Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 200 (2008) 41–57
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

Rainfall forecasting by technological machine learning models

Wei-Chiang Hong

Department of Information Management, Oriental Institute of Technology No. 58, Section 2, Sichuan Road, Panchiao, Taipei 220, Taiwan

Abstract

Accurate forecasting of rainfall has been one of the most important issues in hydrological research. Due to rainfall forecasting involves a rather complex nonlinear data pattern; there are lots of novel forecasting approaches to improve the forecasting accuracy. Recurrent artificial neural networks (RNNS) have played a crucial role in forecasting rainfall data. Meanwhile, support vector machines (SVMs) have been successfully employed to solve nonlinear regression and time series problems. This investigation elucidates the feasibility of hybrid model of RNNs and SVMs, namely RSVR, to forecast rainfall depth values. Moreover, chaotic particle swarm optimization algorithm (CPSO) is employed to choose the parameters of a SVR model. Subsequently, example of rainfall values during typhoon periods from Northern Taiwan is used to illustrate the proposed RSVRCPSO model. The empirical results reveal that the proposed model yields well forecasting performance, RSVRCPSO model provides a promising alternative for forecasting rainfall values.

Keywords: Rainfall forecasting; Support vector regression (SVR); Chaotic particle swarm optimization algorithm (CPSO); Recurrent SVR; Machine learning

1. Introduction

In Taiwan, due to the southwest monsoon, monsoon trough and tropical cyclones, it is typically with heavy raining in the summer. Therefore, unanticipated flash floods with high peak discharges represent the most destructive natural hazard to threaten human life and properties. Accurate forecasts of rainfall information (including the spatial and temporal distribution of rainfalls) are the best approach to avoid life losing and economic loses [1].

1.1. Traditional rainfall forecasting approaches

Generally, there are two major approaches to forecast rainfall, conceptual (physical) modeling and system theoretical modeling [2,3]. Conceptual (physical) approaches are designed to approximate within the physical mechanisms which govern the hydrologic processes (simplified as physical laws), and usually based on the characteristics and knowledge of a specific catchment (watershed). Conceptual approaches are widely applied

E-mail address: samuelhong@ieee.org

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.10.046

in hydrological forecasting, and particularly for non-stationary hydrological data and continuous variables data tracking required [4,5]. This approach for rainfall forecasting may not be feasible due to significant calibration data (explanatory variables) to reasoning rainfall is not easily to be collected [6], and the volume of rainfall calculations require sophisticated mathematical tools [7].

System theoretical approaches are employed mapping models to formulate the relationships between inputs and outputs without consideration of the physical structure processes. Developed by Box and Jenkins [8], the ARMAX (autoregressive moving average with exogenous inputs) models have been one of the most popular approaches in time series forecasting. However, a fundamental limitation for time-series forecasting models is their inability to predict changes that are not based on the past data, particularly for the nonlinearity of rainfall related variables.

1.2. Artificial neural networks in rainfall forecasting

Recently, due to the significant progress in the fields of pattern recognition methodology, artificial neural network (ANN) is possible to be employed to forecast rainfall. ANN is based on a model of emulating the processing of human neurological system to find out related spatial and temporal characteristics from the historical rainfall patterns (especially for nonlinear and dynamic evolutions). Due to without understanding the physical laws and any assumptions of traditional statistical approaches required, ANN is widely applied to solve hydrological problems including rainfall forecasting, French et al. [9] applied ANN to construct rainfall simulation model and provide accurate rainfall forecasting information, temporal and spatial distributions. Luk et al. [10] employed ANN to forecast the short-term rainfall for an urban catchment. Empirical results indicated that the ANN model with lower lag outperformed in terms of forecasting accurate index. Valverde Ramírez et al. [11] proposed a feed-forward neural network with resilient propagation learning algorithm to forecast daily rainfall in São Paulo region, Brazil. The numerical results indicated that proposed model is superior to a multiple linear regression model in terms of forecasting accuracy indices. Pan and Wang [12] employed state space neural network (SSNN) to implement short term rainfall-runoff forecasting in Taiwan's Wu-Tu watershed. The SSNN is used to strengthen the linkage between weights and network physical concepts, and to interchange the network states to evolve the networks into a time-variant model. Empirical performances, from 1966 to 1997, indicate that SSNN is suitable for hydrological forecasts. Lin and Chen [13] applied the radial basis function network (RBFN), which is more convenient to determine the number of hidden layer neurons, to conduct rainfall-runoff forecasting in Taiwan's Fei-Tsui reservoir watershed. In addition, to avoid over-fitting, forecasting accurate indices are employed as stopping criterion. Empirical results showed that RBFN is superior and feasible to explain the relationships between rainfall and runoff. Chiang et al. [14] applied two kinds of back propagation neural networks, traditional conjugate gradient (CG) algorithm and real-time recurrent learning (RTRL) algorithm, to forecast the rainfall-runoff in Taiwan's Lan-Yang River. CG is used for static network; in contrast, RTRL is used for dynamic network. The investigation indicated that CG could produce more accurate performance while sufficient training hydrologic data is available. Then, RTRL outperforms CG due to its ability to update dynamic data continuously. Other useful hydrologic applications of ANN model are also conducted recently, such as monthly runoff forecasting [15], stream flow forecasting [16] and wastewater inflow prediction [17].

1.3. Applications of support vector regression forecasting approaches

SVMs were also originally developed to solve pattern recognition and classification problems. With the introduction of Vapnik's ε-insensitive loss function, SVMs have been extended to solve nonlinear regression (also known as SVR) estimation problems and have been successfully applied to solve forecasting problems in many fields. Tay and Cao [18] applied SVMs to forecast financial time series. Their numerical results demonstrated that SVMs are superior to a multi-layer back-propagation neural network in financial time series forecasting. Wang et al. [19] used SVMs to predict air quality. They reported that SVMs outperform conventional radial basis function networks. Cao and Gu [20] presented a dynamic SVM model for solving non-stationary time series problems. Their experimental results indicated that DSVMs outperform standard SVMs in forecasting non-stationary time series. In the same year, Tay and Cao [21] developed a C-ascending SVM to model

non-stationary financial time series. Their experimental results showed that C-ascending SVMs with sample data of actual orders consistently outperform standard SVMs. Cao [22] applied SVM experts in forecasting time series. Generalized expert SVMs have a two-stage neural network architecture. The numerical results demonstrated that SVM experts can achieve better generalization than single SVM models. Mohandes et al. [23] applied SVMs to predict wind speed. Their experimental results indicated that the SVM model outperformed multilayer perceptron neural networks as measured by root mean square errors. Pai and Lin [24] used SVMs to forecast production values of the machinery industry in Taiwan. They reported that SVMs performed better than the seasonal ARIMA model and the general regression neural network model. Pai and Lin [25] proposed a hybrid model with the strength of an ARIMA model and the SVM model in forecasting the stock prices. By using ten stocks to examine the performance, numerical results show that the proposed hybrid model provided more accurate forecasting results than the ARIMA model and random walk model.

1.4. Recurrent neural networks in time series forecasting

For a feed-forward neural network, links may be established within layers of a neural network, these types of networks are so-called recurrent neural networks (RNNs). The main concept on which RNNs is based, is that every unit is considered as an output of the network, and the provision of adjusted information as input in a training process [26]. RNNs are extensively applied in time series forecasting. Jordan [27] proposed a recurrent neural network model (Fig. 1) for controlling robots. Elman [28] developed a recurrent neural network model (Fig. 2) to solve linguistics problems. Williams and Zipser [29] presented a recurrent network model (Fig. 3) to solve nonlinear adaptive filtering and pattern recognition problems. These three models mentioned all consist of multi-layer perceptron (MLP) with a hidden layer. Jordan networks have a feedback loop from the output layer with past values to an additional input, namely "context layer". Then, output values from the context layer are fed back into the hidden layer. Elman networks have a feedback loop from the hidden layer to the context layer. In Williams and Zipser networks, nodes in the hidden layer are fully connected to each other. Both Jordan and Elman networks include an additional information source from the output layer or the hidden layer. Hence, these models use mainly past information to capture detailed information. Williams and Zipser networks take much more information from the hidden layer and back into themselves. Therefore, Williams and Zipser networks are sensitive when models are implemented [30]. Jordan networks and Elman networks are suited to time series forecasting [31,32]. In this investigation, the Jordan network is employed as a base to construct the recurrent SVR models.

