Chapter 2
Modeling for Energy Demand Forecasting

As mentioned in Chap. 1, the electric load forecasting methods can be classified in three categories [1–12]:

- Traditional approaches, including Box–Jenkins autoregressive integrated moving average (ARIMA) model, autoregressive and moving average with exogenous variables (ARMAX) model, seasonal autoregressive integrated moving average (SARIMA) model, exponential smoothing models [including Holt–Winters model (HW) and seasonal Holt and Winters' linear exponential smoothing (SHW)], state space/Kalman filtering model, and linear regression model
- Artificial intelligent approaches, including knowledge-based expert system (KBES) model, artificial neural networks (ANNs) model, and fuzzy inference system model
- Support vector regression (SVR) model and its related hybrid/combined models

These models are classified on the basis of the forecasting technological development tendency, evolved from mathematical relationship model (e.g., statistics-based model) to application of artificial intelligent model (e.g., ANNs model) and eventually to hybridizing statistical model and artificial intelligent model (e.g., SVR model). Of course, the classifications are not unique, and the classification based on the technological evolution is not always suitable for another. However, based on this classification, interested readers can be inspired to propose another new model to receive more accurate electric load forecasting performance. Additionally, each model has its outstanding advantages compared with other models due to its theoretical innovation while it has been proposed and also has its embedded theoretical limitations; thus, it always has the potential to be improved by hybridizing or combining with other novel approaches.

This book is focused on SVR model, SVR with hybrid evolutionary algorithms, and SVR with combined mechanisms; therefore, to be based on the same comparison conditions and easily to receive full comparison results, only ARIMA,

W.-C. Hong, *Intelligent Energy Demand Forecasting*, Lecture Notes in Energy 10,
DOI 10.1007/978-1-4471-4968-2_2,

SARIMA, HW, SHW, GRNN (general regression neural network), BPNN (back-propagation neural network), and SVR models are introduced in the following subsections, whereas the state space/Kalman filtering, linear regression, and KBES models are beyond the scope of this book.

2.1 Autoregressive Integrated Moving Average Model

Introduced by Box and Jenkins [13], the ARIMA model has been one of the most popular approaches in forecasting. The ARIMA model is composed of three partitions: the autoregressive (AR), the moving average (MA), and the differencing process (also called integrated processes). In the AR process, the current value of electric load is often expressed as linear combination of previous actual electric load values and with a random noise. The order of AR process is determined by the oldest previous electric load value that is regressed with the same series itself. In the MA process, it expresses a white noise error series of linear combination in terms of current against previous (unobserved) white noise error term. The order of MA process is determined by the oldest previous value. The AR and MA processes are combined to be the famous electric load forecasting model, autoregressive moving average (ARMA) process. In the ARMA process, the order is determined by the same method as mentioned in AR and MA processes [14].

The AR, MA, or ARMA models are often viewed as stationary processes, that is, their means and covariances are stationary with respect to time. Therefore, while the process is nonstationary, it is necessarily transformed to a stationary series before conducting their modeling processes. Differencing process is employed to transform a nonstationary series into a stationary one. The order of a differencing process is the number of times of differenced before achieving stationarity. Differencing processes for AR, MA, or ARMA models are also the so-called integrated processes and are named as ARI, IMA, and ARIMA, respectively.

In an ARIMA model, the future value of a variable is supposed to be a linear combination of past values and past errors, expressed as Eq. (2.1):

$$y_t = \theta_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \cdots \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}, \tag{2.1}$$

where y_t is the actual value and ε_t is the random error at time t, ϕ_i and θ_j are the coefficients, and p and q are integers and often referred to as autoregressive and moving average polynomials, respectively. In addition, the difference (∇) is used to solve the nonstationary problem and defined as Eq. (2.2):

$$\nabla^d y_t = \nabla^{d-1} y_t - \nabla^{d-1} y_{t-1}. \tag{2.2}$$

Basically, three phases are included in an ARIMA model: model identification, parameter estimation, and diagnostic checking. Furthermore, the backward shift operator, B, is defined as Eqs. (2.3) and (2.4):

$$B^1 y_t = y_{t-1}, B^2 y_t = y_{t-2}, \ldots, B^p y_t = y_{t-p}, \tag{2.3}$$

$$B^1 \varepsilon_t = \varepsilon_{t-1}, B^2 \varepsilon_t = \varepsilon_{t-2}, \ldots, B^p \varepsilon_t = \varepsilon_{t-p}. \tag{2.4}$$

