Available online at www.isas.org.in/jisas
Journal of the Indian Society of Agricultural Statistics 76(3) 2022 121–130

Combining Linear Kalman filter (KF) and Nonlinear Least Squares Support Vector Machines (LSSVM) Models for Forecasting Time-Series Data

Mohan Kumar T.L., Prajneshu and Bishal Gurung

ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Received 23 July 2022; Revised 21 December 2022; Accepted 26 December 2022

SUMMARY

Time-series data are rarely purely linear or nonlinear and often contain both these patterns. In this article, hybrid models are developed by combining linear Kalman Filter (KF) and Nonlinear Least Squares Support Vector Machine (LSSVM) methodologies for time-series forecasting. Particle Swarm Optimization (PSO), which is a very efficient population-based stochastic optimization technique is employed to estimate the hyper-parameters of these models. The relevant computer program is written in MATLAB function (m file) and MATLAB software package is used for data analysis. As an illustration, developed hybrid models are applied to all-India monthly rainfall time-series data. The superiority of these models over individual linear KF and nonlinear LSSVM methodologies is demonstrated for the data under consideration using Root Mean Square Error (RMSE) and Mean Absolute Percent Error (MAPE) criteria.

Keywords: Hybrid models, Least squares support vector machine, Kalman filter, Particle swarm optimization, Indian monsoon rainfall.

1. INTRODUCTION

Well-known parametric Box-Jenkins Autoregressive Integrated Moving Average (ARIMA) and its variant the Seasonal ARIMA (SARIMA) models have virtually dominated analysis of univariate time-series data since 1930s (Narayanan *et al.*, 2013). However, a limitation of these models is that they are "linear". Further, quite often it is noticed that there is no appropriate parametric form to describe satisfactorily the given time-series data. To this end, the area of "Nonparametric Nonlinear Time-Series Modelling" has been growing rapidly during the last three decades or so. In this class, Artificial Neural Network (ANN) methodology trained Backpropagation algorithm has been extensively used (Chattopadhyay and Chattopadhyay, 2013). However, the main drawbacks of this methodology are that the model generally overfits data and that the solution gets trapped in local minima.

A very powerful methodology, which does not suffer from the above limitations for regression and time-series problems, is the Nonparametric Nonlinear Support Vector Machine (NLSVM) model (Vapnik, 2000). It implements Structural Risk Minimization (SRM) principle, which has been shown to be superior to the traditional Empirical Risk Minimization (ERM) principle implemented in ANN models. The most important concept of the SRM principle is to minimize the upper bound to the 'Generalization error' rather than minimizing the 'Training error' as in the ERM principle. NLSVM is equivalent to solving a linear constrained quadratic programming (QP) problem, and its solution is always unique and globally optimal. However, the main drawbacks of NLSVR are that the amount of computations becomes larger and the learning rate is greatly cut down with the increasing amount of data for training (Zhou and Ma, 2013). Recently, the Least Squares version of SVM, known as Least Squares Support Vector Machine (LSSVM) was proposed (Suykens *et al.* 2002) as a reformulation of standard NLSVM for solving nonlinear regression and time-series problems. A heartening aspect of

Corresponding author: Bishal Gurung
E-mail address: bishal.gurung@icar.gov.in

Nonlinear LSSVM as compared to standard NLSVM for regression problems is that it applies linear least squares criterion to the loss function with only equality type constraints to obtain a set of linear equations instead of the ε-insensitive loss function with inequality type constraints to form traditional convex QP problem. This leads to the advantages of fast convergence, high accuracy, and low computational effort.

It may be emphasized that only linear or only nonlinear models are not adequate for describing real-world time-series. Accordingly, the hybrid methodology that involves both these types of modelling capabilities provides a robust method and a good alternative for forecasting time-series data. Several authors have developed hybrid methodologies by combining linear ARIMA/SARIMA models with Nonlinear ANN or NLSVM models (e.g., Zhang, 2003; Chen and Wang 2007a; and Khashei and Bijari2011). Lama *et al.* (2022) modelled and forecasted rainfall data for hilly regions such as state of Sikkim and adjoining areas of West Bengal through SARIMA, exponential autoregressive, and non-parametric time delay neural network model. Gurung *et al.* (2017) combined two competing models to improve the forecasts of the volatility process. A new method of combining the volatility of these two competing models using the powerful technique of the Kalman filter was also developed. Ghosh *et al.* (2011) combined linear and nonlinear time-series models for cyclical data.

In this article, we have developed four hybrid models by combining Linear KF and Nonlinear LSSVM methodologies. Despite the superior features of these methodologies, their generalization and efficiency are sensitive to the values of hyper-parameters. Therefore, the selection of optimal hyper-parameters is an important step in Nonlinear LSSVM modelling. To this end, Grid-search is the most commonly employed method (Gestel *et al.*, 2004). However, this approach is time-consuming. Pai and Hong (2005) employed simulated annealing algorithms. However, the main drawback of this technique is that the rate of convergence is very slow. Further, Chen and Wang (2007b) discussed the advantages and disadvantages of various efficient estimation procedures and also proposed a Real-valued GA technique. However, GA is capable of finding the solution through evolution operators, such as crossover and mutation (Yang *et al.*, 2011). Evolution is inductive. In reality, life does not always revolve around a good solution. Fortunately, a very efficient population-based stochastic optimization algorithm, viz. Particle Swarm Optimization (PSO) technique is capable of rectifying the above limitations (Parsopoulos and Vrahatis, 2010). In contrast to GA, which exploits competitive characteristics of biological evolution, PSO simulates cooperative and social behavior such as fish schooling, bird flocking, or insect swarming. So, in this article, we have employed PSO for estimating hyper-parameters of various models.