The selection of three parameters (σ, C, and ε) in a SVR model heavily influences the forecasting accuracy. Due to a lack of structured ways in determining three free parameters in SVR, the chaotic particle swarm optimization (CPSO) is applied to determine the values of three parameters in SVR. On the other hand, recurrent

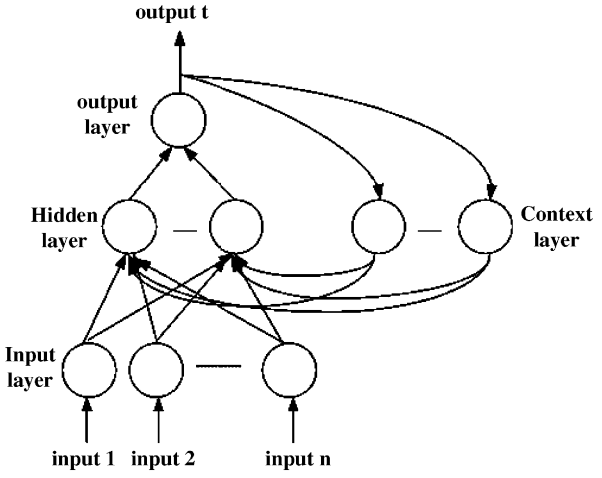

Fig. 1. Jordan networks [27].

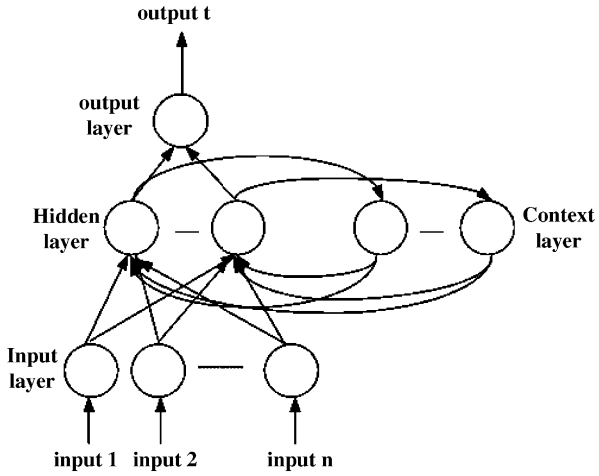

Fig. 2. Elman networks [28].

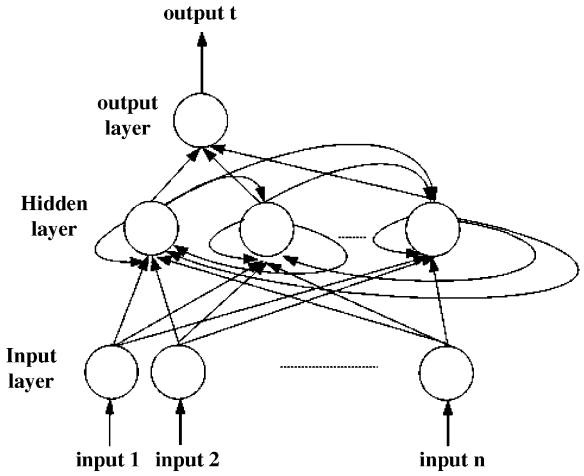

Fig. 3. Williams and Zipser networks [29].

neural networks are, in general, trained by back-propagation algorithms. In this investigation, Support vector regression with chaotic particle swarm optimization (SVRCPSO) is used to determine the weights between nodes. Then, the proposed Recurrent support vector regression with chaotic particle swarm optimization (RSVRCPSO) model is applied to forecast rainfall of Wu-Tu Watershed in northern Taiwan. The rest of the paper is organized as follows. Section 2 introduces the basic concept of RSVRCPSO. Section 3 illustrates a numerical example (Wu-Tu Watershed) that reveals the forecasting performance of the proposed models. Conclusions are finally made in Section 4.

2. Recurrent support vector regression with chaotic particle swarm optimization (RSVRCPSO)

2.1. Support vector regression

The basic concept of the SVR is to map nonlinearly the original data x into a higher dimensional feature space. Hence, given a set of data $G = \{(x_i, a_i)\}_{i=1}^{N}$ (where x_i is the input vector; a_i is the actual value, and N is the total number of data patterns), the SVR function is

$$f = g(x) = w\phi(x_i) + b, \tag{1}$$

where $\phi(x_i)$ is the feature of inputs, and both w and b are coefficients. The coefficients (w and b) are estimated by minimizing the following regularized risk function:

$$R(f) = C\frac{1}{N}\sum_{i=1}^{N} L_\varepsilon(a_i, f_i) + \frac{1}{2}\|w\|^2, \tag{2}$$

where

$$L_\varepsilon(a_i, f_i) = \begin{cases} 0 & \text{if } |a_i - f_i| \leqslant \varepsilon, \\ |a_i - f_i| - \varepsilon & \text{otherwise,} \end{cases} \tag{3}$$

and C and ε are prescribed parameters. In Eq. (2), $L_\varepsilon(a_i,f_i)$ is called the ε-insensitive loss function. The loss equals zero if the forecasted value is within the ε-tube (Eq. (3) and Fig. 4). The second term, $\frac{1}{2}\|w\|^2$, measures the flatness of the function. Therefore, C is considered to specify the trade-off between the empirical risk and the model flatness. Both C and ε are user-determined parameters. Two positive slack variables ξ and ξ^*, which represent the distance from actual values to the corresponding boundary values of ε-tube (Fig. 4), are introduced. Then, Eq. (2) is transformed into the following constrained form:

Minimize

$$R(w, \xi, \xi^*) = \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{N}(\xi_i + \xi_i^*)\right), \tag{4}$$

with the constraints

$$\begin{aligned} &w\phi(x_i) + b - a_i \leqslant \varepsilon + \xi_i^*, \quad i = 1, 2 \ldots, N \\ &a_i - w\phi(x_i) - b \leqslant \varepsilon + \xi_i, \quad i = 1, 2 \ldots, N \\ &\xi_i, \xi_i^* \geqslant 0, \quad i = 1, 2 \ldots, N. \end{aligned}$$

This constrained optimization problem is solved using the following primal Lagrangian form:

$$\begin{aligned} L(w, b, \xi, \xi^*, \alpha_i, \alpha_i^*, \beta_i, \beta_i^*) = {} & \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{N}(\xi_i + \xi_i^*)\right) - \sum_{i=1}^{N}\beta_i[w\phi(x_i) + b - a_i + \varepsilon + \xi_i] \\ & - \sum_{i=1}^{N}\beta_i^*[a_i - w\phi(x_i) - b + \varepsilon + \xi_i^*] - \sum_{i=1}^{N}(\alpha_i\xi_i + \alpha_i^*\xi_i^*). \end{aligned} \tag{5}$$

Eq. (5) is minimized with respect to primal variables w, b, ξ and ξ^*, and maximized with respect to nonnegative Lagrangian multipliers α_i, α_i^*, β_i and β_i^*. Finally, Karush–Kuhn–Tucker conditions are applied to the regression, and Eq. (4) thus yields the dual Lagrangian

$$\vartheta(\beta_i, \beta_i^*) = \sum_{i=1}^{N} a_i(\beta_i - \beta_i^*) - \varepsilon\sum_{i=1}^{N}(\beta_i + \beta_i^*) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\beta_i - \beta_i^*)(\beta_j - \beta_j^*)K(x_i, x_j) \tag{6}$$

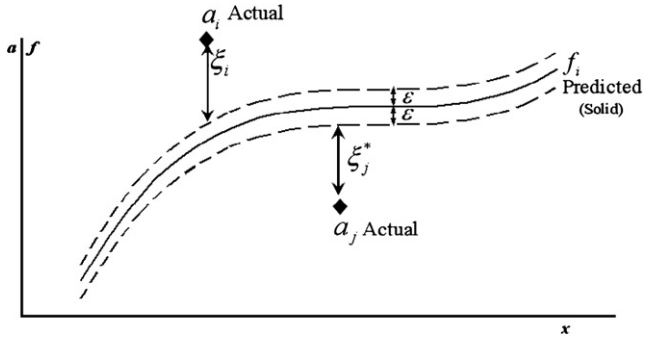

Fig. 4. Parameters used in support vector regression [43].