Then $\phi_p(B)$ and $\theta_q(B)$ can be written as Eqs. (2.5) and (2.6), respectively,

$$\phi_p(B) = 1 - \phi_1 B^1 - \phi_2 B^2 - \cdots\cdots - \phi_p B^p, \tag{2.5}$$

$$\theta_1(B) = 1 - \theta_1 B^1 - \theta_2 B^2 - \cdots\cdots - \theta_q B^q. \tag{2.6}$$

Hence, Eq. (2.1) can be rewritten as Eq. (2.7):

$$\phi_p(B)\nabla^d y_t = C_0 + \theta_q(B)\varepsilon_t. \tag{2.7}$$

Equation (2.7) is denoted as ARIMA(p,d,q) with nonzero constant, C_0. For example, the ARIMA(2,2,1) model can be represented as Eq. (2.8):

$$\phi_2(B)\nabla^2 y_t = C_0 + \theta_1(B)\varepsilon_t. \tag{2.8}$$

In general, the values of p, d, and q need to be estimated by autocorrelation function (ACF) and partial autocorrelation function (PACF) of the differenced series.

2.2 Seasonal Autoregressive Integrated Moving Average Model

For a special-period time series, a seasonal or cyclic component should be considered in ARIMA modeling process. This additional process is well known as the seasonal process, and its abbreviation is used as SARIMA [15]. The SARIMA process is often referred to as the SARIMA(p,d,q)(P,D,Q)$_S$ model. Similar to the ARIMA model, the forecasting values are assumed to be a linear combination of past values and past errors. A time series $\{X_t\}$ is a SARIMA process with seasonal period length S if d and D are nonnegative integers and if the differenced series $W_t = (1 - B)^d(1 - B^S)^D X_t$ is a stationary autoregressive moving average process. In symbolic terms, the model can be written as Eq. (2.9):

$$\phi_p(B)\Phi_P(B^S)W_t = \theta_q(B)\Theta_Q(B^S)\varepsilon_t, \quad t = 1, 2, \ldots, N, \tag{2.9}$$

where N is the number of observations up to time t, B is the backshift operator defined by $B^a W_t = W_{t-a}$, $\phi_p(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$ is called a regular (non-seasonal) autoregressive operator of order p, $\Phi_P(B^S) = 1 - \Phi_1 B^S - \cdots - \Phi_P B^{PS}$ is a seasonal autoregressive operator of order P, $\theta_q(B) = 1 - \theta_1 B - \cdots - \theta_q B^q$ is a regular moving average operator of order q, $\Theta_Q(B^S) = 1 - \Theta_1 B^S - \cdots - \Theta_Q B^{QS}$ is a seasonal moving average operator of order Q, and ε_t is identically and independently distributed as normal random variables with mean zero, variance σ^2, and cov $(\varepsilon_t, \varepsilon_{t-k}) = 0, \forall k \neq 0$.

In the definition above, the parameters p and q represent the autoregressive and moving average order, respectively, and the parameters P and Q represent the autoregressive and moving average order at the model's seasonal period length, S, respectively. The parameters d and D represent the order of ordinary and seasonal differencing, respectively.

Basically, when fitting a SARIMA model to data, the first task is to estimate values of d and D, the orders of differencing needed to make the series stationary and to remove most of the seasonality. The values of p, P, q, and Q then need to be estimated by the autocorrelation function (ACF) and partial autocorrelation function (PACF) of the differenced series. Other model parameters may be estimated by suitable iterative procedures.