Organization of the present article is as follows. After a brief introduction, various models are discussed in Section 2. Section 3 deals with the description of the PSO algorithm and procedure for estimating hyper-parameters of Nonlinear LSSVM and hybrid models. As an illustration, all-India monthly rainfall time-series data are considered, and the entire data analysis is reported in Section 4. Finally, the conclusion is presented in Section 5.

2. MATERIALS AND METHODS

2.1 Linear Kalman Filter (KF) Model

State space modelling includes the State transition equation, i.e. Eq.(1), which allows the state variable $\boldsymbol{\alpha}_t$ to change through time, and the measurement equation, i.e. Eq.(2), which relates the state variable to an observation Y_t:

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{F}_t\boldsymbol{\alpha}_t + \boldsymbol{G}_t\boldsymbol{\varepsilon}_t, \tag{1}$$

$$Y_t = \boldsymbol{H}_t^T\boldsymbol{\alpha}_t + \nu_t, \tag{2}$$

where superscript T indicates transpose. It is assumed that $\{\boldsymbol{\varepsilon}_t\}$ in Eq.(1) and $\{\nu_t\}$ in Eq.(2) are independent, zero mean, Gaussian white noise process with

$$E\left[\nu_t\nu_t^T\right] = R_t \text{ and } E[\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T] = \boldsymbol{Q}_t. \tag{3}$$

The Linear KF is a recursive algorithm for sequentially updating the state vector given past information $\boldsymbol{\psi}_t$. Denote

$$\hat{\boldsymbol{\alpha}}_{t|t-1} = E\{\boldsymbol{\alpha}_t|\psi_{t-1}\} \text{ and } \hat{\boldsymbol{\alpha}}_{t|t} = E\{\boldsymbol{\alpha}_t|\psi_t\}, t = 0,1,2,\ldots \tag{4}$$

and assume $\hat{\boldsymbol{\alpha}}_{0|-1} = E\{\boldsymbol{\alpha}_0\}$ and $\boldsymbol{\Sigma}_{0|-1} = \boldsymbol{P}_0$. The state vector $\boldsymbol{\alpha}_t$ and its mean squared error $\boldsymbol{\Sigma}_t = E\left[(\boldsymbol{\alpha}_t - \hat{\boldsymbol{\alpha}}_t)(\boldsymbol{\alpha}_t - \hat{\boldsymbol{\alpha}}_t)^T\right]$ are recursively estimated by:

$$\hat{\boldsymbol{\alpha}}_{t|t} = \hat{\boldsymbol{\alpha}}_{t|t-1} + \boldsymbol{\Sigma}_{t|t-1}\boldsymbol{H}_t(\boldsymbol{H}_t^T\boldsymbol{\Sigma}_{t|t-1}\boldsymbol{H}_t + R_t)^{-1}\left(X_t - \boldsymbol{H}_t^T\hat{\boldsymbol{\alpha}}_{t|t-1}\right), \tag{5}$$

$$\boldsymbol{\Sigma}_{t|t} = \boldsymbol{\Sigma}_{t|t-1} - \boldsymbol{\Sigma}_{t|t-1}\boldsymbol{H}_t(\boldsymbol{H}_t^T\boldsymbol{\Sigma}_{t|t-1}\boldsymbol{H}_t + R_t)^{-1}\boldsymbol{H}_t^T\boldsymbol{\Sigma}_{t|t-1}. \quad (6)$$

Using the recursive filter Eqs.(1), (5) and (6), we can obtain $\hat{\boldsymbol{\alpha}}_{t+1|t}$ as

$$\hat{\boldsymbol{\alpha}}_{t+1|t} = \boldsymbol{F}_t\hat{\boldsymbol{\alpha}}_{t|t}. \quad (7)$$

and

$$\boldsymbol{\Sigma}_{t+1|t} = \boldsymbol{F}_t\boldsymbol{\Sigma}_{t|t}\boldsymbol{F}_t^T + \boldsymbol{G}_t\boldsymbol{Q}_t\boldsymbol{G}_t^T. \quad (8)$$

Eq.(7) can also be written as

$$\hat{\boldsymbol{\alpha}}_{t+1|t} = \boldsymbol{F}_t\hat{\boldsymbol{\alpha}}_{t|t-1} + \boldsymbol{F}_t\boldsymbol{\Sigma}_{t|t-1}\boldsymbol{H}_t(\boldsymbol{H}_t^T\boldsymbol{\Sigma}_{t|t-1}\boldsymbol{H}_t + R_t)^{-1}\left(Y_t - \boldsymbol{H}_t^T\hat{\boldsymbol{\alpha}}_{t|t-1}\right), \quad (9)$$

which implies that the time update rules for each forecast of state are weighted average of previous forecast $\hat{\boldsymbol{\alpha}}_{t|t-1}$ and forecast error $\left(Y_t - \boldsymbol{H}_t^T\hat{\boldsymbol{\alpha}}_{t|t-1}\right)$. After obtaining $\hat{\boldsymbol{\alpha}}_{t|t-1}$, one may predict Y_t by the optimal predictor $\hat{Y}_{t|t-1}$, where

$$\hat{Y}_{t|t-1} = \boldsymbol{H}_t^T\hat{\boldsymbol{\alpha}}_{t|t-1} \quad (10)$$

and the conditional error variance due to predictor $\hat{Y}_{t|t-1}$ is

$$\boldsymbol{H}_t^T\boldsymbol{\Sigma}_{t|t-1}\boldsymbol{H}_t + R_t \quad (11)$$

An excellent description of this methodology is given in Durbin and Koopman (2001).