subject to the constraints

$$\sum_{i=1}^{N}(\beta_i - \beta_i^*) = 0,$$
$$0 \leqslant \beta_i \leqslant C, \quad i = 1, 2, \ldots, N,$$
$$0 \leqslant \beta_i^* \leqslant C, \quad i = 1, 2, \ldots, N.$$

The Lagrange multipliers in Eq. (6) satisfy the equality $\beta_i * \beta_i^* = 0$. The Lagrange multipliers β_i and β_i^*, are calculated and an optimal desired weight vector of the regression hyperplane is

$$w^* = \sum_{i=1}^{N}(\beta_i - \beta_i^*)\phi(x_i). \tag{7}$$

Hence, the regression function is Eq. (8)

$$g(x, \beta, \beta^*) = \sum_{i=1}^{N}(\beta_i - \beta_i^*)K(x, x_i) + b. \tag{8}$$

Here $K(x_i, x_j)$ is called the Kernel function. The value of the Kernel equals the inner product of two vectors, x_i and x_j, in the feature space $\phi(x_i)$ and $\phi(x_j)$; that is, $K(x_i, x_j) = \phi(x_i)^* \phi(x_j)$. Any function that meets Mercer's condition [33] can be used as the Kernel function. In this work, the Gaussian function, $\exp\left(-\frac{1}{2} * \left(\frac{\|x_i - x_j\|}{\sigma}\right)^2\right)$, is used in the SVR. The forecasting process of a SVR model is illustrated as in Fig. 5.

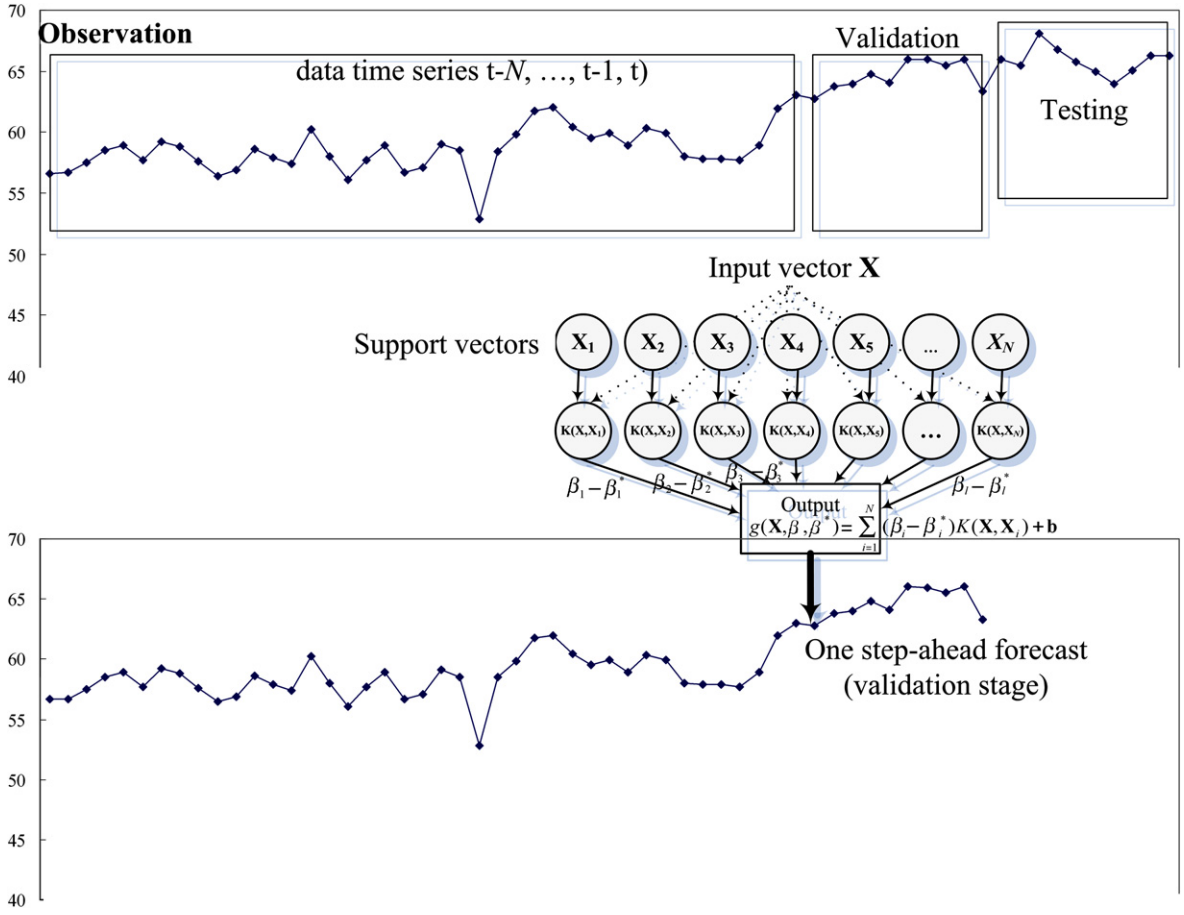

Fig. 5. The forecasting process of a SVR model.

The selection of three parameters, σ, ε and C, of a SVR model is important to the accuracy of forecasting. However, structural methods for confirming efficiently the selection of parameters efficiently are lacking. Therefore, chaotic particle swarm optimization (CPSO) is used in the proposed SVR model to optimize parameter selection.

2.2. CPSO algorithms in selecting parameters of the SVR model

Due to the simple concept, easy implementation and quick convergence, nowadays Particle swarm optimization [34] has gained much attention and wide applications in solving continuous nonlinear optimization problems [35]. In a PSO system, it started with the random initialization of a population (swarm) in the problem search space, multiple candidate solutions coexisted. Each solution, namely particle, flied in the space looking for the optimal *position* to land. Eventually, the global best position of the system could be found out by adjusting the direction of each particle towards its own best location and towards the best particle of the swarm at each generation. The direction of each particle is adjusted by dynamically altering the *velocity* of each particle, according to its own flying experience as well as the experience of neighboring particles. During the searching process, tracking and memorizing the best position encountered could cumulate each particle's experience. Thus, the PSO system essentially had the capability of memory, each particle remembers the best position it reached during the past, then, the PSO system combines local search method (via self experience) with global search methods (via neighboring experience).

The position, the velocity, and own best position of the ith particle pair, due to the three parameters in a SVR model, in the n-dimensional space can be represented as Eqs. (9)–(11), respectively,

$$X_{(k)i} = [x_{(k)i,1}, x_{(k)i,2}, \ldots, x_{(k)i,n}], \tag{9}$$

$$V_{(k)i} = [v_{(k)i,1}, v_{(k)i,2}, \ldots, v_{(k)i,n}], \tag{10}$$

$$P_{(k)i} = [p_{(k)i,1}, p_{(k)i,2}, \ldots, p_{(k)i,n}], \tag{11}$$

$$k = C, \varepsilon, \sigma, \quad i = 1, 2, \ldots, N.$$

The global best position among all particles in the swarm $\mathbf{X}_{(k)} = [X_{(k)1}, X_{(k)2}, \ldots, X_{(k)N}]$ is shown as Eq. (12)

$$P_{(k)g} = [p_{(k)g,1}, p_{(k)g,2}, \ldots, p_{(k)g,d}], \quad k = C, \varepsilon, \sigma, \quad i = 1, 2, \ldots, N. \tag{12}$$

Then, the new velocity of each particle is computed by Eq. (13).

$$V_{(k)i}(t+1) = lV_{(k)i}(t) + q_1 \mathrm{rand}(\cdot)(P_{(k)i} - X_{(k)i}(t)) + q_2 \mathrm{Rand}(\cdot)(P_{(k)g} - X_{(k)i}(t)), \quad k = C, \varepsilon, \sigma, \quad i = 1, 2, \ldots, N, \tag{13}$$

where l is called the inertia weight that controls the impact of the previous velocity of the particle on its current one; q_1 and q_2 are two positive constants called acceleration coefficients; rand$(\cdot)$ and Rand$(\cdot)$ are two independently uniformly distributed random variables with range $[0, 1]$.

After the velocity has been updated, the new position of the particle for each parameter in the next generation is determined as Eq. (14)

$$X_{(k)i}(t+1) = X_{(k)i}(t) + V_{(k)i}(t+1), \quad k = C, \varepsilon, \sigma, \quad i = 1, 2, \ldots, N. \tag{14}$$

Notice that the value of each component in $V_{(k)i}$ can be limited to the range $[-v_{\max}, v_{\max}]$ to control excessive roaming of particles outside the search space. This process is repeated until the defined stopping threshold is reached. The procedure of PSO is illustrated as follow and the flowchart is shown as Fig. 6. Interested readers could refer to [34] for more detail.