2.3 Holt–Winters Model

The Holt–Winters (HW) model is proposed by Holt [16] and Winter [17]. HW model is an extension of exponentially weighted moving average procedure. The exponentially weighted moving average approach forecasts future values based on past observations and places more weight on the recent observations. HW model smoothes the trend values separately with two smoothing coefficients (with values between 0 and 1) and incorporates an explicit linear trend in the forecast. The approach of Holt–Winter linear exponential smoothing is shown as Eqs. (2.10)–(2.12):

$$s_t = \alpha a_t + (1 - \alpha)(s_{t-1} + b_{t-1}), \tag{2.10}$$

$$b_t = \beta(s_t - s_{t-1}) + (1 - \beta) b_{t-1}, \tag{2.11}$$

$$f_t = s_t + i b_t, \tag{2.12}$$

where a_t is the actual value at time t, s_t is the smoothed estimate at time t, b_t is the trend value at time t, α is the level smoothing coefficient, and β is the trend smoothing coefficient.

Equation (2.10) lets the actual value be smoothed in a recursive manner by weighting the current level (α) and then adjusts s_t directly for the trend of the

previous period, b_{t-1}, by adding it to the last smoothed value, s_{t-1}. This helps to eliminate the lag and brings s_t to the approximate base of the current data value. Equation (2.11) updates the trend, which is expressed as the difference between the last two smoothed values. It modifies the trend by smoothing with β in the last period ($s_t - s_{t-1}$) and adding that to the previous estimate of the trend multiplied by $(1 - \beta)$. Equation (2.12) is used to forecast ahead. The trend, b_t, is multiplied by the number of periods ahead to be forecast, i, and added to the base value, s_t. The forecast error (e_t) is defined as the actual value minus the forecast (fitted) value for time period t, which is shown in Eq. (2.13):

$$e_t = a_t - f_t. \tag{2.13}$$

The forecast error is assumed to be an independent random variable with zero mean and constant variance. Values of smoothing coefficients, α and β, are determined to minimize the forecast error index.

2.4 Seasonal Holt–Winters (SHW) Model

To consider the seasonal effect, the seasonal Holt and Winters' linear exponential smoothing (SHW) approach is also employed. HW model cannot be extended to accommodate additive seasonality if the magnitude of the seasonal effects does not change with the series or multiplicative seasonality if the amplitude of the seasonal pattern changes over time. Therefore, the forecast for SHW model is shown as Eqs. (2.14)–(2.17):

$$s_t = \alpha \frac{a_t}{I_{t-L}} + (1 - \alpha)(s_{t-1} + b_{t-1}), \tag{2.14}$$

$$b_t = \beta(s_t - s_{t-1}) + (1 - \beta)b_{t-1}, \tag{2.15}$$

$$I_t = \gamma \frac{a_t}{s_t} + (1 - \gamma)I_{t-L}, \tag{2.16}$$

$$f_t = (s_t + ib_t)I_{t-L+i}, \tag{2.17}$$

where L is the length of seasonality, I is the seasonal adjustment factor, and γ is the seasonal adjustment coefficient. Equation (2.14) differs slightly from Eq. (2.15) in that the first term is divided by the seasonal number I_{t-L}; this is done to deseasonalize a_t (eliminate seasonal fluctuations from a_t). Equation (2.16) is comparable to a seasonal index that is found as a ratio of current values of the series, a_t, divided by the smoothed value for the series, s_t. If a_t is larger than s_t, the ratio will be greater than 1, else, the ratio will be less than 1. In order to smooth the randomness

of a_t, Eq. (2.17) weights the newly computed seasonal factor with γ and the most recent seasonal number corresponding to the same season with $(1-\gamma)$.

2.5 General Regression Neural Network Model

The GRNN model, proposed by Specht [18], can approximate any arbitrary function from historical data. The foundation of GRNN operation is based on the theory of kernel regression. The procedure of the GRNN model can be equivalently represented as Eq. (2.18):

$$E[N|M] = \frac{\int_{-\infty}^{\infty} Nf(M,N)\mathrm{d}N}{\int_{-\infty}^{\infty} f(M,N)\mathrm{d}N}, \tag{2.18}$$

where N is the predicted value of GRNN, M is the input vector (M_1, M_2, …, M_n) which consists of n variables, $E[N|M]$ is the expected value of the output N given an input vector M, and $f(M, N)$ is the joint probability density function of M and N.

