Applied Soft Computing 58 (2017) 35–52

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Forecasting financial time series volatility using Particle Swarm Optimization trained Quantile Regression Neural Network

Dadabada Pradeepkumar[a,b], Vadlamani Ravi[a,*]

[a] Center of Excellence in Analytics, Institute for Development and Research in Banking Technology (IDRBT), Castle Hills, Masab Tank, Hyderabad 500 057, India

[b] School of Computer and Information Sciences (SCIS), University of Hyderabad, Hyderabad 500 046, India

ARTICLE INFO

Article history:
Received 3 May 2016
Received in revised form 3 March 2017
Accepted 6 April 2017
Available online 24 April 2017

Keywords:
Financial time series volatility forecasting
GARCH
Quantile regression
QRNN
PSO

ABSTRACT

Accurate forecasting of volatility from financial time series is paramount in financial decision making. This paper presents a novel, Particle Swarm Optimization (PSO)-trained Quantile Regression Neural Network namely PSOQRNN, to forecast volatility from financial time series. We compared the effectiveness of PSOQRNN with that of the traditional volatility forecasting models, i.e., Generalized Autoregressive Conditional Heteroskedasticity (GARCH) and three Artificial Neural Networks (ANNs) including Multi-Layer Perceptron (MLP), General Regression Neural Network (GRNN), Group Method of Data Handling (GMDH), Random Forest (RF) and two Quantile Regression (QR)-based hybrids including Quantile Regression Neural Network (QRNN) and Quantile Regression Random Forest (QRRF). The results indicate that the proposed PSOQRNN outperformed these models in terms of Mean Squared Error (MSE), on a majority of the eight financial time series including exchange rates of USD versus JPY, GBP, EUR and INR, Gold Price, Crude Oil Price, Standard and Poor 500 (S&P 500) Stock Index and NSE India Stock Index considered here. It was corroborated by the Diebold–Mariano test of statistical significance. It also performed well in terms of other important measures such as Directional Change Statistic (Dstat) and Theil's Inequality Coefficient. The superior performance of PSOQRNN can be attributed to the role played by PSO in obtaining the better solutions. Therefore, we conclude that the proposed PSOQRNN can be used as a viable alternative in forecasting volatility.

1. Introduction

Volatility refers to the adverse movement of a financial variable. It is latent and difficult to estimate. The accurate forecasting of financial time series volatility is crucial to (1) the investors and the portfolio managers for assessing investment risk based on volatility of asset prices, (2) the traders for pricing of derivative securities based on the volatility of the underlying asset, (3) the financial risk managers for reserving the capital of at least three times that of Value-at-Risk (VaR) based on the given volatility forecast and (4) the policy makers as a barometer to measure the vulnerability of financial markets and the economy [1].

In finance, volatility is the spread of all likely outcomes of an uncertain variable (e.g., spread of asset returns). Statistically, volatility [1] is often referred to the standard deviation, σ, or the variance, σ^2, of returns of the financial variable of interest over some period. In general, a forecast of volatility ($\hat{\sigma}$) that involves considering historical data (e.g., daily returns) for a given period (e.g., N days), can be computed from a set of observations using Eq. (1) is as follows:

$$\hat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(r_t - \bar{r})} \quad (1)$$

where r_t is the return of financial variable on day t, $\bar{r}$ is the average return over the N-day period. The standard deviation is the unconditional volatility over the N-day period. In this paper, we used σ_t to refer to the volatility at time t and $\hat{\sigma}_t$ to refer to its forecast at time t. The returns series are difference series. As the volatility does not remain constant through time, the conditional volatility that utilizes the volatility reference period is more relevant information for various financial applications such as risk management, asset pricing, investment analysis and option pricing [2].

Conventional volatility forecasting models include GARCH, MLP, GRNN, GMDH and to a less extent RF. Their respective advantages

* Corresponding author.
E-mail addresses: dpkumar@idrbt.ac.in (D. Pradeepkumar), vravi@idrbt.ac.in (V. Ravi).

http://dx.doi.org/10.1016/j.asoc.2017.04.014
1568-4946/

and disadvantages are as follows. GARCH [3], while modeling conditional heteroskedasticity, allows reducing the number of estimated parameters to just a few. Further, various neural networks such as MLP [4], GRNN [5], and GMDH [6] are employed in volatility forecasting as they are data-driven, nonlinear and adaptive. Compared to linear forecasting models, neural networks can capture complex nonlinear relationships. They can generalize well and are good universal approximators [7]. On the other hand, Random Forest [8] is a robust method for both forecasting and classification problems, because of its design that is replete with ensembling several decision trees and random subspace modeling of the feature space. It grows as an ensemble of trees. Each tree, in turn, can handle different types of predictors and missing data too.

Despite their advantages, GARCH variants cannot model asymmetries of the volatility concerning the sign of past shocks. In MLP, convergence is slow, local minima can affect the training process, and it is hard to scale. The GRNN is often more accurate than MLP but relatively insensitive to outliers and requires more memory space to store the model. The GMDH generates a complex polynomial for a simple system. Because of its limited architecture, it does not consider input–output relationship well. It also produces complex networks as it tries to stretch for the last bit of accuracy. Conclusively, we can say that the GMDH is very ineffective in modeling nonlinear systems which exhibit different characteristics in different environments [9]. In RF, a large number of trees result in slow convergence and extreme values are often not accurately predicted as it underestimates high values and overestimates lower values. All of these disadvantages point to the need for better volatility forecasting approaches.

To bridge the gap, one can attempt the existing two QR-based hybrids – QRRF and QRNN, which nobody applied yet to solve this problem. Quantile regression (QR) is a regression mechanism that models the relationship between dependent variable and independent variables at different quantiles. The QRRF [10,11] is a QR-based hybrid and a variant of RF. It can yield predictions of volatility at different quantiles without assuming the unambiguous form of the underlying distribution of data. On the other hand, QRNN [12] is also a QR-based hybrid and a feedforward neural network that can be used to estimate the nonlinear models at different quantiles. It avoids the need for a distributional assumption successfully by applying quantile regression to the historical returns to produce various quantile models. However, the Back Propagation algorithm that trains the QRNN suffers from entrapment in local minima problem; Therefore, QRNN cannot yield accurate predictions. In this paper, we solved this problem by proposing a QRNN trained by PSO [13] named as PSOQRNN.

The major contributions of this paper are:

1. We propose a novel ANN architecture namely PSOQRNN, and apply for forecasting volatility from financial time series. The PSOQRNN is a variant of QRNN with the only difference that the PSO trains QRNN there by yielding optimal weights and biases. We also compared the performance of the PSOQRNN with that of other volatility forecasting models such as GARCH, MLP, GRNN, GMDH, RF, QRRF and QRNN on eight financial time series.
2. In literature, to the best of our knowledge, there is no work reported on financial time series volatility forecasting models involving RF, QRRF, and QRNN. We also presented the application of these volatility forecasting models on eight financial time series. This is the auxiliary contribution of the current study.

The remainder of this paper is structured as follows. Section 2 reviews various volatility forecasting models of financial time series found in the literature. Section 3 presents the overview of the techniques used. Section 4 describes the proposed PSOQRNN and its application in forecasting volatility in detail. Section 5 presents the experimental design best. The results of the proposed model and other volatility forecasting models are presented and discussed in Section 6. Finally, Section 7 concludes the paper.

2. Literature review

There are many types of models have been used for forecasting volatility from financial time series. The reviews of the volatility forecasting models are presented by Franses and McAleer [14], Poon and Granger [2], Andersen et al. [15] and Knight and Satchell [16] respectively. In this section, we briefly present the review of the applications of various forecasting models including GARCH and its variants, MLP, GRNN, GMDH, hybrid models and Quantile regression respectively used for volatility forecasting.

2.1. Forecasting financial time series volatility using Autoregressive Conditional Heteroschedasticity (ARCH) variants

The most widely used are the ARCH models proposed by Engle [17], and then generalized by Bollerslev [3]. These models have led the researchers to model and forecast volatilities from financial time series. Bollerslev et al. [18] presented a review of ARCH models in finance. Later, researchers adopted GARCH and its variants to forecast volatility from different financial time series and obtained better volatility forecasts. Noh et al. [19] assessed the performance of two volatility prediction models – Implied Volatility Regression (IVR) Model and GARCH and concluded that GARCH outperforms IVR by returning a greater profit after experimenting with S&P 500 Stock Index.

Vilasuso [20] adopted the Fractionally Integrated GARCH (FIGARCH) model that was introduced by Baillie et al. [21] to forecast FOREX rate volatility and concluded that FIGARCH model could capture features of exchange rate volatility better than GARCH and Integrated GARCH (IGARCH) models. The FIGARCH model also generated better out-of-sample volatility forecasts after working with exchange rates of Canada, France, Germany, Italy, Japan, and the UK. Chang et al. [22] modeled the volatility of exchange rate RM/Sterling using the Stationary GARCH-in-Mean (GARCH-M) model and concluded that the volatility of RM/Sterling is constant, and the Stationary GARCH-M outperformed other GARCH models in out-of-sample and one-step-ahead forecasting. Agnolucci [23] compared GARCH-type models with an implied volatility model for forecasting volatility in Crude Oil price and concluded that GARCH-type models could better predict the volatility.

2.2. Forecasting financial time series volatility using Artificial Neural Networks (ANNs)

The stand-alone ANNs such as MLP and GRNN are applied to forecast volatility, and these outperformed the GARCH and its variants. Donaldson and Kamstra [24] applied MLP to forecast stock return volatility in London, New York, Tokyo and Toronto and concluded that MLP can capture volatility effects overlooked by GARCH, Exponential GARCH (EGARCH) and Glosten–Jagannathan–Runkle (GJR) models and produced better out-of-sample volatility forecasts. Similarly, Miranda [25] investigated the use of MLP for forecasting volatility implied in the transaction prices of Ibex35 index options and concluded that the MLP yielded better forecasting results. The author also tested and rejected the hypothesis that volatility changes are unpredictable on an hourly basis. Hamid and Iqbal [26] reported that MLP yielded better volatility forecasts than implied volatility forecasts on S&P 500 future stock indices and these are not different from realized volatility.

Aragonés et al. [27] investigated whether ANNs could improve the traditional volatility forecasts from both the time series models

and the implied volatilities obtained from Spanish stock market index, the IBEX-35 and concluded that GRNN yielded better predictions than implied volatility, GARCH, Threshold ARCH (TARCH) and MLP. Mohsen et al. [28] reported that GRNN outperformed GARCH (1,1) in terms of Root Mean Squared Error (RMSE) in forecasting volatility of the prices of two Crude Oil markets of Brent and WTI.

2.3. Forecasting financial time series volatility using hybrid models

During last few years, various hybrid volatility forecasting models are proposed. These hybrid models outperformed statistical volatility forecasting models and various ANNs. Zhuang and Chan [29] developed Hidden Markov Model-GARCH (HMM-GARCH) model that allows both different volatility states in time series and state-specific GARCH models within each state. The authors concluded that the proposed model had overcome excessive persistence problems and outperformed GARCH for both in-sample and out-of-sample evaluation. Roh [30] proposed hybrid volatility forecasting models with a neural network such as Neural Network-Exponentially Weighted Moving Averages (NN-EWMA), NN-GARCH and NN-EGARCH to forecast stock price index and concluded that NN-EGARCH outperformed other proposed hybrid models. Tseng et al. [31] proposed an ANN-EGARCH model of the volatility of the Taiwan stock index option prices. The ANN-EGARCH model could capture the asymmetric volatility. It also simultaneously decreased the stochasticity and nonlinearity of the error term sequence. The proposed model outperformed EGARCH and Grey-EGARCH in terms of Mean Absolute Error (MAE), RMSE, and Mean Absolute Percentage Error (MAPE).

Tseng et al. [32] proposed a new hybrid model for stock volatility prediction by using Grey-GARCH model and concluded that proposed model enhanced the one-period-ahead volatility forecasts of the GARCH model using MAE, RMSE and MAPE. But, the model failed to outperform the GARCH(1,1) model for certain cases. Wang [33] proposed Grey-GJR-GARCH volatility model for forecasting stock index option price. It yielded better predictions than other volatility forecasting approaches including both GARCH and GJR-GARCH in terms of MAE, RMSE, and MAPE. Bildirici and Ersin [34] proposed ANN-APGARCH model that increased the performance of APGARCH model by applying it to the daily returns of Istanbul Stock Exchange. The ANN-extended versions of GARCH models improved forecast results than various variants of GARCH in terms of RMSE.

Hung [35] proposed a Fuzzy-GARCH model for the forecasting volatility of the stock market. The Genetic Algorithm (GA), in the proposed model, is used to achieve a global optimal solution with a fast convergence rate. The author concluded that the proposed hybrid model outperformed the stand-alone GARCH model. Chang et al. [22] introduced a hybrid Adaptive Neuro-Fuzzy Inference System (ANFIS) model based on AR model and volatility for Taiwan Futures Exchange (TAIEX) forecasting. The proposed model outperformed AR model and two other works in literature in terms of RMSE. Hajizadeh et al. [36] proposed a hybrid modeling approach for forecasting the volatility of S&P 500 index returns. In this approach, three variants of GARCH had been calibrated. The EGARCH(3,3) turned out to be the best model. Monfared and Enke [27] proposed hybrid GJR-GARCH Neural Network Model for volatility forecasting and concluded that the proposed hybrid model outperformed the GJR-GARCH model after experimenting with 10 NASDAQ indices.