Step 1: Initialize a defined population of particle pairs $(C_i, \varepsilon_i, \sigma_i)$ with random positions $(X_{Ci}, X_{\varepsilon i}, X_{\sigma i})$ and velocities $(V_{Ci}, V_{\varepsilon i}, V_{\sigma i})$, where each particle contains n variables.

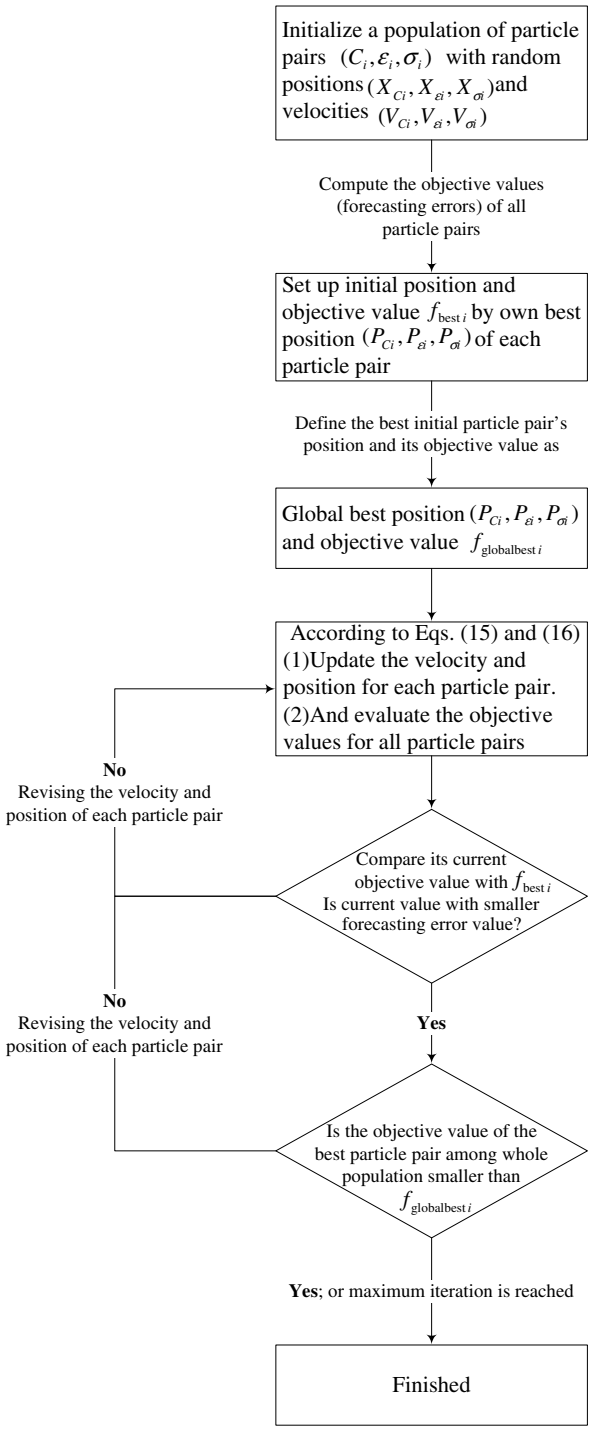

Fig. 6. Particle swarm optimization flowchart.

Step 2: Compute the objective values (forecasting errors) of all particle pairs. Let own best position (P_{Ci}, $P_{\varepsilon i}$, $P_{\sigma i}$) of each particle pair and its objective value $f_{\text{best}i}$ equal to its initial position and objective value. Let global best position (P_{Cg}, $P_{\varepsilon g}$, $P_{\sigma g}$) and its objective value $f_{\text{globalbest}i}$ equal to the best initial particle pair's position and its objective value.

Step 3: According to Eqs. (15) and (16) to update the velocity and position for each particle pair. And evaluate the objective values for all particle pairs.

Step 4: For each particle pair, compare its current objective value with $f_{\text{best}i}$. If current value is better (with smaller forecasting accuracy index value), then, update $(P_{Ci}, P_{\varepsilon i}, P_{\sigma i})$ and its objective value with the current position and objective value.

Step 5: Determine the best particle pair of whole population based on the best objective value. If the objective value is smaller than $f_{\text{globalbest}i}$, then update $(P_{Cg}, P_{\varepsilon g}, P_{\sigma g})$ and its objective value with the current best particle pair's position and objective.

Step 6: If a stopping threshold (forecasting accuracy) is reached, then $(P_{Cg}, P_{\varepsilon g}, P_{\sigma g})$ and its $f_{\text{globalbest}i}$ would be determined; otherwise go back to step 3.

Based on the PSO procedure, the performance is mainly depends on its parameters, and it often lead to be trapped in local optimum [36]. For example, the inertia weight l that impacts by the previous velocity on the current one, a larger inertia weight pressures towards global exploration, while a smaller inertia weight pressures toward fine-tuning of the current search area. Thus, proper control of l is very important to find the optimum solution accurately. To deal with this shortcoming, Liu et al. [37] employed a chaotic PSO (CPSO) method that combined PSO with AIWF (adaptive inertia weight factor) and chaotic local search (CLS). In addition, Cai et al. [38] also introduce another CPSO model, by applying other famous chaotic system, Tent equation, to combine PSO with AIWF. Based on their results, there is no significant difference among these two CPSO. Due to overcoming trapped into local minimum of PSO, this investigation only applied Liu et al.'s CPSO to test the potentiality in searching three parameters in a SVR model.

AIWF is used to encourage good particles (pairs) to revise their exploration to refine results by local search, and bad ones to modify searching space with large step. AIWF is determined as Eq. (15)

$$l = \begin{cases} l_{\min} + \frac{(l_{\max} - l_{\min})(f_i - f_{\min})}{f_{\text{avg}} - f_{\min}}, & f_i \leqslant f_{\text{avg}}, \\ l_{\max}, & f_i \geqslant f_{\text{avg}}, \end{cases} \tag{15}$$

where $l_{\max}$ and $l_{\min}$ are the maximum and minimum of l, respectively. f_i is the current objective value of ith particle pair, f_{avg} and $f_{\min}$ are the average and minimum objective values of all particle pairs, respectively.

CLS is used to perform locally oriented search (exploitation) for the solution $f_{\text{globalbest}i}$, which is resulted from PSO. CLS is based on the logistic equation, with sensitive dependence on initial conditions, and is defined by the following equation:

$$cx_{(k)i}^{(\lambda+1)} = 4cx_{(k)i}^{(\lambda)}(1 - cx_{(k)i}^{(\lambda)}), \quad k = C, \varepsilon, \sigma, \ i = 1, 2, \ldots, N, \tag{16}$$

where $cx_{(k)i}$ is the ith chaotic variable; λ represents the iteration number. $cx_{(k)i}^{(\lambda)}$ is distributed in the range $(0, 1)$ and $cx_{(k)i}^{(0)} \in (0, 1)$ but $cx_{(k)i}^{(0)} \notin \{0.25, 0.5, 0.75\}$.

The procedure of CLS is illustrated as follows:

Step 1: Setting $\lambda = 0$, and employing Eq. (17) to map the three parameters $X_{(k)i}^{(\lambda)}$, $k = C, \varepsilon, \sigma$, $i = 1, 2, \ldots, N$ among the intervals $(x_{\min(k)i}, x_{\max(k)i})$ into chaotic variable $cx_{(k)i}^{(\lambda)}$ located in the interval $(0, 1)$

$$cx_{(k)i}^{(\lambda)} = \frac{X_{(k)i}^{(\lambda)} - x_{\min(k)i}}{x_{\max(k)i} - x_{\min(k)i}}, \quad i = 1, 2, \ldots, N. \tag{17}$$

Step 2: By using Eq. (14) to compute the next iteration chaotic variable, $cx_{(k)i}^{(\lambda+1)}$.

Step 3: Transform $cx_{(k)i}^{(\lambda+1)}$ to obtain three parameters for the next iteration, $X_{(k)i}^{(\lambda+1)}$, by the following Eq. (18):

$$X_{(k)i}^{(\lambda+1)} = x_{\min(k)i} + cx_{(k)i}^{(\lambda+1)}(x_{\max(k)i} - x_{\min(k)i}). \tag{18}$$

Step 4: Compute the new objective value with $X_{(k)i}^{(\lambda+1)}$.