GRNN model primarily has four layers (Fig. 2.1). Each layer is assigned with a specific computational function when nonlinear regression function is performed. The first layer of the network is to receive information. The input neurons then feed the data to the second layer. The primary task of the second layer is to memorize the relationship between the input neuron and its proper response. Therefore, the neurons in the second layer are also called pattern neurons. A multivariate Gaussian function of θ_i is given in Eq. (2.19), and the data from the input neurons are used to compute an output θ_i by a typical pattern neuron i:

$$\theta_i = \exp\left[\frac{-(M-U_i)'(M-U_i)}{2\sigma^2}\right], \tag{2.19}$$

where U_i is a specific training vector represented by pattern neuron i and σ is the smoothing parameter. In the third layer, the neurons, namely, the summation neurons, receive the outputs of the pattern neurons. The outputs from all pattern neurons are augmented. Basically, two summations, the simple summation and the weighted summation, are conducted in the neurons of the third layer. The simple summation and the weighted summation operations can be represented as Eqs. (2.20) and (2.21), respectively:

$$S_\mathrm{s} = \sum_i \theta_i, \tag{2.20}$$

$$S_\mathrm{w} = \sum_i w_i\theta_i, \tag{2.21}$$

where w_i is the pattern neuron i connected to third layer of weights.

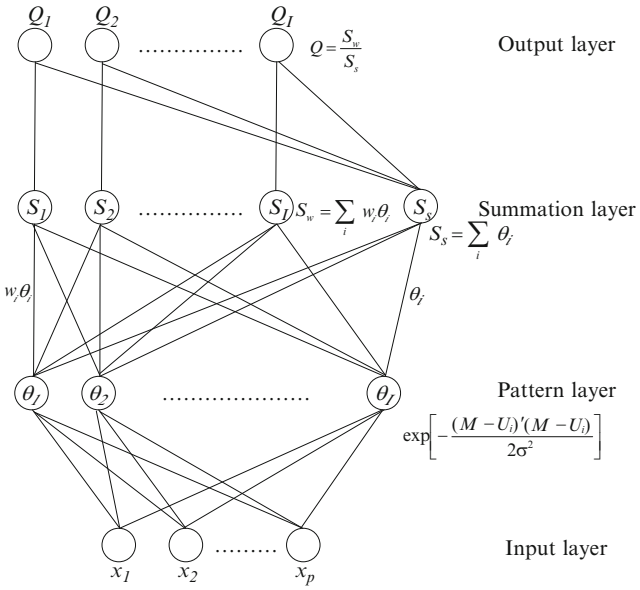

Fig. 2.1 The architecture of the GRNN model

The summations of neurons in the third layer are then fed into the fourth layer. The GRNN regression output Q is calculated as Eq. (2.22):

$$Q = \frac{S_s}{S_w}. \tag{2.22}$$

2.6 Back-Propagation Neural Networks Model

The multilayer back-propagation neural network (BPNN) is one of the most widely used neural network models. Consider the simplest BPNN architecture (Fig. 2.2) including three layers: an input layer (x), an output layer (o), and a hidden layer (h). The computational procedure of this network is described as Eq. (2.23):

$$o_i = f\left(\sum_j g_{ij} x_{ij}\right), \tag{2.23}$$

where o_i denotes the output of node i, $f(\cdot)$ represents the activation function, g_{ij} is the connection weight between nodes i and j in the lower layer which can be replaced with v_{ji} and w_{kj}, and x_{ij} denotes the input signal from the node j in the lower layer.