Komijani et al. [38] introduced a hybrid approach namely Autoregressive Fractionally Integrated Moving Averages-FIGARCH (ARFIMA-FIGARCH) for forecasting Crude Oil prices volatility. The proposed model is based on the long memory property that uses wavelet decomposed data. They concluded that applying hybrid methods led to more accurate forecasts. Kristjanpoller et al. [42] implemented hybrid neural network models for volatility prediction by applying it in three different Latin-American emerging stock markets. It outperformed GARCH and ANN, in their stand-alone mode, in terms of MAPE, whereas, with respect to the measures of both Mean SquareMSE and Mean Absolute Deviation (MAD), the differences between the performances of GARCH and ANN are not significant in most of the cases. Choudhury et al. [39] proposed a novel Self-organizing Map (SOM)-based hybrid clustering technique that is integrated with Support Vector Regression for portfolio selection, accurate price, and volatility predictions. The research considered the top 102 stocks for National Stock Exchange (NSE) Stock market, India to identify a set of best portfolios that an investor can maintain for risk reduction and high profitability. The authors concluded that the work could find various applications in software development acting as investing guide to a target trader in a volatility market.

Rosa et al. [40] suggested an evolving hybrid neural fuzzy network (eHFN) approach for realized volatility forecasting using S&P 500 and Nasdaq (United States), FTSE (United Kingdom), DAX (Germany), IBEX(Spain) and Ibovespa (Brazil) and concluded that the proposed hybrid outperformed GARCH and related stochastic volatility models. Babu and Reddy [41] predicted NSE Indian stock using partitioning-interpolation based Autoregressive Integrated Moving Averages-GARCH (ARIMA-GARCH) model. The proposed model improved prediction accuracy compared to ARIMA, GARCH, and ANN. It also preserved the data trend over the prediction horizon better than these models. Kristijanpoller and Minutolo [42] predicted Gold Price volatility using hybrid ANN-GARCH model that outperformed GARCH. The authors also demonstrated an innovative method to determine which financial variables are the most important in affecting the volatility of spot gold prices and future prices. Dash et al. [43] proposed a new hybrid model namely Interval Type2 Fuzzy-Computationally Efficient-EGARCH (IT2F-CE-EGARCH) integrating an Interval Type2 Fuzzy Logic System (IT2FLS) with a Computationally Efficient Link Artificial Neural Network (CEFLANN) and EGARCH model for accurate forecasting and modeling of stock market (BSE Sensex and CNX Nifty) volatility. The Differential Harmony Search (DHS) algorithm gets used for optimizing the parameters of the entire fuzzy time series model.

2.4. Forecasting financial time series volatility using Quantile Regression (QR)

There are, however, very few works reported related to the application of Quantile Regression (QR) to forecast financial time series volatility. Taylor [12] proposed a QR-based approach to estimate the conditional probability distribution of multi-period financial returns. After experimenting with exchange rates, the author concluded that the QR-based approach is best suitable to estimate the conditional density compared to GARCH-based quantile estimates. Huang et al. [44] adopted QR to forecast volatility from exchange rates. After experimenting it with nine exchange rates using 19 years of data, the authors concluded that QR produced more reliable volatility forecasts. Except for these works, to the best of our knowledge, there is no other published work related to volatility forecasting using quantile regression and related hybrids. In this context, our work bridges the gap by adopting two quantile regression based hybrids including QRRF and QRNN to forecast volatility. In addition to these, there are no works found that applied RF in the context of volatility forecasting.

3. Overview of techniques used

In this paper, we employed various volatility forecasting models including GARCH, MLP, GRNN, GMDH, RF, QRRF and QRNN. This section presents the overview of the techniques employed.

3.1. Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model

GARCH model [3] is a generalized version of ARCH model [17]. The term AR comes from the fact that this model is an autoregressive model in squared returns. The term conditional comes from the fact that in this model, next period's volatility is predicted based on information of this period. Heteroskedasticity means non-constant variance/volatility. It is based on the assumption that volatility forecast changing in time depends on lagged values of standard deviation of asset returns. It differs from ARCH by the form of σ_t. In the GARCH model, the forecast σ_t can be obtained using a constant depicting constant variance throughout trading days, a sum of p weighted products of last periods' forecasts (GARCH terms) and the sum of q weighted products of last periods' squared residual terms (ARCH terms). The GARCH (p, q) model can be defined as in Eq. (2).

$$\begin{aligned} r_t &= \mu + \epsilon_t \\ \epsilon_t &= \sqrt{\sigma_t} z_t \\ \sigma_t &= K + \sum_{i=1}^{p} A_i \sigma_{t-i} + \sum_{j=1}^{q} B_j \epsilon_{t-j}^2 \end{aligned} \tag{2}$$

where r_t is the return at time t, μ is the mean return, ϵ_t is return residual term (also known as innovation) which is a product of a stochastic piece z_t and a time-dependent volatility, a random variable z_t is a standardized, independent and identically distributed (i.i.d) random draw from normal distribution with zero mean/unit variance. The parameters of the model are K, A_1, A_2, . . ., A_p, B_1, B_2, . . ., B_q;$K > 0$ is a constant, $A_i \geq 0$, $B_j \geq 0$ are the coefficients used in multiplying GARCH and ARCH terms respectively, p is the degree of GARCH terms σ_t and q is the degree of ARCH terms ϵ_t which are, in turn, obtained by using Akaike Information Criterion (AIC) as in Autoregressive Integrated Moving Averages (ARIMA) model.

3.2. Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron [4] is one of the most commonly employed neural network architectures. It is the widely used ANN for pattern classification and prediction. It can yield accurate predictions for challenging problems. It is too popular to be described here in detail.

3.3. General Regression Neural Network (GRNN)

The GRNN [5] is a 4-layered neural network that has the unique features of learning in one pass and simple training algorithm. In GRNN, each training sample is considered as a kernel during the training process, and the estimation is based on non-parametric regression analysis. It is discriminative against occasional outliers and erroneous observations. It can converge to any arbitrary nonlinear function of data with only a few training samples. The additional knowledge needed to get the best fit of the function is relatively small. These features make GRNN very useful tool to perform predictions in real time.

3.4. Group Method of Data Handling (GMDH)

GMDH, proposed by Ivakhnenko [6], is a self-organized feed-forward network based on short-term polynomial transfer functions, called Ivakhnenko polynomials. The coefficients of these transfer functions are obtained using least square regression [45]. It is best suitable ANN for dealing with inaccurate, noisy, or small data sets. It yields higher accuracy and is a simpler structure than traditional ANN models [46]. It is the earliest proposed deep learning neural network architecture, where nodes in the hidden layers are dropped if they are found to be not having sufficient predictive power.

3.5. Random Forest (RF)

In order to understand Random Forest [8], we first need to appreciate how Classification and Regression Tree (CART) [47] works. CART is a recursively partitioned binary decision tree, and it is one of the non-parametric statistical techniques that are used to solve classification and regression problems using the 'if-then' rules it generates in the process of training [47]. It can model nonlinearity very well, and it yields readily interpretable results. However, it is worth to note that CART becomes unstable even when there are small changes in the training data. A bootstrap aggregating technique namely bagging can be applied to overcome the problem of instability, [48]. In bagging, a large number of bootstrap samples are taken from the dataset and a single tree is fit to each bootstrap sample. The predictions are obtained using the average of predictions obtained from each of the fitted trees.

The Random Forest (RF) is a modified version of bagged tree. In this RF, a random subset of the predictor variables is used for each tree and at each node, which is not possible with bagging. It includes the effective methods of handling missing data too.

3.6. Quantile Regression Random Forest (QRRF)

Quantile ($0 < \tau < 1$) is a boundary of uniformly sized consecutive subsets. The median (50th percentile), 70th percentile, 90th percentile etc are examples of different quantiles. Both simple linear regression and multiple linear regression models can model the behavior of predictand by assuming average Gaussian distribution whereas Quantile Regression [49] can model it at different quantiles. This departure results in various conclusions compared to examining only the average of the predictand. Quantile Regression Random Forest (QRRF), introduced by Meinshausen [10], is a generalization of RF. Each node in an RF stores only the mean of the observations excluding other information different from mean. In contrast, each node in QRRF keeps track of the spread of the predictand. It allows the construction of prediction intervals that could cover highly probable new observations.

3.7. Particle Swarm Optimization (PSO)

PSO, developed by Kennedy and Eberhart [13], is very simple to implement and has very few parameters to tweak. It progresses towards the solution by mutual sharing of knowledge of every particle collectively. In PSO, the population of particles with velocity V_{id}^{old} is initially randomly generated. Each particle's velocity gets updated with respect to its corresponding old position x_{id}^{old} using neighborhood best p_{id} (see Eqs. (3) and (4)) and global best particle p_{gd} until the convergence criterion is satisfied. After the

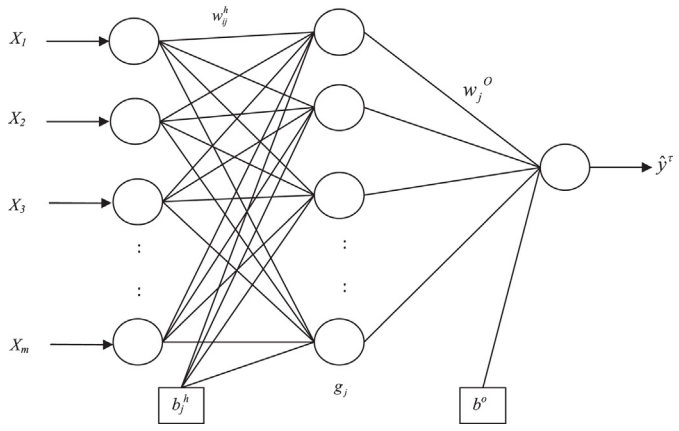

Fig. 1. Quantile Regression Neural Network.

convergence criterion is satisfied, the global best particle is the optimal solution.

$$V_{id}^{New} = \gamma * V_{id}^{Old} + C_1 * rand * (p_{id} - X_{id}^{old}) + C_2 * rand * (p_{gd} - X_{id}^{old}) \quad (3)$$

$$X_{id}^{New} = X_{id}^{Old} + V_{id}^{New} \quad (4)$$

where V_{id}^{Old} is old velocity, V_{id}^{New} is updated velocity, X_{id}^{Old} is old particle, X_{id}^{New} is updated particle, p_{id} is a local best particle, p_{gd} is the global best particle, C_1 and C_2 are two positive constants, γ is the inertia weight and finally, *rand* is a random number between 0 and 1.

3.8. Quantile Regression Neural Network (QRNN)

For the purpose of forecasting, the most extensively used neural network is single hidden-layer feedforward network [7]. QRNN [50], whose architecture is depicted in Fig. 1, is one such neural network. It consists of m input neurons for predictors $X_1, X_2, \ldots, X_m$, which are connected to n hidden neurons in a single hidden layer, which, in turn, are connected to one output neuron that yields predictand. The difference between traditional feedforward ANN and QRNN lies in the process of training. If Eq. (5) is used as the cost function to train the feedforward ANN, then the outputs are estimates of the conditional regression quantiles [51] and the resultant model is QRNN [12]. QRNN is a flexible model that represents nonlinear predictor-predictand relationships, including the ones involving interactions between predictors, without prior specification of the form of the relationships by the modeler [50].

$$E^\tau = \frac{1}{N}\sum_{t=1}^{N}\rho^\tau(y_t - \hat{y}_t) \quad (5)$$

$$\rho^\tau(u) = \begin{cases} \tau u, & \text{if } u \geq 0 \\ (\tau - 1)u, & \text{otherwise} \end{cases} \quad (6)$$

In Eq. (5), y_t is actual value at time t, $\hat{y}_t$ is predicted value at time t and N is the total number of observations trained. $\rho^\tau(.)$ is the tilted absolute value function (also known as the check, tick or pinball function). It accepts input namely u with the error value obtained from $(y_t - \hat{y}_t)$ and returns the value using Eq. (6).

The output at the hidden layer can be calculated using Eq. (7):

$$g_{j,t} = \tanh\left(\sum_{i=1}^{m} X_{i,t} w_{ij}^h + b_j^h\right) \quad (7)$$

where $j = 1, 2, \ldots, n$, w_{ij}^h is the weight at connection from ith node of input layer to jth node of hidden layer and b_j^h is a bias added to jth node of hidden layer. An estimate of the conditional τ-quantile, $\hat{y}_t^\tau$ is given by Eq. (8):

$$\hat{y}_t^\tau = f\left(\sum_{j=1}^{n} g_{j,t} w_j^O + b^O\right) \quad (8)$$

where $f(.)$ is an output layer transfer function, w_j^O is the weight at the connection from jth node of the hidden layer to the node of output layer and b^O is the bias added to the node of the output layer. Usually, the activation function at output layer is linear transfer function. In our proposed model, it is same as hidden layer activation function. The output can be calculated at each time t for each quantile individually.