Step 5: If the new objective value with smaller forecasting accuracy index value or maximum iteration is reached, then, the new chaotic variable $X_{(k)i}^{(\lambda+1)}$ and its corresponding objective value is the final solution; otherwise, let $\lambda = \lambda + 1$ and go back to Step 2.

In the investigation, the normalized mean squared error (NMSE), shown as Eq. (19), serves as the forecasting accuracy index for identifying suitable parameters, determined in Step 4 of PSO and in Step 5 of CLS, for use in the SVRCPSO model.

$$\mathrm{NMSE} = -\frac{1}{n\delta^2}\sum_{i=1}^{n}(a_i - f_i)^2, \tag{19}$$

where $\delta^2 = \frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})^2$ and n is the number of forecasting periods; a_i is the actual value at period i; f_i denotes is the forecasting value at period i.

2.3. Recurrent SVR with CPSO algorithms

In this investigation, the Jordan network is employed as a recurrent neural network framework. Fig. 1 presents the architecture of a Jordan network [27]. All neurons in a layer except those in the context layer are connected with all neurons in the next layer. A context layer is a special hidden layer. Interactions only occur between neurons in the hidden layer and those in the context layer. For a Jordan network with p inputs, q hidden and r output neurons, the output of the n th neuron, $f_n(t)$, is [39–42]

$$f_n(t) = \sum_{i=1}^{q} W_i\varphi_i(t) + b_i(t), \tag{20}$$

where W_i are weights between the hidden and output layers, and $\varphi_i(t)$ is the output function of the hidden neurons, which is,

$$\varphi_i(t) = g\left(\sum_{j=1}^{P} v_{ij}x_j(t) + \sum_{k=1}^{s}\sum_{v=1}^{r} w_{ikv}f_v(t-k) + b_i(t)\right), \tag{21}$$

where v_{ij} are weights between the input and the hidden layer; w_{ikv} are weights between the context and the hidden layer with k delay periods and s is the total number of context layers in past output data.

Back-propagation yields gradients for adapting weights of a neural network. The back-propagation algorithm is presented as follows. First, the output of the n th neuron in Eq. (21) is rewritten as

$$f_n(t) = h(x^{\mathrm{T}}(t)\phi(t)), \tag{22}$$

where $h(\cdot)$ is the nonlinearity function of $x^{\mathrm{T}}(t)$ and $f_n(t)$; $x^{\mathrm{T}}(t) = [x_1(t), \ldots, \mathrm{x}x_P(t)]^{\mathrm{T}}$ is the input vector; $\phi(t) = [\phi_1(t), \ldots, \phi_P(t)]^{\mathrm{T}}$ is the weight vector; a cost function is then presented to be the instantaneous performance index

$$J(\phi(t)) = \frac{1}{2}[d(t) - f_n(t)]^2 = \frac{1}{2}[d(t) - h(x^{\mathrm{T}}(t)\phi(t))]^2, \tag{23}$$

where $d(t) = [d_1(t), \ldots, \mathrm{x}d_P(t)]^{\mathrm{T}}$ is the desired output.

Secondly, the instantaneous output error at the output neuron and the revised weight vector in the next moment are given by Eqs. (24) and (25) respectively

$$e(t) = d(t) - f_n(t))) = d(t) - h(x^{\mathrm{T}}(t)\phi(t)), \tag{24}$$

$$\phi(t+1) = \phi(t) - \eta\nabla_{\phi}J(\phi(t)), \tag{25}$$

where η is the learning rate.

Third, the gradient $\nabla_{\phi}J(\phi(t))$ can be calculated as

$$\nabla_{\phi}J(\phi(t)) = \frac{\partial J(\phi(t)}{\partial\phi(t)} = e(t) \times \frac{\partial e(t)}{\partial\phi(t)} = -e(t)h'(x^{\mathrm{T}}(t)\phi(t))x(t), \tag{26}$$

where $h'(\cdot)$ is the first derivative of the nonlinearity $h(\cdot)$. Finally, the weight is revised as

$$\phi(t+1) = \phi(t) + \eta e(t)h'(x^{\mathrm{T}}(t)\phi(t))x(t). \tag{27}$$

Fig. 7 shows the architecture of the proposed RSVRCPSO model. The output of RSVRCPSO ($\tilde{f}_n(t)$) is

$$\tilde{f}_n(t) = \sum_{i=1}^{P} W^{\mathrm{T}} \psi(x^{\mathrm{T}}(t)) + b(t). \tag{28}$$

Then, Eq. (28) replaces Eq. (1) in the SVRCPSO algorithms, to run the loop of SVRCPSO in the search for values of three parameters. Finally, the forecast values $\tilde{f}_n(t)$ are calculated using Eq. (28).

3. A numerical example and experimental results

3.1. The study watershed and indices of performance evaluation

The Wu-Tu Watershed rainfall data is employed as a case study for development of rainfall forecasting model in this investigation. The watershed is located in northern Taiwan, its main river is Kee-Lung River, the watershed size is about 204 km^2, and the annual average rainfall is about 2500 mm. Within the watershed, the dominant land use is typical of urban environments with a mix of residential, cultural and educational areas, therefore, the watershed serve as security for human life and property. There are three rain gauges (Wu-tu, Jui-fang, and Huo-shao-liao) and one discharge site (Wu-tu) within the Wu-Tu Watershed, locations of these gauges is shown in Fig. 8.

Rainfall data is obtained from August 1985 to August 1997 at the three rain gauges. During this period, nine typhoon events occurred with duration hours total greater than 40 h, cumulative rainfall depths greater than 300 mm, with 673 rainfall amounts at each site (Table 1). In this paper, these 673 rainfall amounts are divided into three periods: training (300 h), validation (206 h), and testing (167 h).

The accuracy of the proposed rainfall forecasting model is measured as normalized mean squared error (NMSE), given by Eq. (19). The minimum values of NMSE indicate that the deviations between actual values and forecast values are very small. The accurate efficiency of the proposed model is measured in terms of coefficient of efficiency (CE) and coefficient of correlation (CC), given by Eq. (29) and (30) respectively. The higher values of CE (maximum value is 1) indicate that the forecasting performance of the proposed

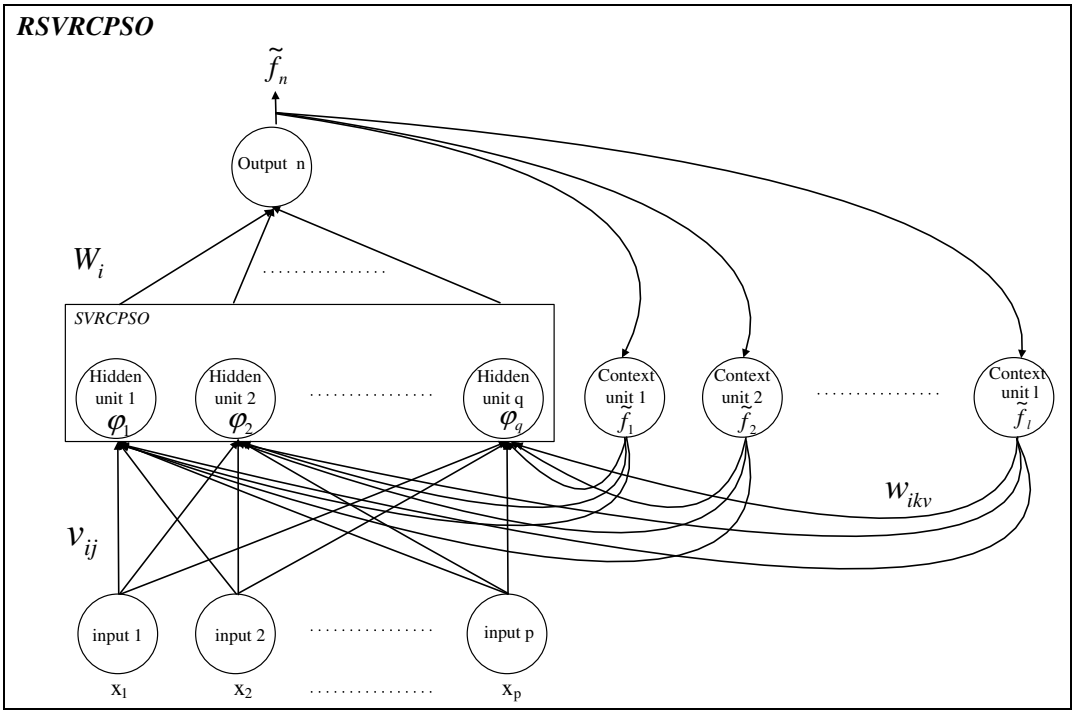

Fig. 7. The architecture of RSVRCPSO model.