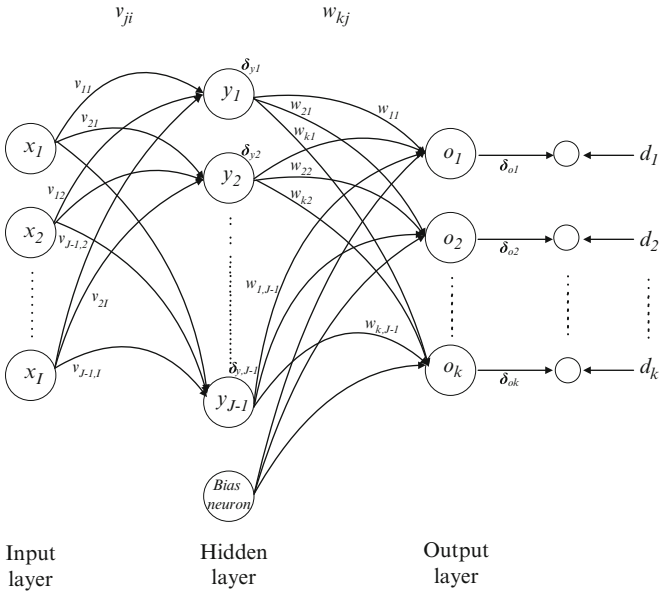

Fig. 2.2 The architecture of the BPNN model

The BPNN algorithm attempts to improve neural network performance (reduce the total error) through changing the gradient weights. The BPNN algorithm minimizes the sum of square error, which can be calculated by Eq. (2.24):

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{j=1}^{K}(d_{pj} - o_{pj})^2, \tag{2.24}$$

where E denotes the square errors, K represents the output layer neurons, P is the training data pattern, d_{pj} denotes the actual output, and o_{pj} represents the network output.

The BPNN algorithm is expressed as follows: Let Δv_{jt} denote the weight change for any hidden layer neuron and Δw_{kj} for any output layer neuron, shown as Eqs. (2.25) and (2.26):

$$\Delta v_{ji} = -\eta\frac{\partial E}{\partial v_{ji}} \quad i = 1,\ldots,I, j = 1,\ldots,J-1 \tag{2.25}$$

$$\Delta w_{kj} = -\eta\frac{\partial E}{\partial w_{kj}} \quad j = 1,\ldots,J-1, k = 1,\ldots,K, \tag{2.26}$$

where η represents the learning rate parameter. Notably, the Jth node in Fig. 2.2 is the bias neuron without weight. Equations (2.27) and (2.28) express the signal (s_j) to each hidden layer neuron and the signal (u_k) to each neuron in the output layer:

$$s_j = \sum_{i=1}^{I} v_{ji} x_i, \tag{2.27}$$

$$u_k = \sum_{j=1}^{J-1} w_{kj} y_j. \tag{2.28}$$

The error signal terms for the jth hidden neuron δ_{yj} and for the kth output neuron δ_{ok} are defined as Eqs. (2.29) and (2.30), respectively:

$$\delta_{yj} = -\frac{\partial E}{\partial s_j}, \tag{2.29}$$

$$\delta_{ok} = -\frac{\partial E}{\partial u_k}. \tag{2.30}$$

Applying the chain rule, the gradient of the cost function with respect to weights v_{ji} and w_{kj} is

$$\frac{\partial E}{\partial v_{ji}} = \frac{\partial E}{\partial s_j} \frac{\partial s_j}{\partial v_{ji}}, \tag{2.31}$$

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial u_k} \frac{\partial u_k}{\partial w_{kj}}, \tag{2.32}$$

and

$$\frac{\partial s_j}{\partial v_{ji}} = \frac{\partial (v_{j1} x_1 + v_{j2} x_2 + \cdots + v_{ji} x_i + \cdots + v_{jI} x_I)}{\partial v_{ji}} = x_i, \tag{2.33}$$

$$\frac{\partial u_k}{\partial w_{kj}} = \frac{\partial (w_{k1} y_1 + w_{k2} y_2 + \cdots + w_{kj} y_j + \cdots + w_{kJ} y_J)}{\partial w_{kj}} = y_j. \tag{2.34}$$

By combining Eqs. (2.29), (2.31), and (2.33) and Eqs. (2.30),(2.32), and (2.34), one obtains Eqs. (2.35) and (2.36):

$$\frac{\partial E}{\partial v_{ji}} = -\delta_{yj} x_i, \tag{2.35}$$

$$\frac{\partial E}{\partial w_{kj}} = -\delta_{ok} y_j. \tag{2.36}$$

The weight change from Eqs. (2.25) and (2.26) can now be written as Eqs. (2.37) and (2.38), respectively:

$$\Delta v_{ji} = -\eta \frac{\partial E}{\partial v_{ji}} = \eta \delta_{yj} x_i, \tag{2.37}$$

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial e_{kj}} = \eta \delta_{ok} y_j. \tag{2.38}$$