It is important to note that all of these volatility forecasting models used in this paper accept the previous day volatility σ_{t-1} and current day innovation ϵ_t as inputs and produce the current day volatility σ_t.

4. Proposed methodology

In the current work, we propose a novel ANN namely PSO-trained Quantile Regression Neural Network (PSOQRNN) and its application for predicting volatilities from financial time series. The architecture of PSOQRNN is the same as that of QRNN (see Fig. 1). The weights and biases are estimated using PSO. Typically, weights and biases in QRNN are determined using Back Propagation (BP) algorithm at different quantiles. The BP algorithm has the drawbacks such as large computational time, slow convergence rate and entrapment in local minima [52]. The PSO overcomes these drawbacks. PSO is a population-based evolutionary technique which is derivative-free, has fewer parameters to tweak in, has fast convergence rate and provides a near-global optimal solution. Therefore, it was used effectively by researchers as an alternative to BP algorithm to train ANN [53–56]. Now, in the current work, it is also employed for training the QRNN.

The reasons for choosing PSO to train the QRNN are as follows. Compared with other evolutionary algorithms, PSO is very intuitive and flexible, less sensitive to the nature of the objective function and is able to handle objective functions with stochastic nature. Moreover, the heuristics involved in PSO are easy to comprehend and implement. Further, it has fewer user-defined parameters to tweak and does not require a good initial solution to start its iteration process [57,58].

In the proposed methodology of obtaining volatility predictions using PSOQRNN, (1) the return series are obtained from original financial time series using Return Series Generator, (2) the innovations series is obtained from the returns series obtained using Innovations Series Generator, (3) all volatilities are computed using Volatility generator, (4) both volatilities and innovations series are divided into training set and test set using partition generator, (5) the training sets are input to PSOQRNN, (6) the QRNN is trained using weights obtained from PSO, (7) the trained PSO-QRNN yields accurate training set predictions, (8) the test sets are input to trained PSOQRNN and (9) finally, the predictions of test set volatilities are obtained. This procedure is entirely depicted by Fig. 2.

Fig. 3 depicts the same process of obtaining predictions of volatilities using various forecasting models. For forecasting the volatilities, GARCH, MLP,GRNN,GMDH and RF are trained. As for as QRRF and QRNN are concerned, the models are trained at different quantiles. Finally, PSOQRNN is trained with the weights obtained by PSO at different quantiles.

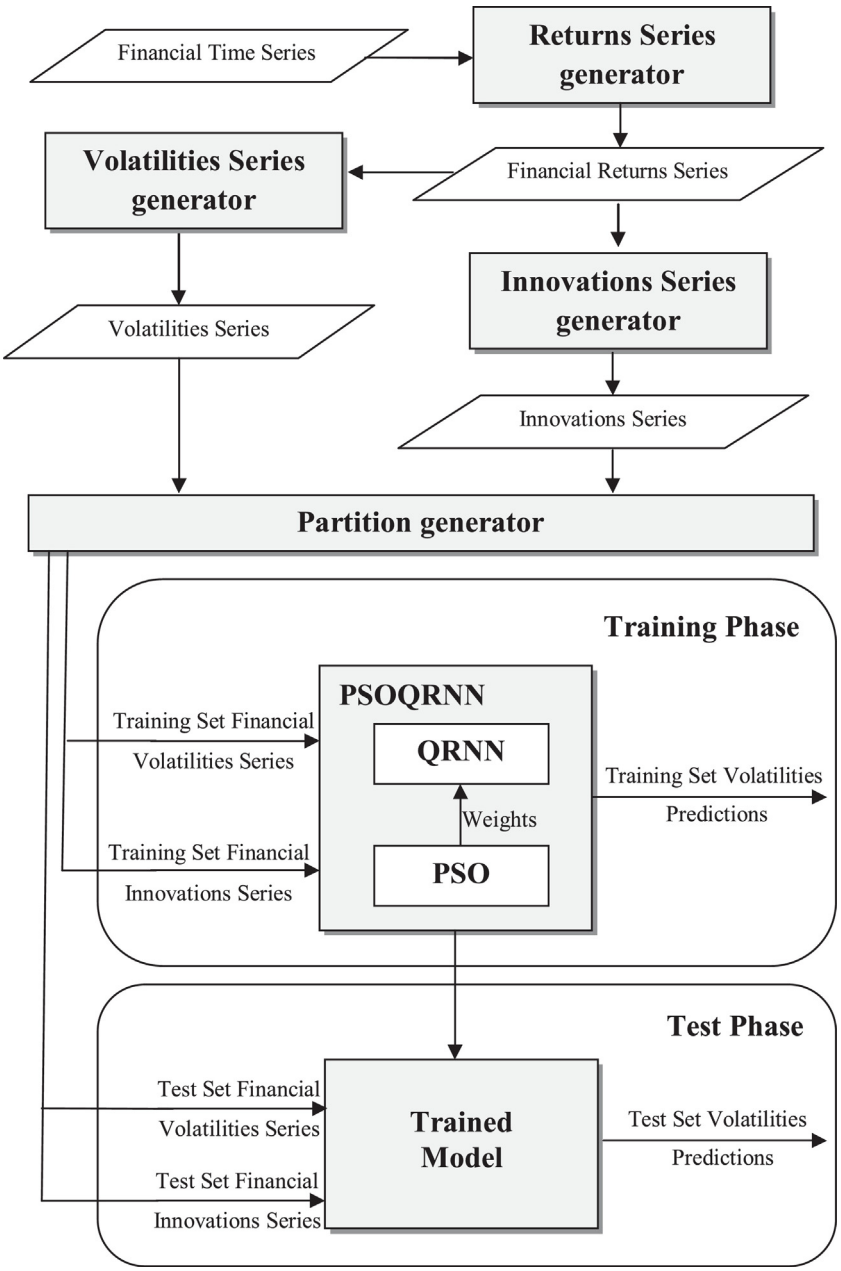

Fig. 2. Architecture of proposed methodology using PSOQRNN.

The step-by-step procedure of obtaining predictions of volatilities using the proposed PSOQRNN is clearly described in detail as follows. Let $Y = \{y_1, y_2, \ldots, y_k, y_{k+1}, \ldots, y_N\}$ be a set of N observations of a financial variable recorded at $t = \{1, 2, \ldots, k, k+1, \ldots, N\}$ respectively. Then the volatilities of financial variable can be forecast as follows:

1. Obtain returns series r_t from Y using Eq. (9) (returns series generator):

$$r_t = \frac{y_t - y_{t-1}}{y_t} \tag{9}$$

where y_t is the actual observation of time series at the time t with $t = \{1, 2, \ldots, k, k+1, \ldots, N\}$. Usually, the volatilities from financial time series can be obtained with the help of returns series.

2. Obtain the innovations series ϵ_t from r_t using Eq. (10) (innovations series generator):

$$\epsilon_t = r_t - \bar{r} \tag{10}$$

where r_t is return of financial variable at the time t and $\bar{r}$ is mean return over n periods. The innovations (residual terms) series together with returns series help the model in predicting volatilities.

3. Calculate all of the volatilities σ_t with $t = \omega, \omega+1, \ldots, k, k+1, \ldots, N$ based on window length ω, trading days 252, and standard deviation $std(.)$ using Eq. (11) (volatilities generator):

$$\sigma_t = std(r_i, r_{i+1}, \ldots, r_{i+\omega-1}) * \sqrt{252}; \qquad i = 1, 2, \ldots, N - \omega \tag{11}$$

On average, there are 252 trading days in a financial year. So, a moving window length of past 252 days [42] is used to obtain volatilities. In other words, each day's volatility is measured with the help of its previous 252 days' returns. In order to

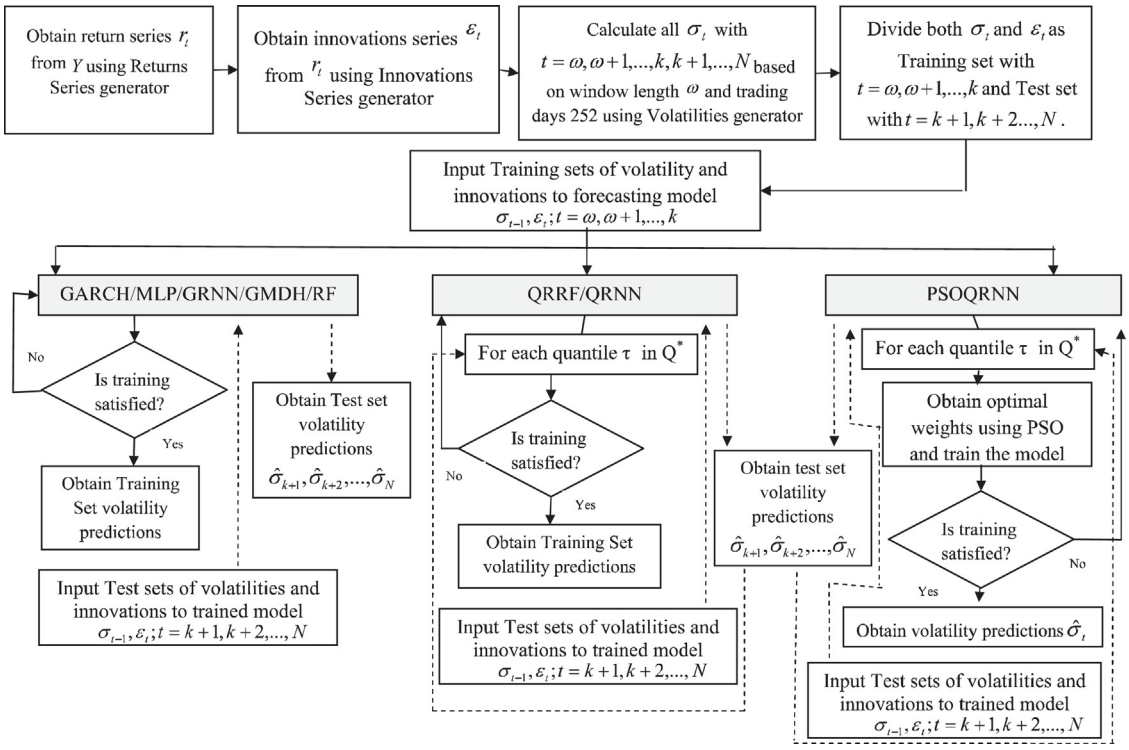

Fig. 3. Detailed flow of forecasting volatility from financial time series using various forecasting models.

present the volatility in annualized terms, we simply need to multiply the standard deviation value by the square root of 252.

4. Partition both of the volatilities series σ_t and the innovations series ϵ_t as training set with $t = \omega, \omega+1, \ldots, k$ and test set with $t = k+1, k+2, \ldots, N$ respectively using partition generator.
5. Input the training set $(\sigma_{t-1}, \epsilon_t, \sigma_t;\ t = \omega, \omega+1, \ldots, k)$ and Quantile values $Q^* = \{\tau_1, \tau_2, \ldots, \tau_q\}$, $(0 < \tau_i < 1;\ i = 1, 2, \ldots, q)$. Let n_H represent the number of hidden nodes of PSOQRNN. Here, σ_{t-1} and ϵ_t are inputs while σ_t is output to PSOQRNN.
6. For each quantile value τ_i in Q^*, repeat steps 7, 8, 9 and 10.
7. Obtain weights using PSO algorithm as follows:
   (a) Initialize the particles with random values within specified range [0,1]. Each particle is a vector, whose length is determined by the total number of weights and biases of the PSOQRNN. For example, if $n_H = 3$, then the length of a particle is 13 (No. of weights = 2 * 3 + 3 * 1 = 9, biases = 3 + 1 = 4).
   (b) Evaluate the fitness of each particle using quantile regression error function as in Eqs. (5) and (6).
   (c) Update individual and global best fitnesses fitness values and positions.
   (d) Update velocity and position of each particle using Eqs. (3) and (4) respectively.
   (e) Repeat steps (b), (c) and (d) until all iterations are finished. The coordinates of the global best particle are the weights and bias values to PSOQRNN for training the network.
8. Train PSOQRNN with the weights obtained from PSO in order to yield predictions of training set volatilities $\hat{\sigma}_t$ where $t = \omega, \omega+1, \ldots, k$.
9. Input test sets of volatilities and innovations σ_{t-1}, ϵ_t with $t = k+1, k+2, \ldots, N$ to the trained PSOQRNN.
10. Obtain the predictions of Test set volatilities $\hat{\sigma_{k+1}}, \hat{\sigma_{k+2}}, \ldots, \hat{\sigma_N}$.

Computational complexity is the count of function evaluations. In the proposed PSOQRNN, there are activation functions at each node and error fitness function to be minimized. If there are χ nodes in PSOQRNN, then it performs χ^2 multiplications needed to compute activation functions of all neurons. PSO in PSOQRNN evaluates the error fitness function for λ iterations using population size Z. Therefore, the total computational complexity of PSOQRNN is $O(\chi^2.\ \lambda.\ Z)$.