2.2 Nonlinear Least Squares Support Vector Machine (LSSVM) Model

Suppose training data set is $D = \{x_i, y_i\}_{i=1}^N$, where $x_i \in R^n$ is the i^{th} input data, $y_i \in R$ is the i^{th} target data and N corresponds to size of training data. In the primal weight space, Nonlinear LSSVM function for regression is given by

$$y(x) = w^T\varphi(x) + b. \quad (12)$$

where $\varphi(.): R^n \to R^{n_h}$ is a nonlinear function, which maps input space into a higher dimensional feature space, $w \in R^{n_h}$ is weight vector and b is a bias term. Nonlinear LSSVM for regression formulation can be described as

Objective function:

$$\min_{w,b,e} \quad J_p(w,e) = \frac{1}{2}w^Tw + \frac{\gamma}{2}\sum_{i=1}^N e_i^2 \quad (13)$$

Subject to the constraint:

$$y_i = w^T\varphi(x_i) + b + e_i, \quad i = 1,2,\ldots,N, \quad (14)$$

Where $e_i \in R$ is error. Note that the cost function J_p consists of regularization term and sum of squared fitting error in primal feature space. The relative importance of these two terms is determined by the positive real constant γ. In case of noisy data, one avoids over-fitting by taking a smaller γ value. The weight vector w can be infinite-dimensional, which makes calculation of w impossible, in general. The Lagrangian function can be constructed as

$$L(w,b,e:\alpha) = J_p(w,e) - \sum_{i=1}^N \alpha_i\left\{w^T\varphi(x_i) + b + e_i - y_i\right\}, \quad (15)$$

where $\alpha_i, i = 1,2,\ldots,N$ are Lagrangian multipliers, also called support values. The Karush-Kuhn-Tucker (KKT) conditions for optimality are given by partially differentiating L:

$$\begin{cases} \dfrac{\partial L}{\partial w} = 0 \;\rightarrow\; w = \sum_{i=1}^N \alpha_i\varphi(x_i) \\ \dfrac{\partial L}{\partial b} = 0 \;\rightarrow\; \sum_{i=1}^N \alpha_i = 0, i = 1,2,\ldots,N \\ \dfrac{\partial L}{\partial e_i} = 0 \;\rightarrow\; \alpha_i = \gamma e_i,\ i = 1,2,\ldots,N \\ \dfrac{\partial L}{\partial \alpha_i} = 0 \;\rightarrow\; w^T\varphi(x_i) + b + e_i - y_i = 0. \end{cases} \quad (16)$$

After eliminating w and e_i, optimization problem can be transferred into the following linear solution system in dual space:

$$Solve\ for\ \alpha, b$$

$$\begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 & 1_v^T \\ 1_v & \Omega + \dfrac{1}{\gamma}I \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ y \end{bmatrix}, \quad (17)$$

where

$y = [y_1, y_2, \ldots, y_N]^T, 1_v = [1,1,\ldots,1]^T, \alpha = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T$ and $\Omega = \{\Omega_{ij}\}$ is given by

$$\Omega_{ij} = \varphi(x_i)^T\varphi(x_i) = k(x_i, x_j), i = 1,2,\ldots,N, \quad (18)$$

where $k(x_i, x_j)$ is called the Inner-product kernel function. The value of the kernel equals the inner product of $\varphi(x_i)$ and $\varphi(x_j)$, which are produced by mapping two vectors x_i and x_j into the higher dimensional feature space i.e. $K(x_i, x_j) = \varphi(x_i).\varphi(x_j)$. Furthermore, the computation of $\varphi(x_i).\varphi(x_j)$ in the higher dimensional feature space may be too complex

to perform. An advantage of Nonlinear LSSVM is that the nonlinear function $\varphi(x_i)$ is need not to be used. The computation in input space can be performed using a kernel function to yield inner products in higher dimensional feature space, avoiding having to perform a mapping $\varphi(x_i)$. Thus, Nonlinear LSSVM for regression is obtained as

$$y(x) = w^T \varphi(x) + b = \sum_{i=1}^{N} \alpha_i k(x_i, x_j) + b, \quad (19)$$

Where α_i and b are solutions of the linear system.

For univariate time-series forecasting problem, dimension of input vectors are the past lagged observations. Nonlinear LSSVM model in fact performs a nonlinear functional mapping from past observations to future value.

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-p}) + \omega_t, \quad (20)$$

where y_t is output value, p is the dimension of input vectors (i.e. number of lagged observations) and ω_t is error term. However, in practice, the choice of dimension of input vectors (i.e. number of lagged observations) is difficult; often trial and error method is conducted. Therefore, we have employed PSO for optimizing time lag p and model hyper-parameters.

2.3 Hybrid Models

A time-series data is considered as comprising linear and nonlinear components. Then, the hybrid model (Y_t) is also composed of these components and can be represented as:

$$Y_t = L_t + N_t, \quad (21)$$

Where L_t and N_t respectively denote the linear and nonlinear components to be estimated respectively from linear and nonlinear models. $\hat{L}_t$ is first estimated using fitted linear KF model. Then, the residual at time *t*, i.e. ε_t contains only nonlinear pattern:

$$\varepsilon_t = Y_t - \hat{L}_t. \quad (22)$$

In this article, we propose the four various hybrid models:

Model 1: LKF-NLSSVM1

This model is given by

$$\varepsilon_t = f(\varepsilon_{t-1}, \varepsilon_{t-12}, \ldots . \varepsilon_{t-p}) + \omega_t, \quad (23)$$

Where ε_t is t^{th} residual obtained from Linear KF model, ω_t is random term and $f(.)$ is the nonlinear function determined by Nonlinear LSSVM. Therefore, the combined forecast is

$$\hat{Y}_t = \hat{L}_t + \hat{N}_t, \quad (24)$$

where $\hat{N}_t$ is the forecast value of nonlinear component obtained from fitting Eq.(23).