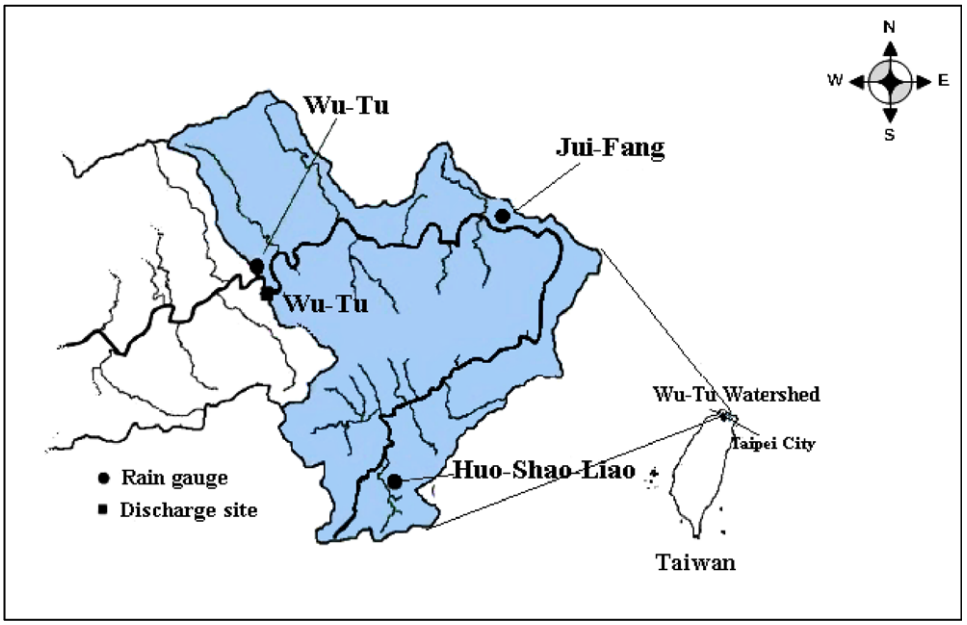

Fig. 8. The Wu-Tu Watershed in Northern Taiwan.

Table 1
Description of typhoon selected from Wu-Tu discharged site

Number	Name	Date (yyyy/mm/dd)	Rainfall duration (h)	Rainfall depth (mm)	Remark
1	Nelson	1985/08/22	41	341.4	Training
2	Abby	1986/09/17	88	521.3	Training
3	Sarah	1989/09/10	71	322.5	Training
4	Ruth	1991/10/28	72	448.6	Validation
5	Polly	1992/08/29	85	500.6	Validation
6	Seth	1994/10/09	49	300.7	Validation
7	Herb	1996/07/31	43	313.6	Testing
8	Zane	1996/09/27	77	440.6	Testing
9	Winnie	1997/08/17	47	343.5	Testing

model is effective. Similarly, the higher values of CE (maximum value is 1) indicate that the proposed model can capture the average change tendency of the cumulative rainfall data.

$$\mathrm{CE} = 1 - \frac{\sum_{i=1}^{n}(a_i - f_i)^2}{\sum_{i=1}^{n}(a_i - \bar{a})^2}, \tag{29}$$

$$\mathrm{CC} = \frac{\sum_{i=1}^{n}(a_i - \bar{a})(f_i - \bar{f})}{\sqrt{\sum_{i=1}^{n}(a_i - \bar{a})^2 * \sum_{i=1}^{n}(f_i - \bar{f})^2}}, \tag{30}$$

where a_i and f_i represent the actual and forecast rainfall volumes, respectively; $\bar{a}$ and $\bar{f}$ represent the actual and forecast mean rainfall volumes, respectively; and n is the number of forecasting periods.

3.2. Parameters setting in the CPSO algorithm

The parameters of the CPSO algorithm in the proposed model for two numerical examples are experimentally set as shown in Table 2. The population sizes are both 20; the total number of function evaluation is fixed

Table 2
CPSO's Parameters setting in both numerical examples

Numerical models	Population size	Maximal iteration	Velocity limit						Inertia weight factor l		Acceleration coefficients					
			$-v_{max}$			v_{max}			l_{min}	l_{max}	q_1			q_2		
			σ	C	ε	σ	C	ε			σ	C	ε	σ	C	ε
SVRCPSO	20	10,000	−0.5	−25	−0.05	0.5	25	0.05	0.8	1.25	0.05	100	0.5	0.05	100	0.5
RSVRCPSO	20	10,000	−0.5	−25	−0.05	0.5	25	0.05	0.8	1.25	0.05	100	0.5	0.05	100	0.5

as 10,000; q_1 and q_2 for each particle pair (σ, C, ε) are set to 0.05, 100, 0.5, respectively. v_{max} for σ particle are both clamped to be 10% of its search space (where $\sigma \in [0,5]$). v_{max} for C particle in example one is clamped to be 12.5% of its search space ($C \in [0,20000]$); v_{max} for C particle in example two is clamped to be 15% of its search space ($C \in [0,300000]$). v_{max} for ε particle are both clamped to be 15% of its search space ($\varepsilon \in [0,100]$). The standard PSO [34] uses a linearly varying inertia weight over the generations, varying from 1.2 at the beginning of the search to 0.2 at the end. The CPSO uses the AIWF defined in Eq. (15) with $l_{max} = 1.2$ and $l_{min} = 0.2$.

3.3. Parameter determination of SVRCPSO and RSVRCPSO models

A rolling-based forecasting procedure was conducted and a one-hour-ahead forecasting policy adopted. Generally, several types of data-rolling should be considered as a time series to feed into the SVRCPSO and RSVRCPSO models to forecast rainfall depth in the next hour. However, in the training stage, the rainfall data contains three typhoon events, therefore, the suitable numbers of rainfall data fed into SVRCPSO and RSVRCPSO models should be determined under the following two considerations. Firstly, it is necessary to completely capture rainfall data pattern from each typhoon event during the training stage. Secondly, it is also required to avoid the over-fitting problem which could be learned from training experiences. Hence, the suitable numbers of rainfall data fed into SVRCPSO and RSVRCPSO model should be set as the total duration rainfall hours of the previous two typhoons, i.e., 129 rainfall duration hours. To compare the forecasting performance regarding different input numbers of rainfall data fed into SVRCPSO and RSVRCPSO models, only containing one typhoon event, i.e., 41 rainfall duration hours, is also conducted.

Table 3
Forecasting results and associated parameters of the SVRCPSO model

Numbers of input data	Parameters			NMSE of training	NMSE of validation	NMSE of testing
	σ	C	ε			
41[a]	0.008097	7.3571	0.49353	1.17718	0.816256	0.940341
129[b]	**0.067658**	**16.897**	**0.30338**	**1.26967**	**0.499716**	**0.496088**

[a] Including the total rainfall durations of one typhoon event, Nelson, in training stage.
[b] Including the total rainfall durations of two typhoon events, Nelson and Abby, in training stage.

Table 4
Forecasting results and associated parameters of the RSVRCPSO model

Numbers of input data	Parameters			NMSE of training	NMSE of validation	NMSE of testing
	σ	C	ε			
41[a]	0.035642	34.99	0.49762	1.13518	0.659666	0.896193
129[b]	**0.047490**	**36.05**	**0.35581**	**1.15924**	**0.402853**	**0.375218**

[a] Including the total rainfall durations of one typhoon event, Nelson, in training stage.
[b] Including the total rainfall durations of two typhoon events, Nelson and Abby, in training stage.

Then, via training processes, the parameters of the SVRCPSO and RSVRCPSO models and their forecast accuracy results are presented as Tables 3 and 4, respectively. However, both SVRCPSO and RSVRCPSO models with contained one typhoon event (41 rainfall duration hours) suffered over-fitting problem, it is clearly to observe that the NMSE values of the two models are smaller in training stage, but finally increase in validation stage and testing stage.