5. Experimental design

This section presents the description of various datasets used, performance measures used and the execution environment used to show the effectiveness of the proposed PSOQRNN along with other volatility forecasting models.

5.1. Datasets used

The datasets collected are of daily US Dollar (USD) exchange rates with respect to four currencies – Japanese Yen (JPY), Great Britain Pound (GBP), Euro (EUR) and Indian Rupees (INR), Gold Price in terms of USD, Crude Oil Price in terms of USD, S&P 500 Stock Index, and NSE India Stock Index. These eight financial datasets are used for testing the effectiveness of the proposed volatility forecasting models. The foreign exchange data are obtained from http://www.federalreserve.gov/releases/h10/hist/, Gold Price data is obtained from http://www.quandl.com/LBMA/GOLD-Gold-Price-London-Fixing, Crude Oil Price data is obtained from https://www.quandl.com/data/FRED/DCOILBRENTEU-Crude-Oil-Prices-Brent-Europe, S&P 500 Stock Index data is obtained from https://www.quandl.com/data/YAHOO/INDEX_GSPC-S-P-500-Index and NSE India Stock Index is obtained from https://www.quandl.com/data/GOOG/NSE_BANKINDIA-Bank-of-India-BANKINDIA. Each of the datasets is divided into both Training set (80%) and Test set (20%) respectively (see Table 1).

5.2. Performance measures used

Mean Squared Error (MSE), Directional Change Statistic (Dstat) and Theil's Inequality Coefficient (U) are used to measure the

Table 1
Datasets used.

Dataset	Total observations	Training set	Test set
USD–JPY	6036	4829	1207
USD–GBP	6036	4829	1207
USD–EUR	3772	3018	754
USD–INR	6028	4825	1203
Gold Price	7602	6081	1521
Crude Oil Price	6857	5486	1371
S&P 500 Stock Index	7581	6065	1516
NSE India Stock Index	4232	3386	846

performance of the proposed model and are defined as in Eqs. (12)–(14):

$$MSE = \frac{\sum_{t=1}^{N}(\sigma_t - \hat{\sigma_t})^2}{N} \tag{12}$$

$$Dstat = \frac{1}{N}\sum_{t=1}^{N} a_t * 100\% \tag{13}$$

$$\text{where } a_t = \begin{cases} 1, & \text{if } (\sigma_{t+1} - \sigma_t) * (\hat{\sigma_{t+1}} - \hat{\sigma_t}) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

$$U = \frac{\sqrt{\frac{1}{N}\sum_{t=1}^{N}(\sigma_t - \hat{\sigma_t})^2}}{\sqrt{\frac{1}{N}\sum_{t=1}^{N}(\sigma_t)^2} + \sqrt{\frac{1}{N}\sum_{t=1}^{N}(\hat{\sigma_t})^2}} \tag{14}$$

In Eqs. (12)–(14), N is the number of forecasts obtained, σ_t is the actual volatility at time t and $\hat{\sigma_t}$ is forecasted volatility at time t respectively.

The MSE (see Eq. (12)) measures the average of the squares of the errors. MSE is useful when we are concerned about significant errors whose negative consequences are proportionately much bigger than equivalent smaller one [59]. In forecasting volatility, it is not only important to measure the accuracy of predictions but also the directional change of time series. For this purpose, Yao and Tan [60] developed a measure (expressed in percentages) namely Dstat as in Eq. (13). We used this measure in our work. It is well known that Theil's Inequality Coefficient [61,59] measures how well a forecasted time series is closer to actual time series. Generally, the value of U lies in between 0 and 1. $U = 0$ means that $\sigma_t = \hat{\sigma_t}$ for all observations and there is a perfect fit and $U = 1$ means that the performance is bad.

5.3. Execution environment

The execution of the proposed work is carried out using Windows 7 *Professional*® platform. However, it can also be carried out on other platforms. It is carried out under the system with the specifications of 8 GB RAM, 500 GB HDD. Table 2 presents various tools employed in numerous experiments with the datasets.

Table 2
Tools and techniques used.

Technique used	Used for	Tool used
MLP/GRNN/GMDH (http://www.neuroshell.com/)	Obtaining predictions	Neuroshell 2. 0®
GARCH	Obtaining predictions	*MATLAB*®
RF (http://cran.r-project.org/web/packages/randomForest/)	Obtaining predictions	R
QRRF (http://cran.r-project.org/web/packages/quantregForest/)	Obtaining predictions	R
QRNN (http://cran.us.r-project.org/web/packages/qrnn/)	Obtaining predictions	R
PSO	Obtaining coefficients	Java
PSOQRNN	Obtaining predictions	Java

Various parameters are tuned to obtain better results for all datasets. In literature, even though, Masters [62] recommended that the number of hidden nodes as $\sqrt{N_{in} * N_O}$, where N_{in} represents No. of input nodes and N_O is number of output nodes. We experimented with various number of hidden nodes (2, 3, 4 and 5). After thorough experimentation, the number of hidden nodes of PSOQRNN is fixed as follows: 3 is selected for USD–JPY dataset, 2 is selected for USD–GBP, 5 is selected for USD–EUR, 4 is selected for USD–INR, 3 is selected for Gold Price (USD), 5 is selected for Crude Oil Price (USD), 3 is selected for S&P Stock Index and 5 is selected for NSE India Stock Index datasets respectively. Therefore, the recommendation made by Timothy did not fit in our work. The quantile (τ) values fixed are with in the range [0, 1]. Among these, $\tau = 0.5$ helped PSOQRNN in yielding better predictions. The parameters commonly used in PSO for all datasets are as follows. The inertia weight (γ), controls the momentum of a particle. V_{max} and V_{min} determine the maximum and minimum changes a particle can undergo in its positional coordinates during an iteration respectively. Shi and Eberhart found that "When $V_{max} \geq 3$, $\gamma = 0.8$ is a good choice" [63]. Based on this, in the current work, we selected −5 for V_{min}, 5 for V_{max} and $\gamma = 0.8$. In PSO literature, it is quite a common practice to limit the swarm size to the range 20–60 [63–65]. In the current work, we selected the number of particles as 60 after thorough experimentation which is the same that we carried out in [66]. Usually, the acceleration coefficients C_1(Self-confidence) and C_2(Swarm confidence) are within the range of [0, 4] [67,65]. In the current work, we selected $C_1 = C_2 = 2$ which is the same that we carried out in [66], and the maximum number of iterations is 5000 so that global best particle is converged to near optimal solution. These parameters are chosen after rigorous experimentation with PSOQRNN.

Table 3 presents various parameters of forecasting models after thorough experimentation. These parameters helped the models in yielding better predictions. In GARCH model, p determines the order of GARCH terms and q determines the order of ARCH terms. For all datasets, $p = 1$ and $q = 1$ are obtained as the best parameters. In MLP, the learning rate ($0 < \eta < 1$) controls the size of weight and bias changes in learning of the training algorithm and momentum ($0 < \alpha < 1$) simply adds a fraction α of the previous weight update to the current one. In this paper, we have chosen $\eta = 0.6$ and $\alpha = 0.9$ for all datasets. We experimented with various number of hidden nodes of MLP(3, 4 and 5) and, for each dataset, different numbers of hidden nodes are selected. In GRNN, the only adjustable parameter is smoothing factor(ρ) for kernel function and its value is varied as per dataset. For GMDH, as part of Neuroshell, the parameters selected (MD, MC and MO) are as follows. Model Diversity (MD) defines the maximum number of Survivors which are allowed to pass from the output of each layer to the input of the next one. Model Complexity (MC) determines the allowed length of the formula of a candidate for survival by adjusting the relative penalty for overall model complexity at the output of each layer. Both MD and MC are fixed at the option of "Medium" for all datasets. Model Opitmization (MO) is set to "Smart" which is an optimal tradeoff between calculation speed and model quality. In RF and QRRF, *ntree* determines number of trees to grow. For both models, *ntree* is chosen as 100. Quantile determines boundary of data and, in QRRF and QRNN, the quantile (τ) values fixed are with in the range [0, 1]. Among these, $\tau = 0.5$ helped both models in yielding better predictions. The number of hidden nodes chosen in QRNN are same as that of PSOQRNN.

6. Results and discussion

This section presents the dataset-wise results obtained and discusses them.

Table 3
Parameters of various models tuned for different datasets.

Dataset	Parameters used	Dataset	Parameters used
USD–JPY	GARCH ($p=1$, $q=1$) MLP ($\eta=0.6$, $\alpha=0.9$,$n_{HM}=5$) GRNN ($\rho=0.0332941$) GMDH (MD = Medium, MC = Medium, MO = Smart) RF ($ntree=100$) QRRF ($\tau=0.5$, $ntree=100$) QRNN ($\tau=0.5$, $n_{HQ}=3$)	Gold Price (USD)	GARCH ($p=1$, $q=1$) MLP ($\eta=0.6$, $\alpha=0.9$, $n_{HM}=4$) GRNN ($\rho=0.0449412$) GMDH (MD = Medium, MC = Medium, MO = Smart) RF ($ntree=100$) QRRF ($\tau=0.5$, $ntree=100$) QRNN ($\tau=0.5$, $n_{HQ}=3$)
USD–GBP	GARCH ($p=1$, $q=1$) MLP ($\eta=0.6$, $\alpha=0.9$, $n_{HM}=4$) GRNN ($\rho=0.0138824$) GMDH (MD = Medium, MC = Medium, MO = Smart) RF ($ntree=100$) QRRF ($\tau=0.5$, $ntree=100$) QRNN ($\tau=0.5$, $n_{HQ}=2$)	Crude Oil Price (USD)	GARCH ($p=1$, $q=1$) MLP ($\eta=0.6$, $\alpha=0.9$, $n_{HM}=5$) GRNN ($\rho=0.0138824$) GMDH (MD = Medium, MC = Medium, MO = Smart) RF ($ntree=100$) QRRF ($\tau=0.5$, $ntree=100$) QRNN ($\tau=0.5$, $n_{HQ}=5$)
USD–EUR	GARCH ($p=1$, $q=1$) MLP ($\eta=0.6$, $\alpha=0.9$, $n_{HM}=3$) GRNN ($\rho=0.0332941$) GMDH (MD = Medium, MC = Medium, MO = Smart) RF ($ntree=100$) QRRF ($\tau=0.5$, $ntree=100$) QRNN ($\tau=0.5$, $n_{HQ}=5$)	S&P 500 Stock Index	GARCH ($p=1$, $q=1$) MLP ($\eta=0.6$, $\alpha=0.9$, $n_{HM}=5$) GRNN ($\rho=0.0138824$) GMDH (MD = Medium, MC = Medium, MO = Smart) RF ($ntree=100$) QRRF ($\tau=0.5$, $ntree=100$) QRNN ($\tau=0.5$, $n_{HQ}=3$)
USD–INR	GARCH ($p=1$, $q=1$) MLP ($\eta=0.6$, $\alpha=0.9$, $n_{HM}=5$) GRNN ($\rho=0.0216471$) GMDH (MD = Medium, MC = Medium, MO = Smart) RF ($ntree=100$) QRRF ($\tau=0.5$, $ntree=100$) QRNN ($\tau=0.5$, $n_{HQ}=4$)	NSE India Stock Index	GARCH ($p=1$, $q=1$) MLP ($\eta=0.6$, $\alpha=0.9$, $n_{HM}=5$) GRNN ($\rho=0.0216471$) GMDH (MD = Medium, MC = Medium, MO = Smart) RF ($ntree=100$) QRRF ($\tau=0.5$, $ntree=100$) QRNN ($\tau=0.5$, $n_{HQ}=5$)

p = order of GARCH terms, q = order of ARCH terms, η = learning rate, α = momentum rate, n_{HM} = No. of hidden nodes of MLP, ρ = Smoothing factor, τ = Quantile, $ntree$ = No. of trees to grow, n_{HQ} = No. of hidden nodes of QRNN, MD = Model Diversity, MC = Model Complexity, MO = Model Optimization. Note: For all of the above models, No. of input variables = 2 and No. of output variables = 1.

6.1. USD–JPY

Table 4 presents MSE, Dstat and Theil's U values obtained by using the volatility forecasting models including GARCH, MLP, GRNN, GMDH, RF, QRRF, QRNN and PSOQRNN for both of the training set and the test set of USD–JPY. For the USD–JPY dataset, PSOQRNN yielded 80% less MSE than that of QRNN on Test set. Further, PSOQRNN yielded the highest Dstat too. These reveal that PSOQRNN yielded better predictions than QRNN in both training set and test set respectively. Figs. 4 and 5 depict the predictions of volatilities of both training set and test set respectively. The minimum MSE obtained by the proposed PSOQRNN in 30 runs is also better when compared to the MSE obtained by other techniques. From Table 4, it is also evident that QR hybrids best predicted the volatility of financial time series than GARCH, MLP, GRNN, GMDH, and RF. The results of the proposed PSOQRNN model are highlighted in bold faces in Tables 4–11 because the model outperformed all other volatility prediction models specified.