Model 2: LKF-NLSSVM2

This model is expressed as

$$Y_t = f(Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}, \varepsilon_{t-1}) + \omega_t. \quad (25)$$

Model 3: LKF-NLSSVM 3

This model is given by

$$Y_t = f(Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}, \hat{L}_t) + \omega_t. \quad (26)$$

Model 4: LKF-NLSSVM 4

This model is expressed as

$$Y_t = f(\hat{L}_t, \varepsilon_{t-1}) + \omega_t. \quad (27)$$

2.4 Estimation of Optimal Hyper-Parameters through PSO Algorithm

Several kernel functions $k(x_i, x_j)$ are available in the literature, like Polynomial function, Gaussian kernel, and Radial-basis function (RBF). In this article, most commonly used RBF kernel function $k(x_i, x_j) = exp\{-x - x_i^2 / (2\sigma^2)$ for nonlinear regression is adopted to train Nonlinear LSSVM model. It has only one hyper-parameter that needs to be pre-determined and yields good performance under general conditions for nonlinear time-series forecasting. Training Nonlinear LSSVM with RBF kernel function is required to be optimized through two hyper-parameters, viz. (i) Regularization parameter γ, which balances the complexity and approximation accuracy of the model, and (ii) Kernel bandwidth parameter σ, which represents the variance of RBF kernel function. To this end, PSO is employed to optimize these hyper-parameters (Parsopoulos and Vrahatis, 2010).

The PSO, proposed by Kennedy and Eberhart in 1995, is a population-based stochastic optimization search method. The population is called the 'swarm' and its individuals are called the 'particles'. PSO algorithm investigates solution space using a set of particles vector that are updated from iteration to iteration. Let $A \subset R^n$ be search space and $A \to Y \subseteq R$ be objective function, then swarm is defined as a set $S = \{X_1, X_2, \ldots, X_M\}$ of M particles (candidate

solution), where M is a user-defined parameter of the algorithm. Then i^{th} particle of dimension d is defined as $X_i = (X_{i1}, X_{i2}, \ldots, X_{id})^T, i = 1, 2, \ldots M$. Each particle is a potential solution to a problem, characterized by three quantities: its velocity $(V_i) = (V_{i1}, V_{i2}, \ldots, V_{id})^T$, its current position $(X_i) = (X_{i1}, X_{i2}, \ldots, X_{id})^T$ and personal best position $(pbest_i) = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{id})^T$. Let t denotes current iteration and $gbest$ denotes its global best position achieved so far by any of its particles. Initially, swarm is randomly dispersed within search space, and random velocity is assigned to each particle. Particles interact with one another by sharing information to discover optimal solution. Each particle moves in the direction of its personal best position $(pbest)$ and its global best position $(gbest)$. To search optimal solution, each particle changes its velocity according to the cognitive and social parts given by

$$V_{ij}(t+1) = w(t)V_{ij}(t) + c_1 R_1 \left[pbest_{ij}(t) - X_{ij}(t) \right] + c_2 R_2 \left[gbest_j(t) - X_{ij}(t) \right], \quad (28)$$

where $i = 1, 2, \ldots M$ and $j = 1, 2, \ldots d$. However, in case of swarm explosion effect, corresponding velocity component is restricted to following closest velocity bound:

$$V_{ij}(t+1) = \begin{cases} -V_{max}, & if\ V_{ij}(t+1) < -V_{max} \\ V_{max}, & if\ V_{ij}(t+1) > V_{max}. \end{cases} \quad (29)$$

After updating its velocity, each particle moves to a new potential solution by updating its position as follows:

$$X_{ij}(t+1) = \begin{cases} X_{min}, & if\ X_{ij}(t+1) < X_{min} \\ X_{ij}(t) + \beta V_{ij}(t+1), & if\ X_{min} \le X_{ij}(t+1) \le X_{max} \\ X_{max}, & if\ X_{ij}(t+1) > X_{max} \end{cases}; i = 1, 2, \ldots M; j = 1, 2, \ldots d. \quad (30)$$

In Eqs.(28) and (30), V_{ij}, X_{ij} and $pbest_{ij}$ are respectively velocity, position and personal best position of particle i on the j^{th} dimension and $gbest_j$ is the j^{th} dimension $gbest$ position among all particles at iteration t. R_1 and R_2 are random values, which are mutually independent and uniformly distributed over [0, 1], β is a constraint factor used to control velocity weight, whose value is usually set equal to 1. Positive constants c_1 and c_2 are usually called the "acceleration factors". Factor c_1 is sometimes referred to as "cognitive" parameter, while c_1 is referred to as "social" parameter. Inertia weight at iteration t is $w(t)$ and is used to balance global exploration and local exploitation. This can be determined by:

$$w(t) = w_{up} - \frac{(w_{up} - w_{low})t}{T_{max}} \quad (31)$$

where t is current iteration number, w_{up} and w_{low} are desirable lower and upper limits of w and T_{max} is maximum number of iterations.

Process of optimizing hyper-parameters of Nonlinear LSSVM model by PSO is described in following steps:

Step 1*: Read of Data and Initialization and PSO parameter setting*

When handling time-series data, it is important to realize that the entries are inherently indexed on time. This is crucial as mixing of data (as in *k*-fold cross-validation) may lead to completely different realizations of the underlying phenomenon. The validation technique respecting time-series order can be adopted. Here, training data is divided into two subsets, viz. (i) training set, which is used for actual training of the model, and (ii) validation set, which is used to guide the search (tuning) for optimal hyper-parameter values. Once these are found, machine's generalization performance can be evaluated on a test set.

