frac{\partial L(W, B|t)}{\partial w_t}\right)^2 \tag{18}$$

$$\Delta w_t = -\frac{\eta}{\sqrt{E\left[\left(\frac{\partial L(W, B|t)}{\partial w_t}\right)^2\right] + \varepsilon}} * \frac{\partial L(W, B|t)}{\partial w_t} \tag{19}$$

In order to match the units of the numerator and the denominator an added term is put in (19) which becomes

$$\Delta w_t = -\frac{\sqrt{E[\Delta w_{t-1}{}^2] + \varepsilon}}{\sqrt{E\left[\left(\frac{\partial L(W, B|t)}{\partial w_t}\right)^2\right] + \varepsilon}} * \frac{\partial L(W, B|t)}{\partial w_t} \tag{20}$$

$$E\left[\Delta w_t{}^2\right] = \rho E\left[\Delta w_{t-1}{}^2\right] + (1-\rho) * \left(\Delta w_t{}^2\right) \tag{21}$$

Here ADADELTA is an improvement over these two methods as it avoids the selection of hyperparameter. Since it is difficult to estimate learning rates for a DNN with deep architecture therefore ADADELTA for DNN gives better result.

4 Experimental setup

The purpose of this experiment is to predict the stock closing price and to compare the performance of this model with other ANN models. The latest literature shows that $(2D)^2$PCA along with RBFNN has performed the best among all the other dimensionality reduction techniques combined with ANN [23]. Therefore for this experiment the

dimensionality has been reduced using $(2D)^2PCA$ for RNN, RBFNN and DNN model. The reason for doing this is to bring uniformity across the models. The resultant dimensions of $(2D)^2PCA$ have been chosen to be 10×10,15×15 and 19×35 for a window size of 20 and therefore 36×20 matrix is reduced to the above mentioned dimensions. Window size is the number of days for which the data is being taken into account for predicting the next day's data. Say, window size 20 means data is being taken for 20 days and results are predicted for 21st day.

Once the dimensionality is reduced the output is computed for each day based on Deep learning equations. Both the input data and output data for the training set are passed to the deep learning method. The regularization parameters $\ell 1$ and $\ell 2$ are set to 10^{-5}. It is found that this is the best $\ell 1$ and $\ell 2$ for the range of $\ell 1$ and $\ell 2 = 10^n (n = -5,-6\ldots,-10)$. The loss function is set to the mean square error and the regression stopping criteria is set to 0. For ADADELTA from three possible values for $\rho = 0.9$, 0.99 and 0.999 the best one is selected. The best ε value is selected from the range 10^n ($n = -4, -5, -6\ldots,-10$) and the numbers of epochs are set to 1000.

For the RBFNN model, Mean Squared Error goal is set to 0 for the range $m*10^n (m = 1,2,\ldots,9;\ n = 1,2\ldots,9)$ is found to be and SPREAD is selected through repeated experiments according to performance considerations. The best SPREAD $6*10^3$. The maximum number of neurons is set equal to the total number of dimensions i.e. for 10×10 it is 100. For Elman's RNN [2] the learning rate is 0.1 and the number of units in the hidden layers is 10 where as the maximum number of iterations to learn is 1000. The performance was measured using different error parameters such as Root Mean Square Error (RMSE), Hit Rate (HR) and Total Return (TR) etc. are listed in Table 2.

The experiment is conducted using a PC with 4GB RAM, 2 GHz PCU on an R package of version 3.2.2. However since the state of the art technique RBFNN is implemented on MATLAB 7.1(R2010a) platform [23] therefore the same platform is used for RBFNN.

5 Results and analysis

The experimental results for different window sizes and dimensions of the model are shown in Fig. 3 and Tables 3, 4, 5, and 6. Table 7 compares the performance of the proposed DNN with RBFNN and RNN. The results are drawn for varying window sizes 20,40,60,80,100 to test which window size gives the better result. Along with this $(2D)^2PCA$ is used to reduce the dimensions of input matrix 36×20 to lowest range, middle range and last range.