Furthermore, Eqs. (2.29) and (2.30) can be calculated as Eqs. (2.39) and (2.40):

$$\delta_{ok} = -\frac{\partial E}{\partial u_k} = -\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial u_k} = (d_k - o_k) f'(u_k), \tag{2.39}$$

$$\delta_{yj} = -\frac{\partial E}{\partial s_j} = -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial s_j} = \left\{ \sum_{k=1}^{K} o_k \cdot w_{kj} \right\} \cdot f'_j(u_j). \tag{2.40}$$

The weights, v_{ji} and w_{kj}, are changed as Eqs. (2.41) and (2.42):

$$w_{kj} = w_{kj} + \Delta w_{kj} = w_{kj} + \eta \delta_{ok} y_j, \tag{2.41}$$

$$v_{ji} = v_{ji} + \Delta v_{ji} = v_{ji} + \eta f'_j(u_j) x_i \sum_{k=1}^{K} \delta_{ok} w_{kj}. \tag{2.42}$$

The constant term, η, is specified at the start of training cycle and determines the training speed and stability of the network. The most common activation functions are the squashing sigmoid function, such as the logistic and tangent hyperbolic functions.

2.7 Support Vector Regression Model

2.7.1 Structural Risk Minimization

Artificial intelligent approaches have tended to be based on finding functions to map as training errors over training set, that is, empirical risk minimization (ERM). However, the ERM does not guarantee good generalization to novel testing data set. To separate the classes with a surface (hyperplane) that maximizes the margin between training data set, SVMs employ the SRM principle that aims to minimize a bound on the generalization error, rather than minimizing the mean square error over the training data set. SRM provides a well-defined quantitative measurement

for the capacity of a learned function to capture the true structure of the data distribution and generalize over unknown test data set. Vapnik–Chervonenkis (VC) dimension [19] has been applied for such a capacity; by selecting a function and minimizing its empirical error to a training data set, SRM can guarantee a minimal bound on the test data set.

Give a training data set of N elements $\{(\mathbf{x}_i, y_i), i = 1, 2, \ldots, N\}$, where $\mathbf{x}_i$ is the ith element in n-dimensional space, that is, $\mathbf{x}_i = [x_{1i}, \ldots, x_{ni}] \in \Re^n$, and $y_i \in \{-1, +1\}$ is the label of $\mathbf{x}_i$, then define a deterministic function f:$\mathbf{x} \rightarrow \{-1,+1\}$ for a given input data $\mathbf{x}$ and adjustable weights $\mathbf{w}(\mathbf{w} \in \Re^n)$, according to the same but unknown probability distribution ($P(\mathbf{x}, y)$). The weights $\mathbf{w}$ would be adjusted during the training stage. Since the underlying probability distribution $P(\mathbf{x}, y)$ is unknown, the upper bound for the probability of classification errors on the test data set (i.e., expected error of f, $R(f)$) cannot be minimized directly. Thus, it is feasible to estimate an approximate function of $R(f)$, that is, empirical risk, denoted as $R_{\text{emp}}(f)$, that is close to the optimal one based on the training data pairs $(\mathbf{x}, y)$. Then, according to the SRM principle [20, 21], $R(f)$ and $R_{\text{emp}}(f)$ are expressed as Eqs. (2.43)–(2.46):

$$R(f) \leq R_{\text{emp}}(f) + \varepsilon_1(N, h, \eta, R_{\text{emp}}), \tag{2.43}$$

$$R_{\text{emp}}(f) = \frac{1}{N}\sum_{i=1}^{N} |y_i - f(\mathbf{x}_i)|_{\text{loss function.}}, \tag{2.44}$$

$$\varepsilon_1(N, h, \eta, R_{\text{emp}}) = 2\varepsilon_0^2(N, h, \eta)\left(1 + \sqrt{1 + \frac{R_{\text{emp}}(f)}{\varepsilon_0^2(N, h, \eta)}}\right), \tag{2.45}$$

$$\varepsilon_0(N, h, \eta) = \sqrt{\frac{h\left(\ln\left(\frac{2N}{h}\right) + 1\right) - \ln\left(\frac{\eta}{4}\right)}{N}}. \tag{2.46}$$

Equation (2.43) holds with probability $1-\eta$ for $0 \leq \eta \leq 1$. The $\varepsilon_0(N,h,\eta)$ is the so-called VC confidence interval. The values of $\varepsilon_0(N,h,\eta)$ depend on the number of training data N, the VC dimension h, and the value of η.