3.4. Forecasting results

Figs. 9–11 compares the actual hourly rainfall depth value and 1-h ahead forecasting for training data, validation data and testing data respectively. These figures indicate that the shape as well as the tendency of rainfall hydrograph could be forecasted employing the proposed RSVRCPSO model (rainfall duration hours).

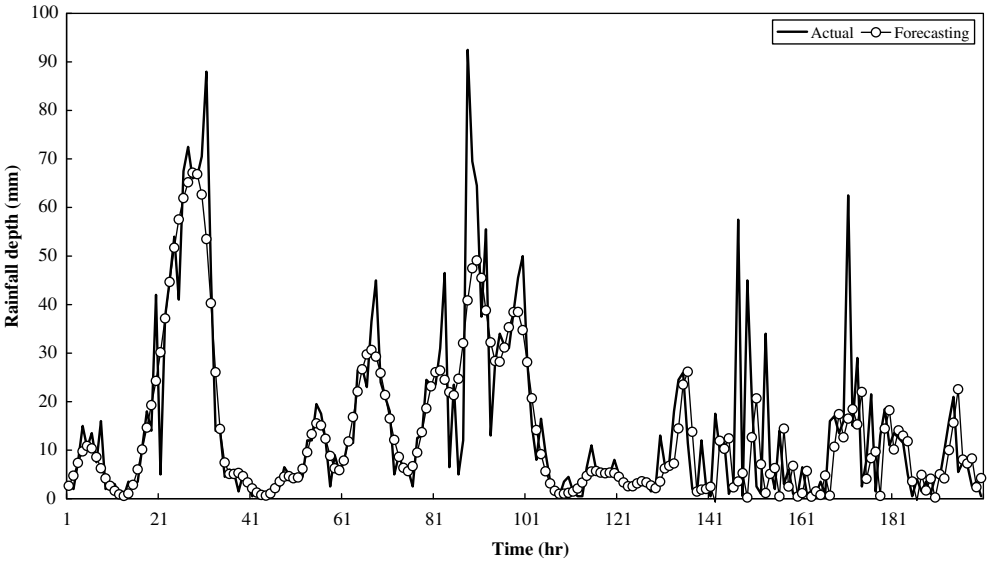

Fig. 9. Comparison of actual rainfall depth value and 1-h forecasts for training data (RSVRCPSO).

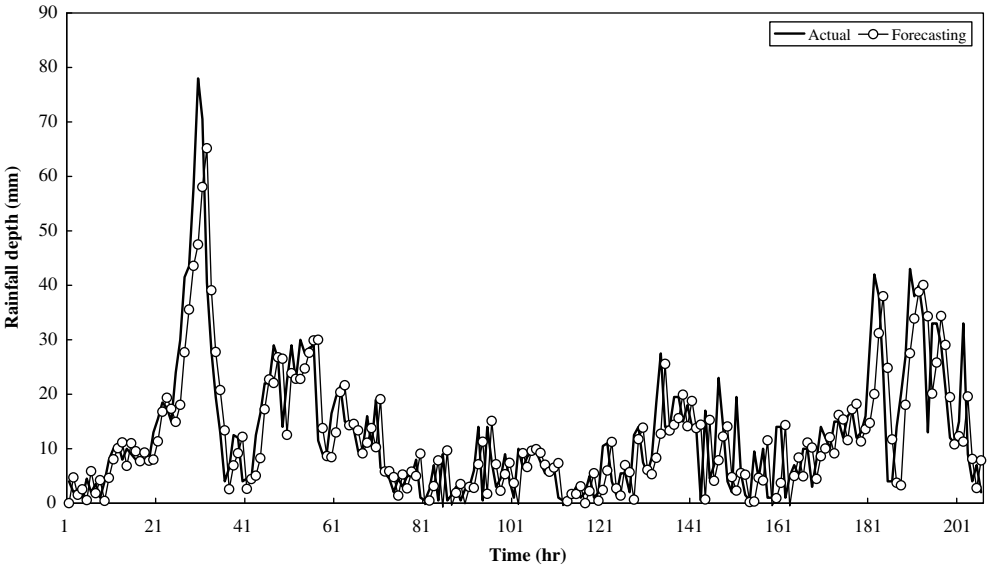

Fig. 10. Comparison of actual rainfall depth value and 1-h forecasts for validation data (RSVRCPSO).

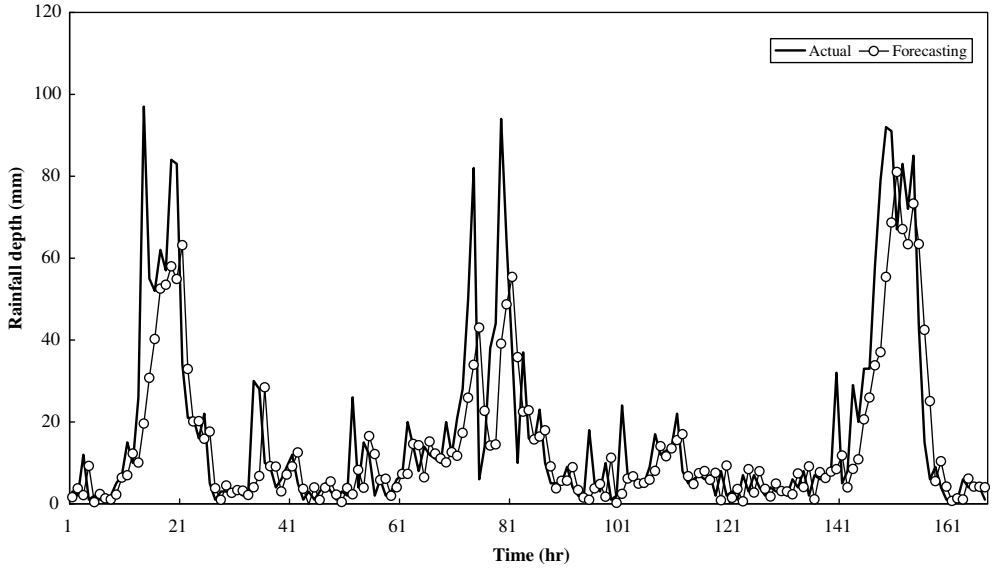

Fig. 11. Comparison of actual rainfall depth value and 1-h forecasts for testing data (RSVRCPSO).

Table 5
Forecasting accuracy of the RSVRCPSO model in terms of various evaluation indices

Forecasting performance evaluation indices	Training	Validation	Testing
NMSE	1.15924	0.402853	0.375218
CE	0.67026	0.597145	0.624783
CC	0.82360	0.784369	0.803865

Table 5 illustrated the forecasting accuracy and efficiency of the RSVRCPSO model (129 rainfall duration hours) in terms of various evaluation indices. For NMSE accuracy index, the proposed RSVRCPSO model with satisfactory forecasting performance and is capable to be employed to forecast rainfall depth during typhoon period. Similarly, for CE efficiency index, the proposed RSVRCPSO model is also deserved to be confident, particularly in testing stage. For CC efficiency index, the forecasting rainfall depth values from RSVRCPSO model have higher correlation relationship with actual rainfall depth values; it also implies that RSVRCPSO model is capably to capture the average change tendency of the cumulative rainfall data. In comparison with SVRCPSO model, the RSVRCPSO model with little values of NMSE in training stage, validation stage, and testing stage reveals its alternative superiority in rainfall forecasting issues.

4. Conclusions

Accurate rainfall forecasting is crucial for a frequent-unanticipated flash flood region to avoid life losing and economic loses. The historical rainfall data of Wu-Tu watershed in northern Taiwan shows a fluctuation trend, particularly during the typhoon season. Therefore, over-prediction or under-prediction rainfall amounts influence the social capability and costs in precaution against flash flood a lot. This study introduced a hybrid forecasting techniques, recurrent neural networks and support vector regression with chaotic particle swarm optimization algorithm, RSVRCPSO, to investigate its feasibility in forecasting typhoon rainfall amounts. The experimental results indicate that the RSVRCPSO model has accurate forecasting performance in terms of forecasting accuracy indices. The superior performance of the RSVRCPSO model has several causes. First, the RSVRCPSO model has nonlinear mapping capabilities and thus can more easily capture electricity load data patterns than can the ANN and regression models. Second, improper determining of these three parameters will cause either over-fitting or under-fitting of a SVR model. In this work, chaotic particle

swarm optimization algorithm can determine suitable parameters to forecast typhoon rainfall depth data. Third, the RSVRCPSO model performs structural risk minimization rather than minimizing the training errors. Minimizing the upper bound on the generalization error improves the generalization performance compared to the ANN and regression models. Finally, Jordan recurrent networks can continually capture data patterns from the output layer with past values into the hidden layer.