In Fig. 3 the x-axis denotes the normalized closing price, y-axis denotes the number of days. For window size 20 both lowest range dimensions 10×10 and the middle range 15×15 give better performance as actual and predicted lines are quite close. However, for the last range dimensions 19×35 the results are not so satisfactory. For window size 40 the lowest range dimensions 10×10 give better performance. However, for the middle range dimensions 25×25 and last range 39×35 the results are not so satisfactory. For all the other window sizes the lowest range dimensions 10×10 provides the best result. Our assumption made based on literature [16] that the short term data provides better forecast is hence correct. The lowest range data performs better than middle range.

The errors are measured for each of the window sizes and the reduced dimensions according to the equations given in Table 2. It is found that amongst the lowest dimension matrix i.e. 10×10 the best performance is for the window size 20 which is shown in Table 3. Similarly it is found that amongst the middle range dimension matrix the best performance is for the window size 20 as shown in Table 4. Further it is found that amongst the last range

Table 2 Formula for different error measures

Name	Description	Formula
r_1	Correlation coefficient between actual value and prediction value	$r1 = \frac{\sum_{t=1}^{N} (y(t)-\bar{y}(t))*(y(t)-\bar{y}(t))}{\sqrt{\sum_{t=1}^{N} (y(t)-\bar{y}(t))^2*(y(t)-\bar{y}(t))^2}}$
R^2	Non-linear regression multiple correlation coefficient	$R^2 = 1-\frac{\sum_{t=1}^{N} (y(t+1)-y(t))^2}{\sum_{t=1}^{N} (y(t)-\bar{y}(t))^2}$
r_2	Correlation coefficient between actual return and prediction return	$r2 = \frac{\sum_{t=1}^{N} (r_e(t)-\overline{r_e}(t))*(r_e(t)-\overline{r_e}(t))}{\sqrt{\sum_{t=1}^{N} (r_e(t)-\overline{r_e}(t))^2*(r_e(t)-\overline{r_e}(t))^2}}$
PCD	Percentage of correct direction	$PCD = \frac{1}{N}\sum_{t=1}^{N} Pcd_t$ $Pcd_t = \begin{cases} 1 & (y(t+1)\text{-}y(t))*(y(t+1)-y(t)) > 0 \\ 0 & \text{else} \end{cases}$
SMAPE	Symmetric mean absolute percentage error	$SMAPE = \frac{1}{N}\sum_{t=1}^{N} 2\lvert y(t)-y(t)\rvert / \lvert y(t)+y(t)\rvert$
MAPE	Mean Absolute Percentage Error	$MAPE = \frac{1}{N}\sum_{t=1}^{N} \frac{\lvert y(t)-y(t)\rvert}{y(t)}$
RMSE	Root mean square error	$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N} \frac{(y(t)-y(t))^2}{y(t)}}$
HR	Hit Rate	$HR = \frac{\sum_{t=1}^{N} P_t}{N}$ $Pt = \begin{cases} 1 & (y(t+1)-y(t))*(y(t+1)-y(t)) > 0 \\ 0 & \text{else} \end{cases}$
TR	Total Return	$TR = \sum_{t=1}^{N} [y(t+1)-y(t)], if\, y(t+1) > y(t)$

Guo et al. (2015) [4]

dimension matrix the best performance is for the window size 20 as shown in Table 5. Finally, the results reflect confidence in our assumption taken from literature [16] that short term forecasting in case of stock prediction is more accurate.

Since window size 20 performs the best amongst all the other window sizes therefore the results are compared for its reduced dimensions as presented in Table 6. It is also found that 10×10 dimension matrix provides the best result and the total return for 10×10 is 1.36 and for 19×35 is 0.41 which is more than 200%. Additionally the forecasting accuracy measured by Hit Rate is 0.68 for 10×10 and 0.65 for 19×35 which is 4.4% better and for the remaining error measures the 10×10 dimension again gives a better result.

Since window size 20 and dimensions 10×10 provide the best result therefore this is used as input to DNN, RBFNN and RNN as displayed in Fig. 4.