6.2. USD–GBP

Table 5 presents MSE, Dstat and Theil's U values obtained by using volatility forecasting models including GARCH, MLP, GRNN, GMDH, RF, QRRF, QRNN and PSOQRNN for both of the training set and the test set of USD–GBP. For the USD–GBP dataset, PSOQRNN yielded 87% less MSE than that of QRNN. Further, PSOQRNN yielded the highest Dstat and Theil's Inequality Coefficient too. These reveal that PSOQRNN yielded better predictions than QRNN in both training set and test set. Figs. 6 and 7 depict the predictions of volatilities of both training set and test set respectively. The minimum MSE obtained by the proposed PSOQRNN in 30 runs is least when compared to the MSE obtained by other techniques. From Table 5,

Table 4
Results of volatility forecasting models for USD–JPY data.

Forecasting model	Training set			Test set		
	MSE	Dstat	Theil's U	MSE	Dstat	Theil's U
GARCH	0.000247	49.94	0.0722	0.000268	51.61	0.0793
MLP	0.000699	46.29	0.1239	0.000505	47.14	0.1089
GRNN	0.000214	50.48	0.0663	0.000247	52.19	0.0744
GMDH	0.000217	52.42	0.0666	0.000243	54.68	0.0739
RF	0.000249	51.79	0.0715	0.000272	55.09	0.0779
QRRF	0.0000007295	98.31	0.0038	0.0000001729	98.09	0.0062
QRNN	0.0000008898	94.80	0.0043	0.0000005145	95.60	0.0034
PSOQRNN	0.0000006601	99.33	0.0037	**0.0000001027**	**99.58**	**0.0042**

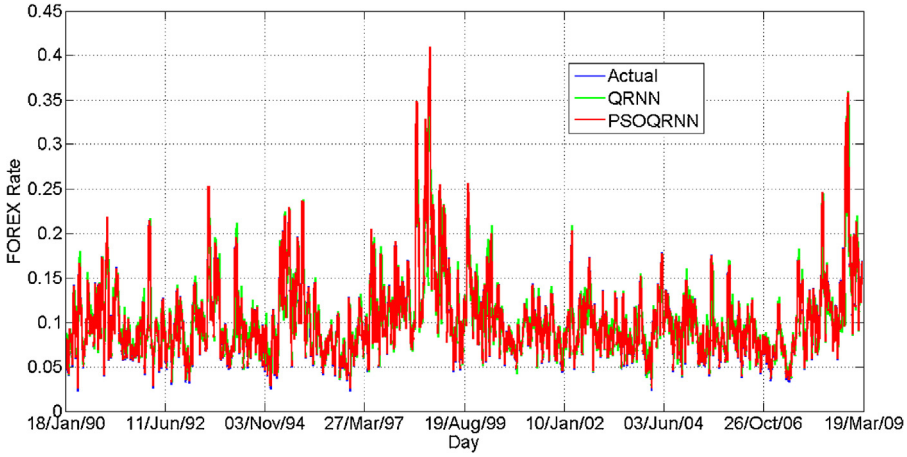

Fig. 4. Predictions of training set of USD–JPY volatilities.

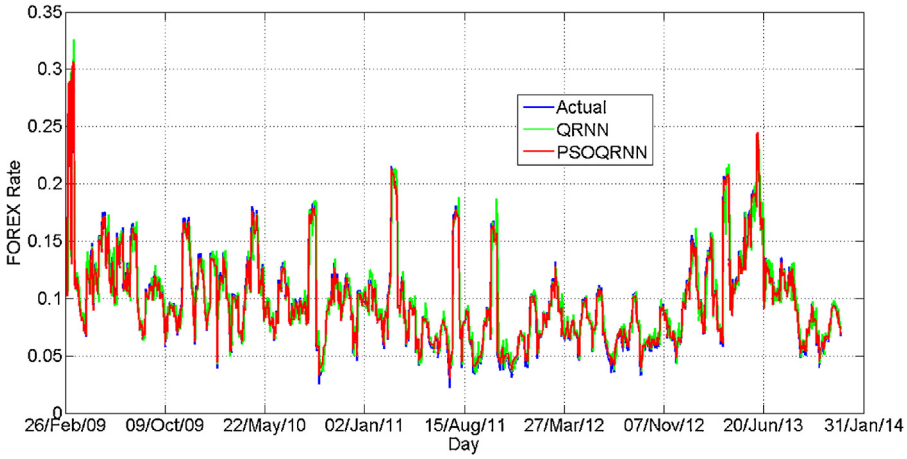

Fig. 5. Predictions of test set of USD–JPY volatilities.

it is also evident that QR hybrids best predicted the volatility of USD–GBP than GARCH, MLP, GRNN, GMDH, and RF.

6.3. USD–EUR

Table 6 presents MSE, Dstat and Theil's U values obtained by using volatility forecasting models including GARCH, MLP, GRNN, GMDH, RF, QRRF, QRNN and PSOQRNN for both of the training set and the test set of USD–EUR. For the USD–EUR dataset, PSOQRNN yielded 85% less MSE than that of QRNN on Test set. Further, PSOQRNN yielded the highest Dstat and Theil's Inequality Coefficient too. These reveal that PSOQRNN yielded better predictions than QRNN in both training set and test set. Figs. 8 and 9 depict the predictions of volatilities of both training set and test set respectively. The minimum MSE obtained by the proposed PSOQRNN in 30 runs is also least when compared to the MSE obtained by other techniques. From Table 6, it is also evident that QR hybrids best predicted the volatility of USD–EUR than GARCH, MLP, GRNN, GMDH, and RF.

6.4. USD–INR

Table 7 presents MSE, Dstat and Theil's U values obtained by using volatility forecasting models including GARCH, MLP, GRNN, GMDH, RF, QRRF, QRNN and PSOQRNN for both of the training set and the test set of dataset of USD–INR. For the USD–INR dataset,

Table 5
Results of volatility forecasting models for USD–GBP data.

Forecasting model	**Training set**			**Test set**		
	MSE	Dstat	Theil's U	MSE	Dstat	Theil's U
GARCH	0.000154	49.22	0.0671	0.0001240	48.38	0.0600
MLP	0.000504	50.19	0.1227	0.0002920	48.05	0.0920
GRNN	0.000108	53.55	0.0550	0.0001017	50.04	0.0532
GMDH	0.000131	52.37	0.0604	0.0001020	51.20	0.0532
RF	0.000152	52.75	0.0653	0.0001181	51.53	0.0572
QRRF	0.0000002029	98.60	0.0024	0.000000205	98.84	0.0024
QRNN	0.0000007900	94.39	0.0047	0.000000756	95.36	0.0046
PSOQRNN	0.0000004410	99.06	0.0035	**0.000000098**	**99.17**	**0.0017**

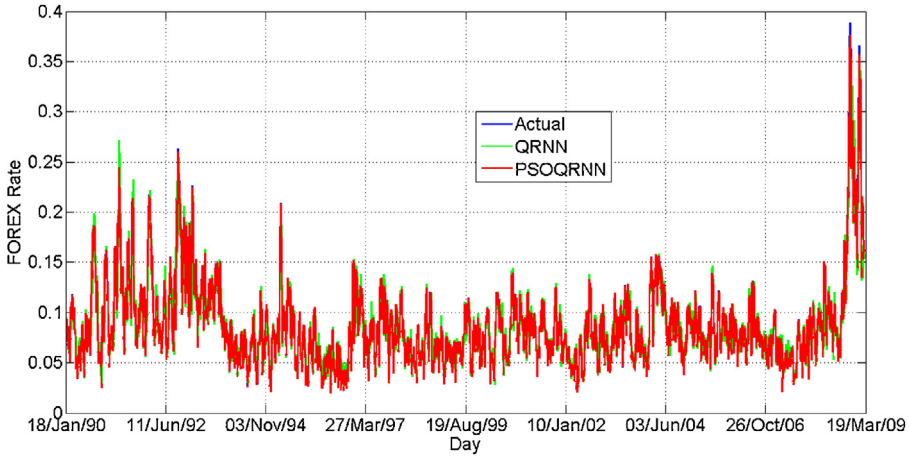

Fig. 6. Predictions of training set of USD–GBP volatilities.

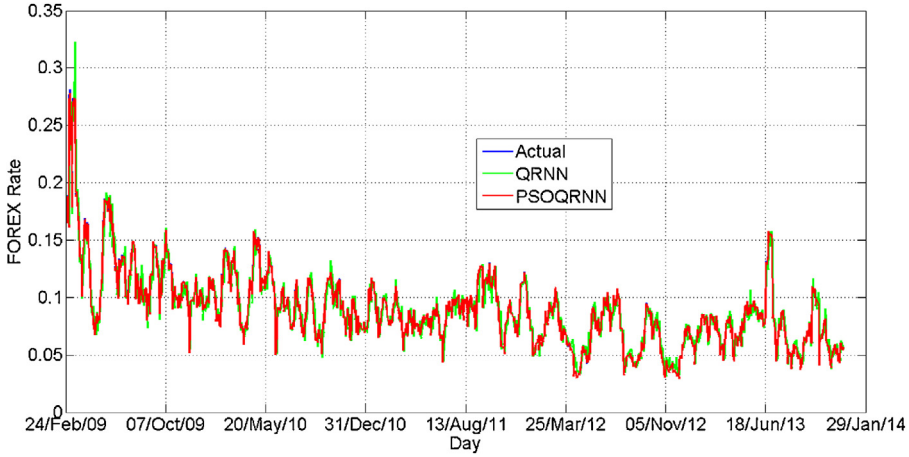

Fig. 7. Predictions of test set of USD–GBP volatilities.

Table 6
Results of volatility forecasting models for USD–EUR data.

Forecasting model	Training set			Test set		
	MSE	Dstat	Theil's U	MSE	Dstat	Theil's U
GARCH	0.000164	48.33	0.0631	0.000138	47.74	0.0630
MLP	0.000421	49.00	0.1028	0.000231	48.40	0.0812
GRNN	0.000135	52.56	0.0565	0.000127	52.25	0.0595
GMDH	0.000149	54.39	0.0593	0.000125	53.05	0.0591
RF	0.000175	51.83	0.0643	0.000143	53.84	0.0629
QRRF	0.0000002820	98.33	0.0026	0.0000001564	98.01	0.0021
QRNN	0.0000014361	93.87	0.0058	0.0000004030	95.88	0.0033
PSOQRNN	0.0000008911	97.80	0.0046	**0.0000000612**	**98.48**	**0.0013**

Table 7
Results of volatility forecasting models for USD–INR data.

Forecasting model	Training set			Test set		
	MSE	Dstat	Theil's U	MSE	Dstat	Theil's U
GARCH	0.000367	59.65	0.1423	0.000332	52.95	0.0960
MLP	0.000597	59.55	0.1801	0.000232	50.37	0.0760
GRNN	0.000267	63.00	0.1155	0.000196	53.44	0.0702
GMDH	0.000221	59.80	0.1046	0.000189	50.12	0.0683
RF	0.000315	57.72	0.1260	0.000262	49.54	0.0812
QRRF	0.00001680	98.56	0.0287	0.000001373	97.75	0.0058
QRNN	0.00000858	97.75	0.0204	0.000047722	95.42	0.0343
PSOQRNN	0.00003340	97.96	0.0410	**0.000000838**	**98.17**	**0.0045**

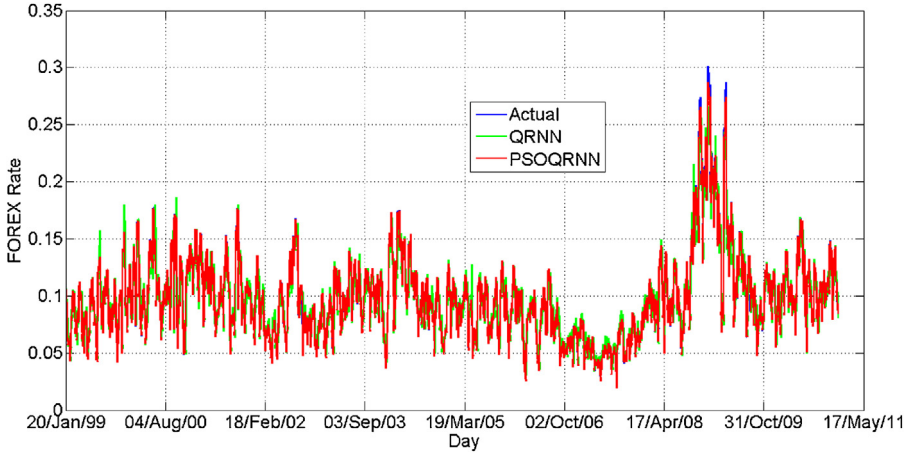

Fig. 8. Predictions of training set of USD–EUR volatilities.