Nonlinear LS-SVM model hyper-parameters, viz. Regularization parameter γ, Kernel bandwidth parameter σ are directly coded with real values within a given search space to randomly generate M number of initial particles of swarm set S. Search space of hyper-parameters γ and σ are respectively restricted to ranges of $[\gamma_{min}, \gamma_{max}]$, $[\gamma_{min}, \gamma_{max}]$ and $[\sigma_{min}, \sigma_{max}]$. Also randomly generate initial particles velocity (V_{ij}) restricted to the range of $[-V_{max}, V_{max}]$.

Step 2*: Fitness function*

The fitness of training data set is easy to calculate, but is prone to over-fitting. In this article, validation technique is adopted to guide (tune) the search of optimal hyper-parameter values by PSO. In this way, given a specific particle, whose current position (X_i) defines a set of hyper-parameters $\{\gamma, \sigma\}$ along with the training and validation sets at hand are used to build the model. In validation technique, first Nonlinear LSSVM function is built for the set of hyper-parameters $\{\gamma, \sigma\}$ considering the training set. Thus, the trained model is used to predict outputs from input values of the

validation set. The performance of trained model is assessed by Root Mean Square Error (RMSE) criterion, which gives an estimate of the expected generalization error for training data set and is given by:

$$RMSE\{\gamma, \sigma\} = \left\{ l^{-1} \left(\sum_{t=1}^{l} \left(y_t - \hat{y}_t \right)^2 \right) \right\}^{1/2}, \quad (32)$$

where y_t and $\hat{y}_t$ are respectively actual and predicted values, and l is number of samples in validation set. At this step, the coupling between PSO and Nonlinear LSSVM takes place.

Step 3*: Train Nonlinear LS-SVM model and evaluate fitness value:*

For each set of hyper-parameters $\{\gamma, \sigma\}$ of initial particles, train Nonlinear LSSVM model. Evaluate fitness value defined by RMSE in Eq.(32) on validation set. The fitness evaluation phase entails the update of particles' personal best position $(pbest)$. If a particle's current position X_i results in a smaller RMSE, then its best position becomes X_i and the calculated fitness value is stored. Then, set the personal best position $(pbest)$ achieved so far by each particle. After fitness evaluation of all particles takes place, set the global best position *gbest* achieved so far by any of its particles according to fitness value.

Step 4*: Update of Particles' velocities and Positions*

Now set the iteration number (t) from one to maximum number of iterations and update initial weight $w(t)$ generation by generation according to Eq.(31). Compute and update velocity of each particle using Eq. (28). Additionally, velocities are bounded to the definition ranges of variables using Eq. (29) to avoid particles going too far from the feasible search space. Then compute and update position of each particle using Eq. (30) subject to constraints in Eq. (30) for each hyper-parameter to restrict particle positions to within a search space range. After updating velocity and position of each particle, find and update the personal best position $(pbest)$ achieved so far by each particle and global best position $(gbest)$ achieved so far by any of its particles according to minimum fitness value.

Step 5: *Termination*

Repeat search process from steps 2 to 4 until stop conditions, such as maximum iteration, are met. Finally, optimal hyper-parameters are utilized to build Nonlinear LSSVM model on training data set and testing data set are used to validate prediction performance of the trained model. To perform the above tasks, LS-SVMLab, Version 1.8 toolbox, developed by De Brabanter *et al.* (2011) for MATLAB platform, is employed. Relevant Computer program for estimating optimal hyper-parameters of Nonlinear LS-SVM model by PSO is developed in MATLAB 2012a function (m file) and the same is appended as ANNEXURE-I.

3. RESULTS AND DISCUSSION

As an illustration, all-India monthly rainfall (in mm) time-series data for the period January 1991 to December 2011, obtained from the website (www.tropmet.res.in) of the Indian Institute of Tropical Meteorology, Pune, India are considered. The data points for the period January 1991 to December 2010 are used as training data set and remaining data points for the period January 2011 to December 2011 are used as testing data set.

First, Linear KF is employed to recursively estimate linear component. The particular form of m*easurement equation* and s*tate transition equation are employed,* which are respectively given by $Y_t = \hat{\mu} + \boldsymbol{H}_t^T \boldsymbol{\alpha}_t + N_t$ and $\boldsymbol{\alpha}_t = \boldsymbol{F}_t \boldsymbol{\alpha}_t + \boldsymbol{K}_t$, where $N_t \sim (0, R)$ and $\boldsymbol{K}_t \sim (0, Q)$, $\boldsymbol{H}_t^T = (0100\ldots0)_{1\times13}$ and

$$\boldsymbol{F}_t = \begin{pmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ \mu & \alpha_1 & \alpha_2 & \alpha_3 & \ldots & \alpha_{12} \\ 0 & 1 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{pmatrix}_{13\times13},$$

which is computed as

$$\hat{\boldsymbol{F}}_t = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 87.350 & 0.31 & 0.17 & 0.36 & 0.76 & 0.81 & 0.11 & 0.39 & 0.91 & 0.81 & 0.19 & 0.22 & 0.67 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The computed value of $\hat{\mu}, \hat{R},$ and $\hat{Q}$ are respectively given by 87.31, 1098.7 and 897.6. Then these values are used to compute the linear component estimated and generate residuals. Fitted values of Linear KF and its residuals are used to develop the four hybrid models. Nonlinear LSSVM and hybrid models with RBF kernel function were trained using LSSVMLab toolbox in MATLAB software package. Note that, before training Nonlinear LSSVM and hybrid models, each data point is normalized to zero mean and unit variance so as to improve generalization power of these