It is found that RNN is performing very poorly. However, RBFNN and DNN predicted values are very close to the actual value. For perspective, it is observed from the results of the DNN model that for window size 20 and dimensions 10×10 the model architecture uses a 4

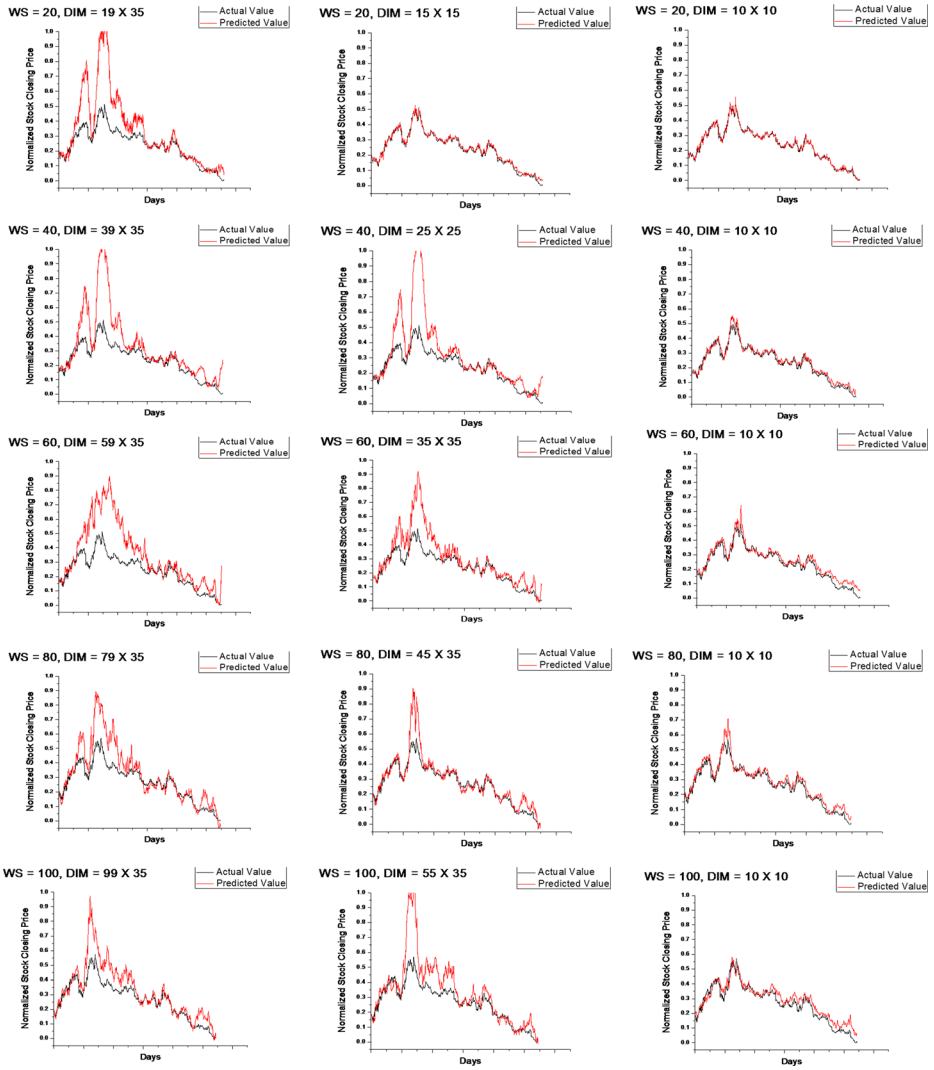

Fig. 3 Actual and Predicted closing prices for different window sizes and dimensions

Table 3 Errors measured for lowest range dimensions

	r1	R2	r2	PCD	SMAPE	MAPE	RMSE	HR	TR
WS = 20 DIM = 10 × 10	0.999806	0.987063	0.75994	0.697321	0.054142	0.081409	0.010119	0.682143	1.364618
WS = 40 DIM = 10 × 10	0.999224	0.959242	0.458956	0.61241	0.12025	0.582232	0.021312	0.59982	0.283717
WS = 60 DIM = 10 × 10	0.999074	0.911757	0.66899	0.594203	0.179519	0.905168	0.031988	0.655797	0.133455
WS = 80 DIM = 10 × 10	0.999524	0.92291	0.773434	0.582117	0.152405	0.612506	0.033329	0.677007	0.206159
WS = 100 DIM = 10 × 10	0.996044	0.888398	0.133287	0.568015	0.204279	0.960425	0.040996	0.522059	−0.08833