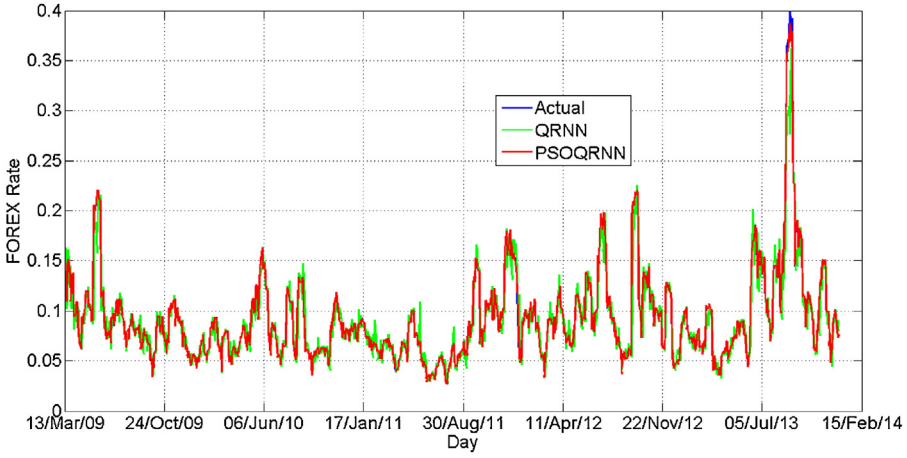

Fig. 9. Predictions of test set of USD–EUR volatilities.

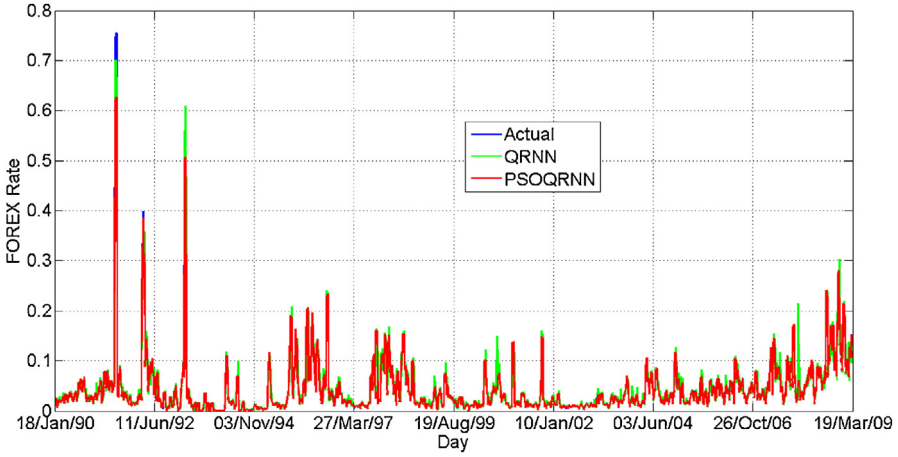

Fig. 10. Predictions of training set of USD–INR volatilities.

PSOQRNN yielded 98% less MSE than that of QRNN on Test set. Further, PSOQRNN yielded the highest Dstat too. These reveal that PSOQRNN yielded better predictions than QRNN in both training set and test set. Figs. 10 and 11 depict the predictions of volatilities of both training set and test set respectively. The minimum MSE obtained by the proposed PSOQRNN in 30 runs is also least when compared to the MSE obtained by other techniques. From Table 7, it is also evident that QR hybrids best predicted the volatility of USD–INR than GARCH, MLP, GRNN, GMDH, and RF.

6.5. Gold Price (USD)

Table 8 presents MSE, Dstat and Theil's U values obtained by using volatility forecasting models including GARCH, MLP, GRNN,

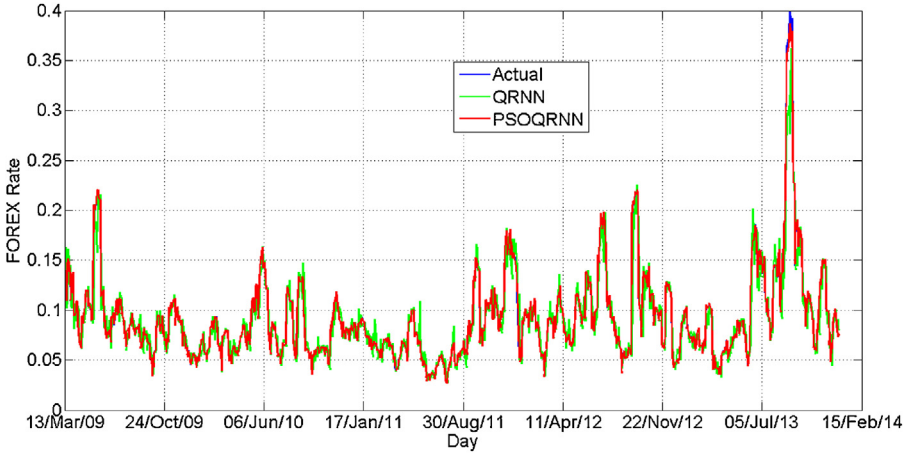

Fig. 11. Predictions of test set of USD–INR volatilities.

Table 8
Results of volatility forecasting models for Gold Price (USD) data.

Forecasting model	Training set			Test set		
	MSE	Dstat	Theil's U	MSE	Dstat	Theil's U
GARCH	0.000565	51.49	0.0810	0.001100	49.50	0.0799
MLP	0.001800	51.26	0.1357	0.003200	47.92	0.1358
GRNN	0.000451	51.96	0.0701	0.000918	49.63	0.0704
GMDH	0.000452	55.11	0.0698	0.000903	50.42	0.0697
RF	0.000498	52.23	0.0736	0.001100	51.08	0.0768
QRRF	0.0000086592	98.13	0.0097	0.000022427	96.64	0.0109
QRNN	0.0000218590	95.18	0.0154	0.000122900	93.75	0.0257
PSOQRNN	0.0000073600	97.01	0.0028	**0.000002050**	**98.51**	**0.0033**

GMDH, RF, QRRF, QRNN and PSOQRNN for both training set and test set of datasets of Gold Price (USD). For the Gold Price dataset (see Table 7), PSOQRNN yielded 98% less MSE than that of QRNN on Test set. Further, PSOQRNN yielded the highest Dstat and Theil's Inequality Coefficient too. These reveal that PSOQRNN yielded better predictions than QRNN in both training set and test set. The predictions of volatilities of both training set and test set are depicted in Figs. 12 and 13 respectively. The minimum MSE obtained by the proposed PSOQRNN in 30 runs is also least when compared to the MSE obtained by other techniques. From Table 8, it is also clear that QR hybrids best predicted the volatility of Gold Price than GARCH, MLP, GRNN, GMDH and RF.

6.6. Crude Oil Price (USD)

Table 9 presents MSE, Dstat and Theil's U values obtained by using volatility forecasting models including GARCH, MLP, GRNN, GMDH, RF, QRRF, QRNN and PSOQRNN for both training set and test set of datasets of Crude Oil Price (USD). For the Crude Oil Price dataset (see Table 8), PSOQRNN yielded 29% less MSE than that of QRNN. Further, PSOQRNN yielded the highest Dstat and Theil's Inequality Coefficient too. The minimum MSE obtained by the proposed PSOQRNN in 30 runs is least when compared to the MSE obtained by other techniques. The predictions of volatilities of both training set and test set are depicted in Figs. 14 and 15 respectively.

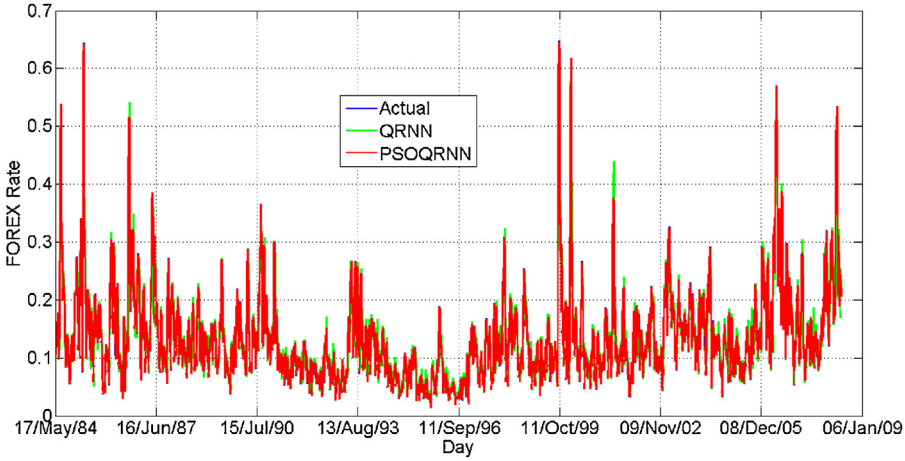

Fig. 12. Predictions of training set of Gold Price (USD) volatilities.

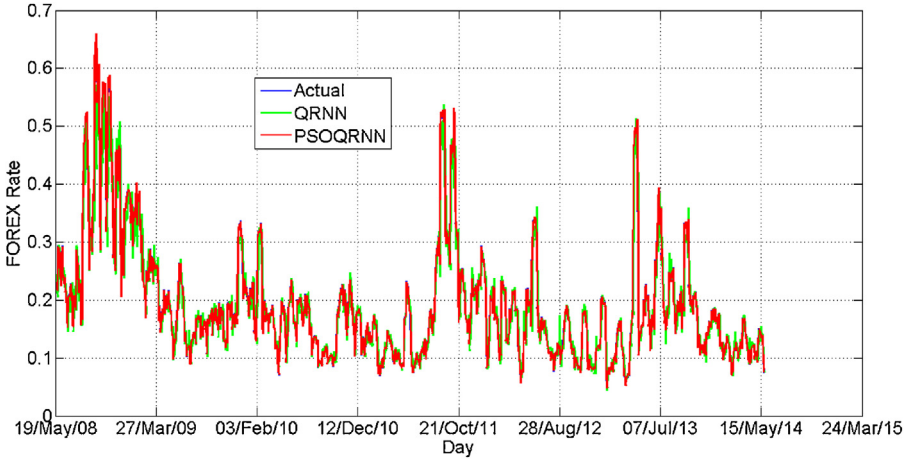

Fig. 13. Predictions of test set of Gold Price (USD) volatilities.

Table 9
Results of volatility forecasting models for Crude Oil Price (USD) data.

Forecasting model	Training set			**Test set**		
	MSE	Dstat	Theil's U	MSE	Dstat	Theil's U
GARCH	0.003527	48.94	0.0832	0.002351	49.05	0.0776
MLP	0.005400	49.09	0.1013	0.002700	51.27	0.0807
GRNN	0.002100	51.21	0.0625	0.001800	49.74	0.0644
GMDH	0.002700	51.03	0.0703	0.001800	51.86	0.0654
RF	0.003500	52.57	0.0802	0.002000	56.30	0.0684
QRRF	0.000040148	96.47	0.0271	0.00001488	95.04	0.0187
QRNN	0.000010479	95.99	0.0044	0.00000518	96.20	0.0035
PSOQRNN	0.000004068	96.83	0.0027	**0.00000369**	**97.69**	**0.0017**

From Table 8, it is also clear that QR hybrids best predicted the volatility of Crude Oil Price than GARCH, MLP, GRNN, GMDH, and RF.

6.7. S & P 500 Stock Index

Table 10 presents MSE, Dstat and Theil's U values obtained by using volatility forecasting models including GARCH, MLP, GRNN, GMDH, RF, QRRF, QRNN and PSOQRNN for both training set and test set of dataset of S&P 500 Stock Index. For the S&P Stock Index dataset, PSOQRNN yielded 12% less MSE than that of QRNN on Test set. Further, PSOQRNN yielded the highest Dstat and Theil's Inequality Coefficient too. These reveal that PSOQRNN yielded better predictions than QRNN in both training set and test set. The predictions of volatilities of both training set and test set are depicted in Figs. 16 and 17 respectively. The minimum MSE obtained by the proposed PSOQRNN in 30 runs is also least when compared to the MSE obtained by other techniques. From Table 10, it is also clear that QR hybrids best predicted the volatility of S&P 500 Stock Index than GARCH, MLP, GRNN, GMDH, and RF.

6.8. NSE India Stock Index

Table 11 presents MSE, Dstat and Theil's U values obtained by using volatility forecasting models including GARCH, MLP, GRNN, GMDH, RF, QRRF, QRNN and PSOQRNN for both of the training set and the test set of dataset of NSE India Stock Index. For the NSE India Stock Index dataset, PSOQRNN yielded 84% less MSE than that

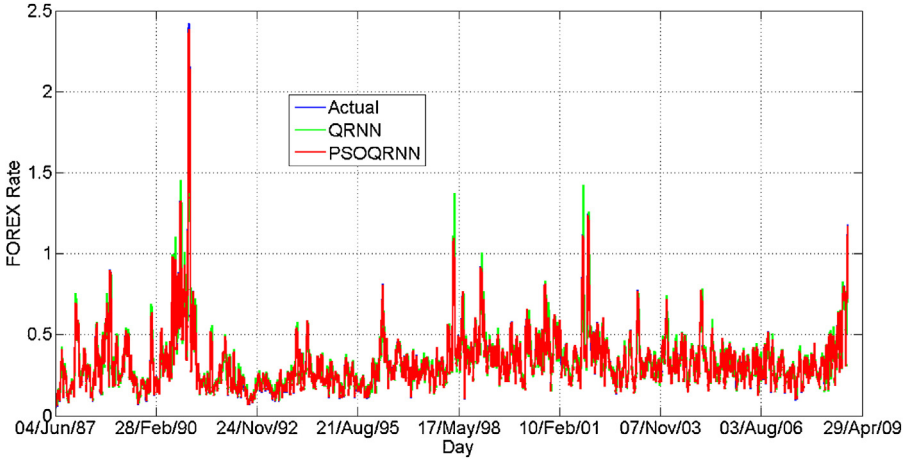

Fig. 14. Predictions of training set of Crude Oil Price (USD) volatilities.