For a small empirical risk $R_{\text{emp}}(f)$, for example, closes to 0, then Eq. (2.43) would approximately reduce to $R_{\text{emp}}(f) + 4{\varepsilon_0}^2(N,h,\eta)$ and, in contrast, for a large empirical risk closes to 1, Eq. (2.43) would approximately reduce to $R_{\text{emp}}(f) + \varepsilon_0(N, h,\eta)$ [22].

Thus, there are two strategies for minimizing the upper bound, $R(f)$. The first one is to keep the VC confidence ($\varepsilon_0(N,h,\eta)$) fixed and to minimize the empirical risk; most of ANN models seek to employ the first one. However, the first way does not perform well because dealing with $R_{\text{emp}}(f)$ lonely cannot guarantee reduction of VC confidence. The second one is to fix the empirical risk to a small value and to minimize the VC confidence, which is the so-called SRM principle. Although

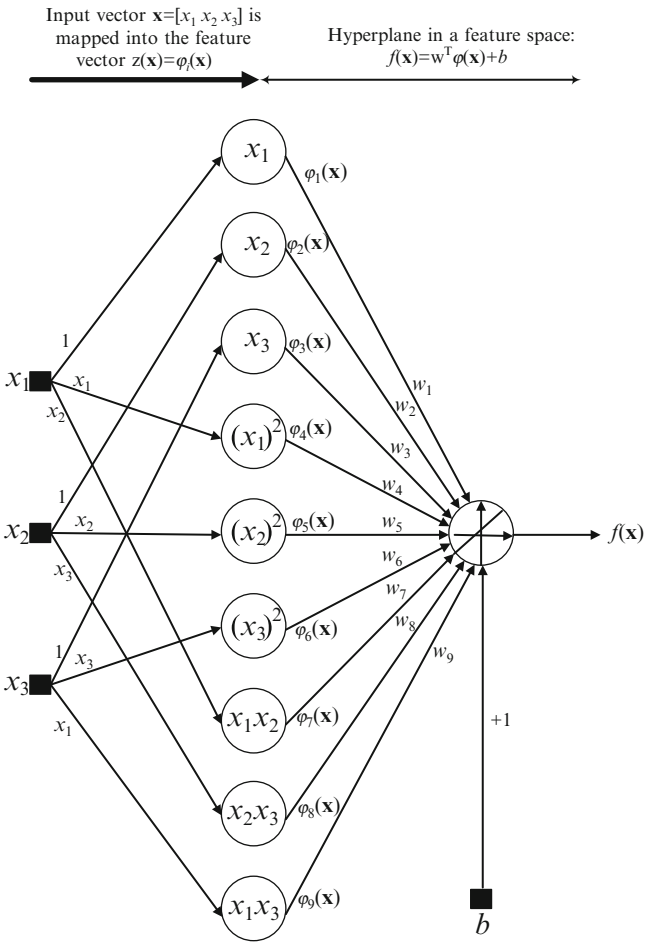

Fig. 2.3 Transformation of the second-order polynomial hyperplane over a three-dimensional original space in an SVR model

SVMs implement this principle, their training algorithm that aims to minimize the VC dimension is still based on a hierarchy that depends on the data [20, 23].