Table 4 Errors measured for middle range dimensions

	r1	R2	r2	PCD	SMAPE	MAPE	RMSE	HR	TR
WS = 20 DIM = 15 × 15	0.999803	0.982213	0.74345	0.665179	0.088049	0.499792	0.012535	0.672314	1.235738
WS = 40 DIM = 25 × 25	0.983043	−1.08365	0.729083	0.56205	0.275887	2.467331	0.161563	0.655576	0.355019
WS = 60 DIM = 35 × 35	0.988322	−0.00643	0.658834	0.554348	0.303593	0.80187	0.112149	0.672123	0.240923
WS = 80 DIM = 45 × 35	0.998148	0.700918	0.665416	0.615876	0.375986	0.699731	0.067551	0.653236	0.393695
WS = 100 DIM = 55 × 35	0.99765	−0.07002	0.512048	0.556066	0.256571	0.598832	0.129662	0.627757	0.039194

Table 5 Errors measured for last range dimensions

	r1	R2	r2	PCD	SMAPE	MAPE	RMSE	HR	TR
WS = 20 DIM = 19 × 35	0.988542	0.46765	0.640625	0.573214	0.288314	1.248716	0.175043	0.651786	0.405198
WS = 40 DIM = 39 × 35	0.974678	−1.34204	0.626882	0.556655	0.31122	2.845007	0.178395	0.651178	0.237793
WS = 60 DIM = 59 × 35	0.975919	−1.31822	0.621122	0.548913	0.363049	1.652996	0.179103	0.642377	0.00297
WS = 80 DIM = 79 × 35	0.978721	0.203557	0.633657	0.539234	0.381063	2.850367	0.181391	0.637044	0.131631
WS = 100 DIM = 99 × 35	0.972354	0.42476	0.627856	0.566397	0.350216	1.456187	0.184837	0.63751	0.12525

Table 6 Errors measured for window size 20

	r1	R2	r2	PCD	SMAPE	MAPE	RMSE	HR	TR
DIM = 10 × 10	0.999806	0.987063	0.75994	0.697321	0.054142	0.081409	0.010119	0.682143	1.364618
DIM = 15 × 15	0.999803	0.982213	0.74345	0.665179	0.088049	0.499792	0.012535	0.672314	1.235738
DIM = 19 × 35	0.988542	0.46765	0.640625	0.573214	0.288314	1.248716	0.175043	0.651786	0.405198

Table 7 Errors measured for various neural networks and DNN

	r1	R2	r2	PCD	SMAPE	MAPE	RMSE	HR	TR
DNN	0.999806	0.987063	0.75994	0.697321	0.054142	0.081409	0.010119	0.682143	1.364618
RBFNN	0.999945	0.991196	0.634118	0.744643	0.0696	0.080059	0.00674	0.649107	1.703763
RNN	0.985493	0.541488	0.427822	0.5375	0.34913	2.227725	0.075669	0.575893	0.05273

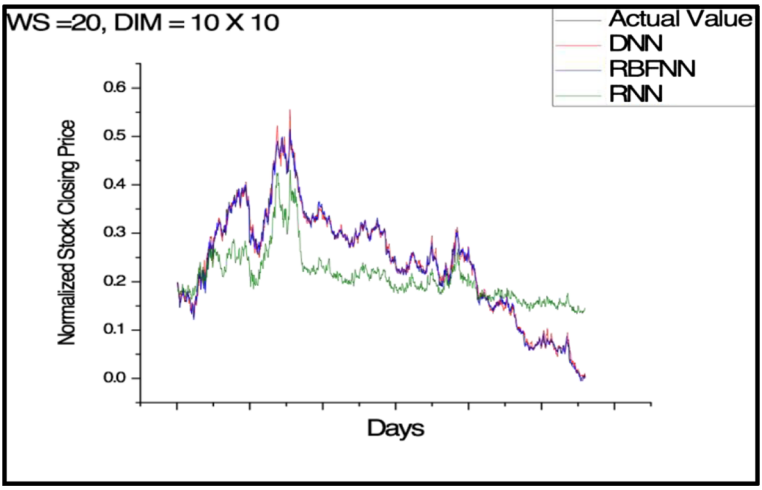

Fig. 4 Comparison of actual stock price and forecasted values from DNN, RBFNN and RNN

layered network with 100 input units, 200 hidden layer units and 1 output unit. The model Mean Square error is 6.43e-05 and the training time is 3 min and 51 s.