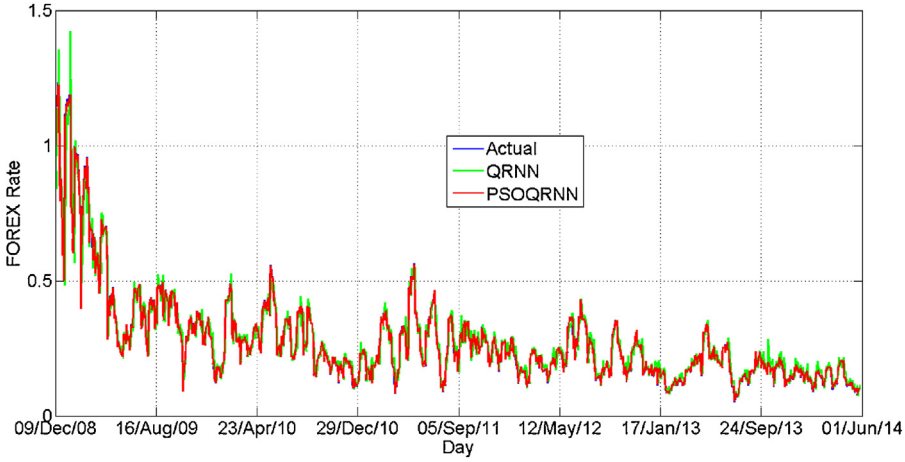

Fig. 15. Predictions of test set of Crude Oil Price (USD) volatilities.

Table 10
Results of volatility forecasting models for S&P 500 Stock Index data.

Forecasting model	Training set			Test set		
	MSE	Dstat	Theil's U	MSE	Dstat	Theil's U
GARCH	0.000975	48.18	0.0897	0.000661	49.10	0.0806
MLP	0.002100	48.51	0.1303	0.001400	50.65	0.1141
GRNN	0.000471	50.33	0.0591	0.000374	51.64	0.0575
GMDH	0.000550	50.87	0.0638	0.000365	51.71	0.0567
RF	0.000746	52.65	0.0747	0.000434	54.68	0.0616
QRRF	0.000062617	98.21	0.0215	0.000001676	97.49	0.0038
QRNN	0.000033184	94.59	0.0156	0.000000928	95.11	0.0029
PSOQRNN	0.000019405	97.55	0.0119	**0.000000815**	**96.83**	**0.0023**

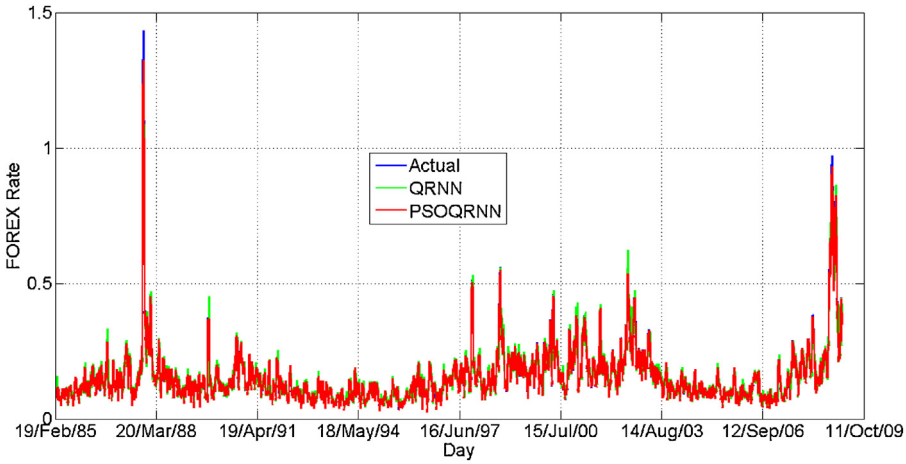

Fig. 16. Predictions of training set of S&P 500 Stock Index volatilities.

Table 11
Results of volatility forecasting models for NSE India Stock Index data.

Forecasting model	Training set			Test set		
	MSE	Dstat	Theil's U	MSE	Dstat	Theil's U
GARCH	0.063788	52.50	0.2564	0.039344	49.11	0.2543
MLP	0.018600	46.30	0.1183	0.005600	42.19	0.0788
GRNN	0.006600	53.33	0.0663	0.003100	49.29	0.0573
GMDH	0.006700	51.61	0.0669	0.003100	45.50	0.0576
RF	0.007900	53.20	0.0727	0.003800	50.82	0.0633
QRRF	0.00053002	96.26	0.0188	0.00027636	95.39	0.0172
QRNN	0.00014159	94.86	0.0097	0.00003378	95.39	0.0060
PSOQRNN	0.00014835	96.53	0.0099	**0.00000537**	**97.99**	**0.0024**

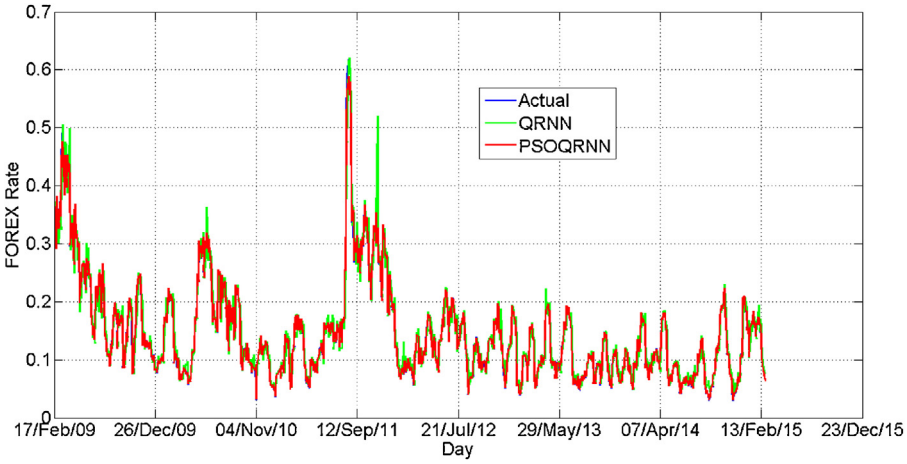

Fig. 17. Predictions of test set of S&P 500 Stock Index volatilities.

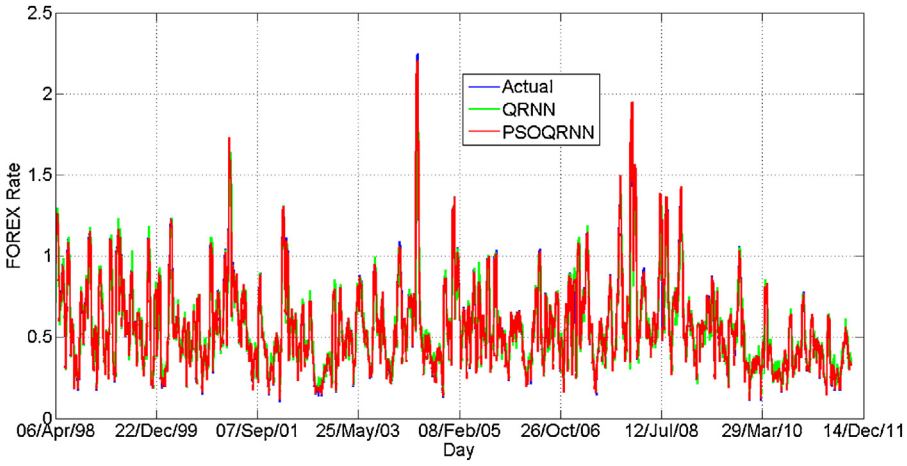

Fig. 18. Predictions of training set of NSE India Stock Index volatilities.

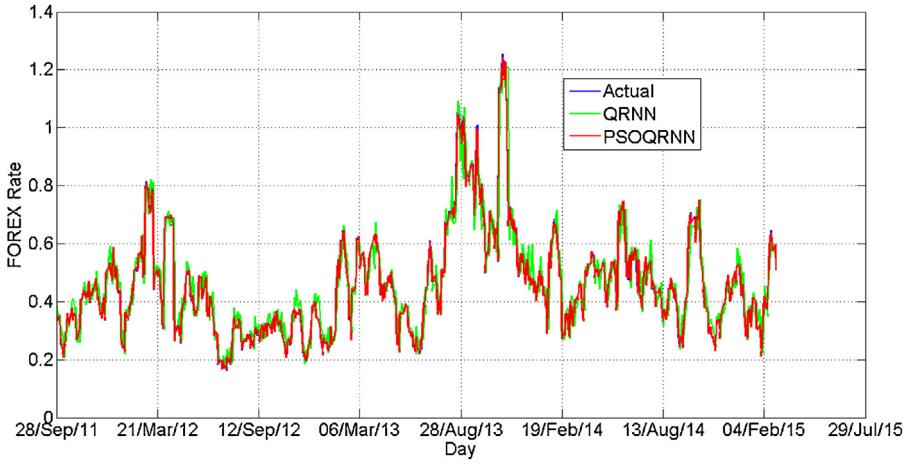

Fig. 19. Predictions of test set of NSE India Stock Index volatilities.

of QRNN on Test set. Further, PSOQRNN yielded the highest Dstat and Theil's Inequality Coefficient too. These reveal that PSOQRNN yielded better predictions than QRNN in both training set and test set. The predictions of volatilities of both training set and test set are depicted in Figs. 18 and 19 respectively. The minimum MSE obtained by the proposed PSOQRNN in 30 runs is least when compared to the MSE obtained by other techniques. From Table 11, it is also clear that QR hybrids best predicted the volatility of NSE India Stock Index than GARCH, MLP, GRNN, GMDH and RF.

6.9. Discussion

Since QR has the capability of characterizing and modeling the variability of the dependent variable across all the quantiles,

Table 12
Results of Diebold–Mariano (DM) test on test sets of all datasets.

Forecasting model	USD–JPY	USD–GBP	USD–EUR	USD–INR	Gold Price	Crude Oil Price	NSE India Stock Index	**S&P 500 Stock Index**
PSOQRNN Vs								
GARCH	2.440949	1.238817	1.379453	3.31054	9.371722	6.731015	6.585208	26.42448
MLP	6.983993	2.917819	2.314754	2.31974	12.86086	7.186184	10.89401	7.575923
GRNN	2.141267	1.015665	1.268521	1.95331	7.733054	5.567485	3.679559	5.727691
GMDH	2.164064	1.019099	1.253273	1.88248	7.806101	5.418079	3.595849	5.626264
RF	2.490336	1.179588	1.428840	2.61283	9.295843	6.023462	4.277927	6.460665
QRRF	0.000695	0.0010651	0.0009519	0.00535	0.2037165	1.357820	0.0457576	1.483677
QRNN	0.002296	0.0065773	0.0034160	0.46864	1.208080	0.014929	0.0532363	0.1353052

covering the whole distribution of a dependent variable, QR when used in conjunction with RF (resulting in the hybrid, QRRF) outperformed the stand-alone RF. The same phenomenon applies to the hybrid QRNN also. These two observations can be easily evidenced by the results presented in Tables 3–10, wherein QRRF ($\tau = 0.5$) and QRNN ($\tau = 0.5$) outperformed RF and Neural Network respectively in terms of MSE, Dstat and Theil's U for all datasets.

However, in particular, QRRF outperformed QRNN on five datasets including USD versus JPY, GBP, EUR, INR and Gold Price (USD) because of the following reasons:

1. QRRF chose an appropriate number of random covariates to find the best split. Once the best split is found, the predictions obtained from it turned to be better predictions than those predictions obtained from the other forecasting models.
2. The bagging procedure in QRRF reduced the variance in predictions so that QRRF could obtain better predictions.
3. QRRFs considered random feature subsets of data at different quantiles, constructed trees for each subset over a bootstrap sample and ensembled the obtained results. The ensembling procedure in QRRF helped it in yielding better predictions than other forecasting models. The last mentioned two features are missing in MLP and other ANNs.

It is also important to note that QRNN outperformed QRRF on three datasets including Crude Oil Price (USD), S&P 500 Stock Index and NSE India Stock Index. The QRRF accepts a range for each observation will be in (with high probability). It will be less accurate when there is the wider range for a new instance [10]. Therefore, based on this observation, QRRF probably could not yield better predictions compared to QRNN as there may be the wider range for the new instance of three datasets.

The Back Propagation algorithm in QRNN implements a gradient descent search through the space of possible network weights. Consequently, these weights converge to a local minimum but not to the global optimum [52]. The PSO, being an evolutionary algorithm, has a higher chance of obtaining the global optimal solution. So, the PSO is utilized for training the QRNN. Therefore, PSOQRNN could yield better predictions.