models. In the training phase, first training data consists of 20 years data points corresponding to the period January 1991 to December 2010 are divided into two subsets viz. i) training set (data points corresponding to the period January 1991 to December 2008), which are used for training Nonlinear and hybrid models, and ii) validation set (24 data points corresponding to the period January 2008 to December 2010), which are used for tuning optimal hyper-parameters. Next, PSO algorithm is employed to estimate hyper-parameters $\{\gamma,\sigma\}$,and time-lag of Nonlinear LSSVM and hybrid models. Through initial experiment, the parameters of PSO technique for Nonlinear LSSVM model are set as follows: 100 initial particles, so that it is enough to cover the search space within the limited iterations based on experimental runs, 300 maximum iterations, inertia weight is initially set as 0.9 and reduced to 0.1 linearly according to Equation (31) and c_1=c_2=2.05. The searching range of Nonlinear LSSVM and hybrid models hyper-parameters γ and σ were respectively set in the range [0.001, 100] and [0.001,10] and their velocity bounds were respectively set to be in the range [-49.99, 49.99] and [-4.99,4.99], time-lag of above models are restricted to 1 to 12 maximum lags. Nonlinear LSSVM and hybrid models are trained using PSO to evaluate fitness value defined by *RMSE* in Equation (32) on validation set. The set of hyper-parameters and time-lag having the minimum *RMSE* are selected as the optimal hyper-parameters and time-lag. The values obtained are reported in Table 1.

Table 1. Optimal hyper-parameters estimated by PSO techniques for various models

Models	γ	σ	p	RMSE
Nonlinear LSSVM	9.61	0.91	5	19.53
LKF-NLSSVM1	17.81	5.76	8	36.80
LKF-NLSSVM2	7.47	1.35	8	19.26
LKF-NLSSVM3	36.21	0.71	6	19.23
LKF-NLSSVM4	1.37	0.82	-	49.02

These optimal hyper-parameters and time lag values were utilized to build Nonlinear LSSVM and hybrid models on whole training data set and forecasting accuracy of these models was examined using testing data. Further, prediction performances of the hybrid models were compared along with Linear KF and Nonlinear LSSVM models on the basis of Root Mean Square Error (RMSE) and Mean Absolute Percent Error (MAPE) criteria, given respectively as

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(y_t - \hat{y}_t\right)^2}, \tag{33}$$

$$\text{MAPE} = \frac{1}{\text{n}}\sum_{\text{t=1}}^{\text{n}}\left|\frac{\text{y}_\text{t} - \hat{\text{y}}_\text{t}}{\text{y}_\text{t}}\right| * 100 \tag{34}$$

where y_t is actual value, $\hat{y}_t$ is predicted value, and n is number of observations. RMSE and MAPE values for fitted models were computed on training dataset and are reported in Table 2.

It may be noted from Table 2 that LKF-NLSSVM3 model has performed the best followed by LKF-NLSSVM2 model.

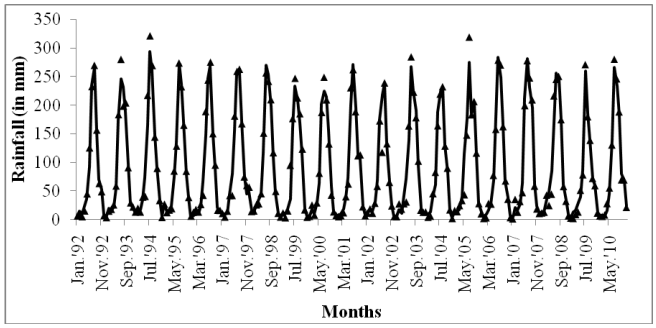

Fig. 1. Fitted LKF-NLSSVM3 model along with training data points

Further, one-step ahead forecasts along with actual values during the period January 2011 to December 2011 for all the six models were computed and are reported in Table 3. The RMSE and MAPE values were also computed for the above models on the testing dataset, and the results are reported in Table 3. On test data also, a perusal indicates that LKF-NLSSVM3 model has performed the best followed by LKF-NLSSVM2 model. To get a visual insight, fitted LKF-NLSSVM3 model along with data are exhibited in Fig. 1. Evidently, the fit is seen to be extremely good.

To sum up, hybrid model, viz. LKF-NLSSVM3 using PSO is found to be best for both training as well as testing data.

Table 2. Comparison of forecasting performance of various models on training dataset

Models Criteria	Linear KF	Nonlinear LSSVM	LKF-NLSSVM1	LKF-NLSSVM2	LKF-NLSSVM3	LKF-NLSSVM4
RMSE	50.75	22.58	24.46	17.30	15.47	41.35
MAPE	158.90	46.12	49.11	38.17	28.74	104.48

Table 3. One-step ahead forecasts for all-India monthly rainfall (in mm)

Months	Actual values	Models					
		Linear KF	Nonlinear LSSVM	LKF-NLSSVM1	LKF-NLSSVM2	LKF-NLSSVM3	LKF-NLSSVM4
Jan., 2011	2.0	19.83	6.55	25.65	5.93	7.86	16.75
Feb., 2011	13.6	24.53	15.74	28.93	16.16	13.05	10.57
Mar., 2011	9.9	49.01	26.38	5.34	20.23	21.98	35.16
Apr., 2011	32.8	38.09	24.28	1.63	18.13	17.74	30.96
May, 2011	47.8	67.20	72.48	51.94	43.19	30.72	72.01
Jun., 2011	192.3	71.96	187.90	107.77	145.41	160.87	98.46
Jul., 2011	231.9	234.00	261.13	276.24	272.41	265.34	257.05
Aug., 2011	268.4	199.08	235.87	241.15	252.61	270.67	201.27
Sep., 2011	188.3	219.77	172.87	171.81	148.58	183.75	228.15
Oct., 2011	37.8	104.63	82.04	80.04	71.78	34.97	163.39
Nov., 2011	22.2	27.71	21.29	55.56	45.80	31.59	31.90
Dec., 2011	4.8	38.93	5.69	18.10	10.26	6.96	17.49
Accuracy Criteria	RMSE	48.57	20.63	35.61	25.45	15.63	52.45
	MAPE	203.51	55.18	173.00	62.74	52.55	151.43