The comparative error measures for these neural networks are shown in Table 7. It is found that RNN is not a good performer, but when DNN and RBFNN are compared it is found that they are very close. On a closer look Hit Rate performance is better for DNN as it is 4.8% more accurate than RBFNN and 15.6% better than the RNN. Therefore DNN can be a better predictor for trend prediction of stock market. The correlation coefficient between the actual and predicted return is 0.76 for DNN, it is 0.63 for RBFNN and 0.43 for RNN. This demonstrates that DNN is 17.1% more highly correlated than RBFNN and it is 43.4% better than RNN.

6 Conclusion

This is the first work using deep learning for stock data forecasting. In this paper it is demonstrated that $(2D)^2$PCA + Deep learning on the Google dataset can improve the accuracy of stock multimedia (chart) prediction compared to conventional neural network methods along with $(2D)^2$PCA. Also for varying window sizes and dimensions the model has been tested to improve the accuracy. It is found that for the window size 20 and dimension 10×10 the results are the best. The deep learning method for higher dimensions and large window sizes is giving limited performance.

Experimental results confirm that the proposed model provides a promising method for stock trend prediction as Hit Rate performance is better for DNN as it is 4.8% more accurate than RBFNN and 15.6% better than the RNN. Therefore DNN can be a better predictor for trend of stock market. The correlation coefficient between the actual and predicted return for DNN is 17.1% more accurate than RBFNN and it is 43.4% better than RNN. It is also found that the proposed model is not giving better results for Total Return and RMSE when compared to RBFNN. However, in future these parameters could be improved with other algorithms for Deep learning such as Deep Belief Network, Regularization, Autoencoders and Advanced Optimizations. Finally, it would be interesting to investigate the effectiveness of deep leaning in portfolio management and trading strategies.