Finally, the above forecasting accuracy measures including MSE, Dstat, and Theil's U do not formally test whether one method is statistically significantly different from another method or not. For this purpose, there is one popular test proposed by Diebold and Mariano [68]. We employed this test statistic that is implemented as a part of 'forecast' package from archives of R (https://cran.r-project.org/web/packages/forecast/) to check whether PSOQRNN is performing statistically significantly different from other volatility forecasting models on average or not. Table 12 presents the results of the Diebold–Mariano test on eight datasets. If the test statistic is less than or equal to 0.05, then the corresponding model is performing equally accurate with PSOQRNN. From the table, it is clear that PSOQRNN is superior to all models on two datasets, superior to all models except QRNN on three datasets, superior to all models except QRRF on four datasets and superior to all models except QRRF and QRNN in all cases.

7. Conclusions

This paper proposed a novel PSO trained QRNN, called PSOQRNN, to forecast volatility from a financial time series. It is observed that the proposed PSOQRNN yielded statistically significant results compared to other popular volatility forecasting models such as GARCH, MLP, GRNN, GMDH, RF, QRRF, and QRNN on eight financial datasets in terms of MSE. It also performed well in terms of other important measures Dstat and Theil's U Inequality Coefficient. The spectacular performance of PSOQRNN is caused by the presence of PSO, which yielded global optimal weights while training PSOQRNN. The results are encouraging, and we suggest its further use in volatility forecasting in similar other financial and non-financial data.

References

[1] S.-H. Poon, A Practical Guide to Forecasting Financial Market Volatility, John Wiley & Sons Ltd., 2005.
[2] S.-H. Poon, C. Granger, Practical issues in forecasting volatility types of volatility models, Financ. Anal. J. 61 (2005) 45–56.
[3] T. Bollerslev, Generalized autoregressive conditional heteroskedasticity, J. Economet. 31 (1986) 307–327.
[4] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, 1986, pp. 318–362.
[5] D. Specht, A general regression neural network, IEEE Trans. Neural Netw. 2 (1991) 568–576.
[6] A. Ivakhnenko, The GMDH: a rival of stochastic approximation, Sov. Autom. Control 3 (1968).
[7] G. Zhang, B. Eddy Patuwo, M.Y. Hu, Forecasting with artificial neural networks: the state of the art, Int. J. Forecast. 14 (1998) 35–62.
[8] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[9] S.-K. Oh, D.-W. Kim, B.-J. Park, H.-S. Hwang, Advanced polynomial neural networks architecture with new adaptive nodes, Trans. Control Autom. Syst. Eng. 3 (2001) 43–50.
[10] N. Meinshausen, Quantile regression forests, J. Mach. Learn. Res. 7 (2006) 983–999.
[11] V. Ravi, A. Sharma, Support vector-quantile regression random forest hybrid for regression problems, in: MIWAI 2014, Bangalore, India, LNAI 8875, December 9–10, 2014, pp. 149–160.
[12] J.W. Taylor, A quantile regression approach to estimating the distribution of multiperiod returns, J. Deriv. 7 (1999) 64–78.
[13] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, Perth, Australia, 27 November to 1 December, 1995, pp. 1942–1948.
[14] P.H. Franses, M. McAleer, Financial volatility: an introduction, J. Appl. Economet. 17 (2002) 419–424.
[15] T.G. Andersen, T. Bollerslev, P.F. Christoffersen, F.X. Diebold, Volatility and correlation forecasting, Handb. Econ. Forecast. 1 (2006) 777–878.
[16] J.L. Knight, S.S. Satchell, Forecasting Volatility in the Financial Markets, 3rd ed., Butterworth-Heinemann, 2007.
[17] R.F. Engle, Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation, Econometrica 50 (1982) 987–1007.
[18] T. Bollerslev, R.Y. Chou, K.F. Kroner, ARCH modeling in finance: a review of the theory and empirical evidence, J. Economet. 52 (1992) 5–59.
[19] J. Noh, R.F. Engle, A. Kane, Forecasting volatility and option prices of the S&P 500 index, J. Deriv. 2 (1994) 17–30.
[20] J. Vilasuso, Forecasting exchange rate volatility, Econ. Lett. 76 (2002) 59–64.
[21] R.T. Baillie, T. Bollerslev, H.O. Mikkelsen, Fractionally integrated generalized autoregressive conditional heteroskedasticity, J. Economet. 74 (1996) 3–30.

[22] J.-R. Chang, L.-Y. Wei, C.-H. Cheng, A hybrid ANFIS model based on AR and volatility for TAIEX forecasting, Appl. Soft Comput. 11 (2011) 1388–1395.
[23] P. Agnolucci, Volatility in crude oil futures: a comparison of the predictive ability of GARCH and implied volatility models, Energy Econ. 31 (2009) 316–321.
[24] R. Donaldson, M. Kamstra, An artificial neural network-GARCH model for international stock return volatility, J. Empir. Finance 4 (1997) 17–46.
[25] F. Gonzalez Miranda, N. Burgess, Modelling market volatilities: the neural network perspective, Eur. J. Finance 3 (1997) 137–157.
[26] S.A. Hamid, Z. Iqbal, Using neural networks for forecasting volatility of S&P 500 Index futures prices, J. Bus. Res. 57 (2004) 1116–1125.
[27] J.R. Aragonés, C. Blanco, P.G. Estévez, Neural network volatility forecasts, Intell. Syst. Acc. Finance Manage. 15 (2007) 107–121.
[28] M. Mohsen, B. Nafiseh, A. Mehdi, M. Mohsen, Forecasting volatility of crude oil price using the GMDH neural network, Q. Energy Econ. Rev. 7 (2010) 89–112.
[29] X.-f. Zhuang, L.-w. Chan, Volatility forecasts in financial time series with HMM-GARCH models, J. Financ. Strateg. Decis. 3177 (2004) 807–812.
[30] T. Hyup Roh, Forecasting the volatility of stock price index, Expert Syst. Appl. 33 (2007) 916–922.
[31] C.-H. Tseng, S.-T. Cheng, Y.-H. Wang, J.-T. Peng, Artificial neural network model of the hybrid EGARCH volatility of the Taiwan stock index option prices, Phys. A: Stat. Mech. Appl. 387 (2008) 3192–3200.
[32] C.-H. Tseng, S.-T. Cheng, Y.-H. Wang, New hybrid methodology for stock volatility prediction, Expert Syst. Appl. 36 (2009) 1833–1839.
[33] Y.-H. Wang, Nonlinear neural network forecasting model for stock index option price: hybrid GJR-GARCH approach, Expert Syst. Appl. 36 (2009) 564–570.
[34] M. Bildirici, Ö.Ö. Ersin, Improving forecasts of GARCH family models with the artificial neural networks: An application to the daily returns in Istanbul Stock Exchange, Expert Syst. Appl. 36 (2009) 7355–7362.
[35] J.-C. Hung, Forecasting volatility of stock market using adaptive Fuzzy-GARCH model, in: 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology, ICCIT09, IEEE, Seoul, Korea, 24–26 November, 2009, pp. 583–587.
[36] E. Hajizadeh, A. Seifi, M. Fazel Zarandi, I. Turksen, A hybrid modeling approach for forecasting the volatility of S&P 500 index return, Expert Syst. Appl. 39 (2012) 431–436.
[27] S.A. Monfared, D. Enke, Volatility forecasting using a hybrid GJR-GARCH neural network model, in: Complex Adaptive Systems, Procedia Computer Science, vol. 36, Philadelphia, PA, November 3–5, 2014, pp. 246–253.
[38] A. Komijani, E. Naderi, N. Gandali Alikhani, A hybrid approach for forecasting of oil prices volatility, OPEC Energy Rev. 38 (2014) 323–340.
[39] S. Choudhury, S. Ghosh, A. Bhattacharya, K.J. Fernandes, M.K. Tiwari, A real time clustering and SVM based price-volatility prediction for optimal trading strategy, Neurocomputing 131 (2014) 419–426.
[40] R. Rosa, L. Maciel, F. Gomide, R. Ballini, Evolving hybrid neural fuzzy network for realized volatility forecasting with jumps, in: 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), London, UK, 27–28 March, 2014, pp. 481–488.
[41] C. Narendra Babu, B. Eswara Reddy, Prediction of selected Indian stock using a partitioning-interpolation based ARIMA-GARCH model, Appl. Comput. Inf. 11 (2015) 130–143.
[42] W. Kristjanpoller, M.C. Minutolo, Gold price volatility: a forecasting approach using the artificial neural network-GARCH model, Expert Syst. Appl. 42 (2015) 7245–7251.
[43] R. Dash, P. Dash, R. Bisoi, A differential harmony search based hybrid interval type2 fuzzy EGARCH model for stock market volatility prediction, Int. J. Approx. Reason. 59 (2015) 81–104.
[44] A.Y. Huang, S.-P. Peng, F. Li, C.-J. Ke, Volatility forecasting of exchange rate by quantile regression, Int. Rev. Econ. Finance 20 (2011) 591–606.
[45] S.J. Farlow, Self-Organizing Methods in Modeling: GMDH Type Algorithms, Illustrated, volume 54 of Statistics: A Series of Textbooks and Monographs Edition, CRC Press, 1984.
[46] S. Ketabchia, H. Ghanadzadehc, A. Ghanadzadehb, S. Fallahia, M. Ganjia, Estimation of VLE of binary systems (tert-butanol+2-ethyl-1-hexanol) and (n-butanol+2-ethyl-1-hexanol) using GMDH-type neural network, J. Chem. Thermodyn. 42 (2010) 1352–1355.
[47] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Belmont, Wadsworth, 1984.
[48] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[49] R. Koenker, G. Bassett, Regression quantiles, Econometrica 46 (1978) 33–50.
[50] A.J. Cannon, Quantile regression neural networks: implementation in R and application to precipitation downscaling, Comput. Geosci. 37 (2011) 1277–1284.
[51] M. Saerens, Building cost functions minimizing to some summary statistics, IEEE Trans. Neural Netw. 11 (2000) 1263–1271.
[52] T.M. Mitchell, Machine Learning, McGraw-Hill, 1997.
[53] R. Mendes, P. Cortez, M. Rocha, J. Neves, Particle swarms for feedforward neural network training, in: 2002 International Joint Conference on Neural Networks (IJCNN02), vol. 6, Honolulu, Hawaii, 12–17 May, 2002, pp. 1895–1899.
[54] G.K. Jha, P. Thulasiraman, R.K. Thulasiram, PSO based neural network for time series forecasting, in: 2009 International Joint Conference on Neural Networks (IJCNN2009), IEEE, Atlanta, Georgia, USA, 14–19 June, 2009, pp. 1422–1427.
[55] R. Adhikari, R.K. Agrawal, Effectiveness of PSO based neural network for seasonal time series forecasting, in: 5th Indian International Conference on Artificial Intelligence (IICAI 2011), Tumkur, India, 14–16 December, 2011, pp. 231–244.
[56] M.S. Innocente, J. Sienz, Particle swarm optimization with inertia weight and constriction factor, in: The 2nd International Conference on Swarm Intelligence (ICSI'2011), Chongqing, China, 12–15 June, 2011, pp. 1–11.
[57] M. AlRashidi, M. El-Hawary, A survey of particle swarm optimization applications in electric power systems, IEEE Trans. Evol. Comput. 13 (2009) 913–918.
[58] R. Eberhart, Y. Shi, Guest editorial. Special issue on particle swarm optimization, IEEE Trans. Evol. Comput. 8 (2004) 201–203.
[59] S. Makridakis, M. Hibon, Evaluating accuracy (or error) measures, Insead (1995) 1–41.
[60] J. Yao, C.L. Tan, A case study on using neural networks to perform technical forecasting of forex, Neurocomputing 34 (2000) 79–98.
[61] H. Theil, Applied Economic Forecasting, North-Holland Pub. Co., Amsterdam, 1966.
[62] T. Masters, Practical Neural Network Recipies in C++, Academic Press, Inc., London, 1993.
[63] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: IEEE World Congress on Computational Intelligence, Anchorage, Alaska, 4–9 May, 1998, pp. 69–73.
[64] J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, in: Congress on Evolutionary Computation-CEC99, IEEE, Washington DC, USA, 6–9 July, 1999, pp. 1931–1938.
[65] S. Das, A. Abraham, A. Konar, Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives, Stud. Comput. Intell. 116 (2008) 1–38.
[66] D. Pradeepkumar, V. Ravi, FOREX rate prediction using chaos, neural network and particle swarm optimization, in: 5th International Conference on Swarm Intelligence, ICSI 2014, volume LNCS 8795, Springer International Publishing, Hefei, China, 17–20 October, 2014, pp. 363–375.
[67] G. Venter, J. Sobieszczanski-Sobieski, Particle swarm optimization, in: 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, American Institute of Aeronautics and Astronautics, Denver, Colorado, 22–25 April, 2002, pp. 1–9.
[68] F.X. Diebold, R.S. Mariano, Comparing predictive accuracy, J. Bus. Econ. Stat. 13 (1995) 253–263.