4. CONCLUSION

In this article, linear KF and nonlinear LSSVM methodologies are described. Also, hybrid models are developed by combining these models. Further, PSO technique, which avoids over-fitting or under-fitting, is employed to optimize the hyper-parameters. As an illustration, all-India monthly rainfall time-series data are considered. It is found that hybrid models achieved greater accuracy than linear KF and nonlinear LSSVM models. Specifically, LKF-NLSSVM3 model performed best among the six models for data under consideration. Hence, in order to exploit the advantages of these kind of hybrid models developed using linear KF and nonlinear LSSVM model could be applied to other various data sets generated in agriculture for forecasting namely, forecasting of price of agriculture commodity, forecasting of production data, forecasting import and export of agricultural commodities, forecasting regional weather data, etc…

REFERENCES

Chattopadhyay, S. and Chattopadhyay, G. (2013). A supervised neural network model for predicting average summer monsoon rainfall in India. *Journal of the Indian Society of Agricultural Statistics,* **67**, 43-49.

Chen, K.Y. and Wang, C.H. (2007a). A hybrid SARIMA and support vector machine in forecasting the production value of the machinery industry in Taiwan. *Expert Systems with Applications,* **32**, 254-264.

Chen, K.Y. and Wang, C.H. (2007b). Support vector regression with genetic algorithm in forecasting tourism demand. *Tourism Management,* **28**, 215-226.

De Brabanter, K., Karsmakers, P., Ojeda, F., Alzate, C., De Brabanter, J., Pelckmans, K., De Moor, B., Vandewalle, J. and Suykens, J. A. K. (2011). LS-SVMlab: Matlab toolbox for Least Squares Support Vector Machines. (*http://www.esat.kuleuven.ac.be/sista/LS-SVMlab*).

Durbin, J. and Koopman, S.J. (2001). *Time-Series Analysis by State Space Methods*. New York: Oxford University Press.

Gestel, T.V., Suykens, J.A.K., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., Moor, B.D. and Vandewalle, J. (2004). Benchmarking least squares support vector machine classifiers. *Machine Learning,* **54** (1), 5–32.

Ghosh, Himadri, Gurung, Bishal and Prajneshu (2011). Methodology for combining linear and nonlinear time-series models for cyclical data. *Journal of Indian Society of Agricultural Statistics*, **65**, 249-56.

Gurung, B., Singh, K.N., Paul, R., Panwar, S., Gurung, B. and Lepcha, L. (2017). An alternative method for forecasting price volatility by combining models, *Communications in Statistics - Simulation and Computation*, **46**(6), 4627-4636.

Khashei, M. and Bijari, M. (2011). A novel hybridization of artificial neural networks and ARIMA models for time-series forecasting. *Applied Soft Computing*, **11**, 2664-2675.

Lama, A., Singh, K.N., Singh, H., Shekhawat, R., Mishra, P. and Gurung, B. (2022). Forecasting monthly rainfall of Sub-Himalayan region of India using parametric and non-parametric modelling approaches. *Modelling Earth Systems and Environment*, **8**, 837-845.

Narayanan, P., Basistha, A., Sarkar, S. and Sachdeva, K. (2013). Trend analysis and ARIMA modelling of pre-monsoon rainfall data for Western India. *ComptesRendus Geoscience,* **345**, 22-27.

Pai, P.F. and Hong, W.C. (2005). Support vector machines with simulated annealing algorithms in electricity load forecasting. *Energy Conversion and Management,* **46**, 2669-2688.

Parsopoulos, K.E. and Vrahatis, M.N. (2010). *Particle Swarm Optimization and Intelligence: Advances and Application.* New York: Information Science Publishing (IGI Global).

Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B. and Vandewalle, J. (2002). *Least Squares Support Vector Machines*. Singapore: World Scientific.

Vapnik, V. (2000). *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.

Yanga, Z., Gub, X.S., Lianga, X.Y. and Linga, L.C. (2011). Genetic algorithm-least squares support vector regression-based predicting

and optimizing model on carbon fiber composite integrated conductivity. *Materials and Design*, **31**,1042–1049.

Zhang, G.P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, **50**,159-175.

Zhou, X., and Ma, Y. (2013). A study on SMO algorithm for solving ε-SVR with non-PSD kernels. *Communications in Statistics – Simulation and Computation*, **42**, 2175-2196.