References

1. Atsalakis GS, Valavanis KP (2009) Surveying stock market forecasting techniques–Part II: soft computing methods. Expert Syst Appl 36:5932–5941
2. Elman JL (1990) Finding structure in time. Cogn Sci 14:179–211
3. Erhan D, Bengio Y, Courville A, Manzagol PA, Vincent P, Bengio S (2010) Why does unsupervised pre-training help deep learning? J Mach Learn Res 11(Feb):625–660
4. Guo Z, Wang H, Yang J, Miller DJ (2015) A stock market forecasting model combining two-directional two-dimensional principal component analysis and radial basis function neural network. PLoS One 10, e0122385
5. Gupta P (2015) Deep Learning - Regularisation. https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiGiNiWyuDJAhXDkI4KHZQbBEcQFggbMAA&url=https%3A%2F%2Fcs.nyu.edu%2Fmishra%2FCOURSES%2F15.Summer%2FL4DNN.pdf&usg=AFQjCNEXq89m3B5prfmdk4s2eThK9YKA&sig2=z-wO9QRxYdD2ZS204B95ig&bvm=bv.110151844,d.c2E. Accessed 5 July 2016
6. Guresen E, Kayakutlu G, Daim TU (2011) Using artificial neural network models in stock market index prediction. Expert Syst Appl 38:10389–10397
7. Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. Neural Comput 18:1527–1554
8. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(80):504–507
9. Huang CJ, Yang DX, Chuang YT (2008) Application of wrapper approach and composite classifier to the stock trend prediction. Expert Syst Appl 34:2870–2878. doi:10.1016/j.eswa.2007.05.035
10. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Adv. Neural Inf. Process. Syst. 1097–1105
11. Kuremoto T, Kimura S, Kobayashi K, Obayashi M (2014) Time series forecasting using a deep belief network with restricted Boltzmann machines. Neurocomputing 137:47–56
12. Kuremoto T, Obayashi M, Kobayashi K, et al (2014) Forecast chaotic time series data by DBNs. In: Image Signal Process. (CISP), 2014 7th. Int Congr 1130–1135
13. Kwon YK, Moon BR (2007) A hybrid neurogenetic approach for stock forecasting. IEEE Trans Neural Netw 18:851–864. doi:10.1109/TNN.2007.891629
14. Larochelle H, Bengio Y, Louradour J, Lamblin P (2009) Exploring strategies for training deep neural networks. J Mach Learn Res 10:1–40
15. Le Roux N, Bengio Y (2010) Deep belief networks are compact universal approximators. Neural Comput 22:2192–2207
16. Lendasse A, de Bodt E, Wertz V, Verleysen M (2000) Non-linear financial time series forecasting-Application to the Bel 20 stock market index. Eur J Econ Soc Syst 14:81–91
17. Malkiel BG (2007) A random walk down Wall Street: the time-tested strategy for successful investing. WW Norton & Company
18. Mohamed A, Sainath TN, Dahl G, et al (2011) Deep belief networks using discriminative features for phone recognition. In: 2011 I.E. Int. Conf. Acoust. speech signal Process. 5060–5063
19. Murphy JJ (1999) Technical analysis of the financial markets: A comprehensive guide to trading methods and applications. Penguin
20. Pan H, Tilakaratne C, Yearwood J (2003) Predicting the Australian stock market index using neural networks exploiting dynamical swings and intermarket influences. In: Australas. Jt. Conf. Artif. Intell. 327–338
21. Patel J, Shah S, Thakkar P, Kotecha K (2015) Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert Syst Appl 42:259–268
22. Patel J, Shah S, Thakkar P, Kotecha K (2015) Predicting stock market index using fusion of machine learning techniques. Expert Syst Appl 42:2162–2172
23. Shen W, Guo X, Wu C, Wu D (2011) Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. Knowledge-Based Syst 24:378–385. doi:10.1016/j.knosys.2010.11.001
24. Situngkir H, Surya Y (2004) Neural network revisited: perception on modified Poincare map of financial time-series data. Phys A Stat Mech Appl 344:100–103
25. Sun F, Toh K-A, Romay MG, Mao K (2014) Extreme Learning Machines 2013: Algorithms and Applications. Springer
26. Sutskever I, Hinton GE (2008) Deep, narrow sigmoid belief networks are universal approximators. Neural Comput 20:2629–2636
27. Takeuchi L, Lee Y-YA (2013) Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks http://cs229.stanford.edu/proj2013/TakeuchiLeeApplyingDeepLearningToEnhanceMomentumTradingStrategiesInStocks.pdf

28. Teixeira LA, De Oliveira ALI (2010) A method for automatic stock trading combining technical analysis and nearest neighbor classification. Expert Syst Appl 37:6885–6890
29. White H (1988) Economic prediction using neural networks: the case of IBM daily stock returns. In: IEEE Int Conf. Neural Networks. 451–458
30. Yu D, Deng L, Wang S (2009) Learning in the deep-structured conditional random fields. In: Proc. NIPS Work. 1–8
31. Yu, K., Xu, W. and Gong, Y. (2009) Deep learning with kernel regularization for visual recognition. In Advances in Neural Information Processing Systems. 1889–1896
32. Zeiler MD (2012) ADADELTA: an adaptive learning rate method. arXiv Prepr. arXiv1212.5701
33. Zhang D, Zhou Z (2005) (2D)2PCA : 2-Directional 2-Dimensional PCA for Efficient Face Representation and Recognition. Neurocomputing 69:224–231. doi:10.1016/j.neucom.2005.06.004
34. Zuo Z, Wang G (2014) Learning discriminative hierarchical features for object recognition. IEEE Sig Proc Lett 21:1159–1163

Ritika Singh was born in Varanasi, India in 1984. She completed her B.Tech. in 2006 from JIIT, Noida, India and the MBA from IMI, Delhi, India, in 2010. She completed her Ph.D. in Industrial Engineering and Management from ISM, Dhanbad, India, in 2015. She has worked at Infosys Ltd. as a software engineer from 2006 to 2008 in India. Her research interest includes Big Data Analytics and Information Systems.

Shashi Shrivastava has done her MBA from FMS, BHU in 1993 and obtained Ph.D. degree in the area of Financial Management. She has also qualified NET (UGC) in 1994. She is presently Assistant Professor in FMS, BHU, Varanasi, India. Her area of interests includes Financial Management, Organizational Behavior and Strategic Management.