This investigation is the first to apply the hybrid model of recurrent neural networks and SVR with CPSO to typhoon rainfall amounts forecasting. The empirical results obtained in this study demonstrate that the proposed model offers a valid alternative for application in the hydrology. In the future, more other meteorological control variables during the typhoon season, surface pressure, temperature, convection condition and wind speed and direction, can be included in the RSVRCPSO model for forecasting rainfall amounts. In addition, some other advanced searching techniques for suitable parameters selection can be combined with SVR to forecast rainfall amounts.

References

[1] D.P. Lettenmaier, E.F. Wood, Hydrology forecasting, in: D.R. Maidment (Ed.), Handbook of Hydrology, McGraw-Hill, New York, 1993.
[2] Q. Duan, A global optimization strategy for efficient and effective calibration of hydrologic models, Ph.D. dissertation, University of Arizona, Tucson, 1991.
[3] K.C. Luk, J.E. Ball, A. Sharma, An application of artificial neural networks for rainfall forecasting, Mathematical and Computer Modelling 33 (6-7) (2001) 683–693.
[4] D.J. Druce, Insights from a history of seasonal inflow forecasting with a conceptual hydrologic model, Journal of Hydrology 249 (1-4) (2001) 102–112.
[5] B.E. Vieux, Z. Cui, A. Gaur, Evaluation of a physics-based distributed hydrologic model for flood forecasting, Journal of Hydrology 298 (1–4) (2004) 155–177.
[6] P. Yapo, V.K. Gupta, S. Sorooshian, Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data, Journal of Hydrology 181 (1–4) (1996) 23–48.
[7] Q. Duan, S. Sorooshian, V.K. Gupta, Optimal use of sce-ua global optimization method for calibrating watershed models, Journal of Hydrology 158 (3–4) (1994) 265–284.
[8] G.E.P. Box, G.M. Jenkins, Time Series Analysis: Forecasting and Control, Holden-Day, San Francisco, 1976.
[9] M.N. French, W.F. Krajewski, R.R. Cuykendall, Rainfall forecasting in space and time using a neural network, Journal of Hydrology 137 (1–4) (1992) 1–31.
[10] K.C. Luk, J.E. Ball, A. Sharma, A study of optimal model lag and spatial inputs to artificial neural network for rainfall forecasting, Journal of Hydrology 227 (1–4) (2000) 56–65.
[11] M.C. Valverde Ramírez, H.F. de Campos Velho, N.J. Ferreira, Artificial neural network technique for rainfall forecasting applied to The São Paulo Region, Journal of Hydrology 301 (1–4) (2005) 146–162.
[12] T.Y. Pan, R.Y. Wang, State space neural networks for short term rainfall-runoff forecasting, Journal of Hydrology 297 (1–4) (2004) 34–50.
[13] G.F. Lin, L.H. Chen, A non-linear rainfall-runoff model using radial basis function network, Journal of Hydrology 289 (1–4) (2004) 1–8.
[14] Y.M. Chiang, L.C. Chang, F.J. Chang, Comparison of static-feedforward and dynamic-feedback neural networks for rainfall-runoff modeling, Journal of Hydrology 290 (3–4) (2004) 297–311.
[15] M. Castellano-Méndez, W. González-Manteiga, M. Febrero-Bande, J.M. Prada-Sánchez, R. Lozano-Calderón, Modelling of the monthly and daily behaviour of the runoff of The Xallas River using Box–Jenkins and neural networks methods, Journal of Hydrology 296 (1–4) (2004) 38–58.
[16] F. Anctil, C. Michel, C. Perrin, V. Andréassian, A soil moisture index as an auxiliary ann input for stream flow forecasting, Journal of Hydrology 286 (1–4) (2004) 155–167.
[17] A.G. El-Din, D.W. Smith, A neural network model to predict the wastewater inflow incorporating rainfall events, Water Research 36 (5) (2002) 1115–1126.
[18] F.E.H. Tay, L. Cao, Application of support vector machines in financial time series forecasting, OMEGA – International Journal of Management Science 29 (4) (2001) 309–317.
[19] W. Wang, Z. Xu, J.W. Lu, Three improved neural network models for air quality forecasting, Engineering Computations 20 (2) (2003) 192–210.
[20] L. Cao, Q. Gu, Dynamic support vector machines for non-stationary time series forecasting, Intelligent Data Analysis 6 (1) (2002) 67–83.
[21] F.E.H. Tay, L. Cao, Modified support vector machines in financial time series forecasting, Neurocomputing 48 (1–4) (2002) 847–861.
[22] L. Cao, Support vector machines experts for time series forecasting, Neurocomputing 51 (2003) 321–339.
[23] M.A. Mohandes, T.O. Halawani, S. Rehman, A.A. Hussain, Support vector machines for wind speed prediction, Renewable Energy 29 (6) (2004) 939–947.

[24] P.F. Pai, C.S. Lin, Using support vector machines in forecasting production values of machinery industry in Taiwan, International Journal of Advanced Manufacturing Technology 27 (1–2) (2004) 205–210.
[25] P.F. Pai, C.S. Lin, A hybrid ARIMA and support vector machines model in stock price forecasting, OMEGA – International Journal of Management Science 33 (6) (2005) 497–505.
[26] G. Kechriotis, E. Zervas, E.S. Manolakos, Using recurrent neural networks for adaptive communication channel equalization, IEEE Transaction on Neural Networks 5 (2) (1994) 267–278.
[27] M.I. Jordan, Attractor dynamics and parallelism in a connectionist sequential machine, in: Proceeding of 8th Annual Conference of the Cognitive Science Society, Hillsdale, 1987, pp. 531–546.
[28] J.L. Elman, Finding structure in time, Cognitive Science 14 (2) (1990) 179–211.
[29] R. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Computation 1 (1989) 270–280.
[30] A.C. Tsoi, A.D. Back, Locally recurrent globally feedforward networks: a critical review of architectures, IEEE Transaction on Neural Networks 5 (2) (1994) 229–239.
[31] W.C. Jhee, J.K. Lee, Performance of neural networks in managerial forecasting, International Journal of Intelligent Systems in Accounting, Finance, and Management 2 (1) (1993) 55–71.
[32] J.A.K. Suykens, T. van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, K.U. Leu-ven, Least Squares Support Vector Machines, World Scientific Publishing Co. Ltd, Belgium, 2002.
[33] V. Vapnik, The Nature of Statistical Learning Theory, Springer Verlag, New York, 1995.
[34] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceeding of IEEE International Conference Neural Networks, 1995, pp. 1942–1948.
[35] R.C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in: Proceeding of the 2001 Congress on Evolutionary Computation, 2001, pp. 81–86.
[36] P.J. Angeline, Evolutionary optimization versus particle swarm optimization: philosophy and performance differences, Evolutionary Programming VII (Proceedings of the Seventh Annual Conference on Evolutionary Programming), 1998, pp. 601–610.
[37] B. Liu, L. Wang, Y.H. Jin, F. Tang, D.X. Huang, Improved particle swarm optimization combined with chaos, Chaos, Solitons, Fractals 25 (5) (2005) 1261–1271.
[38] J. Cai, X. Ma, L. Li, H. Peng, Chaotic particle swarm optimization for economic dispatch considering the generator constraints, Energy Conversion and Management 48 (2) (2007) 645–653.
[39] J.T. Connor, R.D. Martin, L.E. Atlas, Recurrent neural networks and robust time series prediction, IEEE Transactions on Neural Networks 5 (2) (1994) 240–254.
[40] R. Gencay, T. Liu, Nonlinear modeling and prediction with feedforward and recurrent networks, Physica D: Nonlinear Phenomena 108 (1–2) (1997) 119–134.
[41] B. Kermanshahi, Recurrent neural network for forecasting next 10 years loads of nine Japanese utilities, Neurocomputing 23 (1–3) (1998) 125–133.
[42] D.P. Mandic, J.A. Chambers, Recurrent Neural Networks for Prediction, John Wiley and Sons, New York, 2001.
[43] K. Vojislav, Learning and Soft Computing – Support Vector Machines, Neural Networks and Fuzzy Logic Models, The MIT Press, Massachusetts, 2001.