ANNEXURE-1

MATLAB PROGRAM FOR ESTIMATING OPTIMAL HYPER-PARAMETERS OF THE NONLINEAR LS-SVM MODEL USING PSO ALGORITHM FOR TIME-SERIES FORECASTING PROBLEM

```
function [Bestgam, Bestsig, ValRMSE,fittness_
iter] = PSOLSSVMforTS(Xtr,Ytr,Xval,Yval,
gamma,sigma,population,cvalue,max_iteration)

%% Syntex
%>> [Bestgam, Bestsig, ValMSE,fittness_
iter]=PSOLSSVMforTS(Xtr,Ytr,Xval,Yval,
gamma,sigma,population,cvalue,max_iteration);
%% Output:
% Bestgam:Scalar value of Optimal regularization
parameter (gamma) value
% Bestsig2 :Scalar value of Optimal kernel
parameter (sigma) value
% ValRMSE:Estimated cost of the optimal
hyperparameters
% fittness_iter : Estimated cost over itteration in
PSO process
%% Inputs:
% Xtr : M x d matrix of input training data
% Ytr : M x 1 vector of outputs training data
% Xval : N-M*d Matrix of input validation data
% Yval : N-M*1 vector of output validation data
% gamma : 1 x 2 vector of hepreparametergama
range default [0.001,100]
% sigma : 1 x 2 vector of hepreparameter sigma
range default [0.001 100]
% Population : Size of poluation (swarm) scalar or
number of particles default 100
% Cvalue : 1 x 2 vector of c1 and c2 value default
c1=2.05; c2=2.05;
% max_iteration : maximum number of iterations
default 200
%%

%default values
ifnargin==1;
gamma=[0.001,100]; %default value of gamma
range
sig=[0.001, 100]; %default value of sigma range
max_iter=200; %default value of Maximum
iteration
pop=100; %default value of population (swarm)
or number of particles
part_dim=2; % default value ofparticles or
dimension of particles
c1=2.05; c2=2.05; %default value of acceleration
constant
else
gamma=gamma;
sig=sigma;
pop=population;
part_dim=2;
c1=cvalue(1); c2=cvalue(2);
max_iter=max_iteration;
end

%bound on velocity
Vgammax=(gamma(2)-gamma(1))/2; %velocity
maximum bound on gamma
Vgammin=-Vgammax; %velocity minimum
bound on gamma
Vsigmax=(sig(2)-sig(1))/2; %velocity maximum
bound on sigma
Vsigmin=-Vsigmax; %velocity minimum bound
on sigma

% preallocation of particle posistion, velcity, pbest
and fitness
X=zeros(pop,part_dim); %preallocation of X
particle position
V=zeros(pop,part_dim); %preallocation of
velocity
Pbest_position=zeros(pop,part_dim);
%preallocation of personal best position
fitness=zeros(pop,1); % prealloation of global
fitness function (MSE)value

% initialize particle position and velocity and
evaluate initial fitness
fori=1:pop
X(i,1)=gamma(1)+(gamma(2)-
gamma(1))*rand(1);
X(i,2) = sig(1)+(sig(2)-sig(1))*rand(1);
V(i,1)=Vgammin+(Vgammax-
Vgammin)*rand(1);
```

```
V(i,2) = Vsigmin+(Vsigmax-Vsigmin)*rand(1);
%gamma and sig2 value for LSSVM model
gam=X(i,1); sig2=X(i,2);
% training model to estimate fitness value of fitness function MAPE
[alpha,b] = trainlssvm({Xtr,Ytr,'f',gam,sig2,'RBF_ kernel','preprocess'});
predicted = simlssvm({Xtr,Ytr,'f',gam,sig2,'RBF_ kernel','preprocess'},{alpha,b},Xval);
RMSE=sqrt(mean((Yval-predicted).^2));
fitness(i,1)=RMSE;
end

% particles global fitness and its index value
[Gbest_fit, gbestfitindex]=min(fitness);
Gbest_position=X(gbestfitindex,:);% particles globalbest position
Pbest_position=X; %particle personal best position
Pbest_fit=fitness; %particle personal best fitness

fittness_iter=zeros(max_iter,1);
%update particle velocity position
for t=1:max_iter %t is iteration number
fori=1:pop
w_up=0.9; w_low=0.1;
w=w_up-((w_up-w_low)*t)/max_iter;
%update particle velocity
V(i,:)=w*V(i,:)+c1*rand*(Pbest_position(i,:)-X(i,:))+c2*rand*(Gbest_position-X(i,:));

% constrained to velocity upper and lower bound on parameter gamma
if V(i,1)>Vgammax
V(i,1)= Vgammax;
elseif V(i,1)<Vgammin;
V(i,1)= Vgammin;
end

% constrained to particle velocity upper and lower bound on parameter sigma
if V(i,2)>Vsigmax
V(i,2)= Vsigmax;
elseif V(i,2)<Vsigmin;
V(i,2)= Vsigmin;
end

%update particle current position

X(i,:)=X(i,:)+V(i,:);

% constrained to particle current position upper and lower bound on parameter gamma
if X(i,1)>gamma(2)
X(i,1)=gamma(2);
elseif X(i,1)<gamma(1)
X(i,1)=gamma(1);
end

% constrained to particle current position upper and lower bound on parameter sigma
if X(i,2)>sig(2)
X(i,2)=sig(2);
elseif X(i,2)<sig(1)
X(i,2)=sig(1);
end

% evaluation of fitness function for update particles
gam=X(i,1); sig2=X(i,2);
[alpha,b] = trainlssvm({Xtr,Ytr,'f',gam,sig2,'RBF_ kernel','preprocess'});
predicted = simlssvm({Xtr,Ytr,'f',gam,sig2,'RBF_ kernel','preprocess'},{alpha,b},Xval);
RMSE=sqrt(mean((Yval-predicted).^2));
fitness(i,1)=RMSE;
end

% update particles fitness
[current_gbest_fit, current_index]=min(fitness);
current_pbest_fit=fitness;

% update global best fitness and position of particles
ifcurrent_gbest_fit<= Gbest_fit
Gbest_fit=current_gbest_fit;
Gbest_position=X(current_index,:);
end

% update personal best fitness and position of particles
Pbest_index=find(current_pbest_fit<=Pbest_fit);
Pbest_position(Pbest_index,:)=X(Pbest_index,:);
Pbest_fit(Pbest_index)=current_pbest_fit(Pbest_ index);
fittness_iter(t,1)=Gbest_fit;
end
Bestgam=Gbest_position(1);
Bestsig=Gbest_position(2);
ValRMSE=Gbest_fit;
fittness_iter=fittness_iter;
end
```