2.7.2 Support Vector Regression

As mentioned above, SVMs have originally been used for classification purposes, but their principles can be extended easily to the task of regression and time series prediction. The brief ideas of SVMs for the case of regression are introduced. A nonlinear mapping $\varphi(\cdot) : \Re^n \rightarrow \Re^{n_h}$ is defined to map the input data (training data

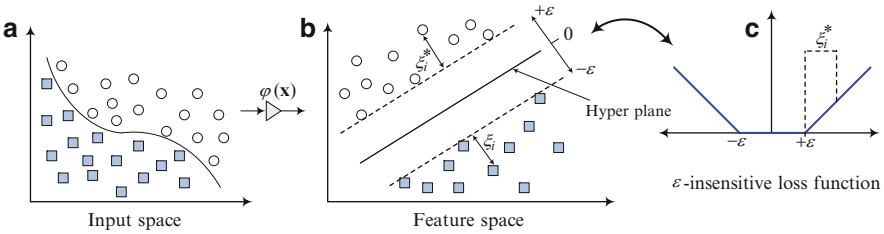

Fig. 2.4 Transformation process illustration of an SVR model

set) $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ into a so-called high-dimensional feature space (Fig. 2.3), which may have infinite dimensions, $\Re^{n_h}$. Then, in the high-dimensional feature space, there theoretically exists a linear function, f, to formulate the nonlinear relationship between input data and output data (Fig. 2.4a, b). Such a linear function, namely, SVR function, is as Eq. (2.47):

$$f(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) + b, \tag{2.47}$$

where $f(\mathbf{x})$ denotes the forecasting values and the coefficients $\mathbf{w}$ ($\mathbf{w} \in \Re^{n_h}$) and b ($b \in \Re$) are adjustable. As mentioned above, using SVM method one aims at minimizing the empirical risk as Eq. (2.48):

$$R_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^{N} \Theta_\varepsilon(y_i, \mathbf{w}^T \varphi(\mathbf{x}_i) + b), \tag{2.48}$$

where $\Theta_\varepsilon(\mathbf{y}, f(\mathbf{x}))$ is the ε-insensitive loss function (as thick line in Fig. 2.4c) and is defined as Eq. (2.49):

$$\Theta_\varepsilon(\mathbf{y}, f(\mathbf{x})) = \begin{cases} |f(\mathbf{x}) - \mathbf{y}| - \varepsilon, & \text{if } |f(\mathbf{x}) - \mathbf{y}| \geq \varepsilon \\ 0, & \text{otherwise} \end{cases}. \tag{2.49}$$

In addition, $\Theta_\varepsilon(\mathbf{y}, f(\mathbf{x}))$ is employed to find out an optimum hyperplane on the high-dimensional feature space (Fig. 2.4b) to maximize the distance separating the training data into two subsets. Thus, the SVR focuses on finding the optimum hyperplane and minimizing the training error between the training data and the ε-insensitive loss function.

Then, the SVR minimizes the overall errors, shown as Eq. (2.50):

$$\underset{\mathbf{w}, b, \xi^*, \xi}{\text{Min}} R_\varepsilon(\mathbf{w}, \xi^*, \xi) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{N} (\xi_i^* + \xi_i), \tag{2.50}$$

with the constraints

$$\begin{aligned}
&\mathbf{y}_i - \mathbf{w}^T\varphi(\mathbf{x}_i) - b \le \varepsilon + \xi_i^*, \quad i = 1, 2, \ldots, N \\
&-\mathbf{y}_i + \mathbf{w}^T\varphi(\mathbf{x}_i) + b \le \varepsilon + \xi_i, \quad i = 1, 2, \ldots, N \\
&\xi_i^* \ge 0, \quad i = 1, 2, \ldots, N \\
&\xi_i \ge 0, \quad i = 1, 2, \ldots, N
\end{aligned}.$$

The first term of Eq. (2.50), employing the concept of maximizing the distance of two separated training data, is used to regularize weight sizes, to penalize large weights, and to maintain regression function flatness. The second term penalizes training errors of $f(\mathbf{x})$ and $\mathbf{y}$ by using the ε-insensitive loss function. C is a parameter to trade off these two terms. Training errors above $+\varepsilon$ are denoted as ξ_i^*, whereas training errors below $-\varepsilon$ are denoted as ξ_i (Fig. 2.4b).

After the quadratic optimization problem with inequality constraints is solved, the parameter vector $\mathbf{w}$ in Eq. (2.51) is obtained:

$$\mathbf{w} = \sum_{i=1}^{N} (\beta_i^* - \beta_i)\varphi(\mathbf{x}_i), \tag{2.51}$$

where β_i^*, β_i are obtained by solving a quadratic program and are the Lagrangian multipliers. Finally, the SVR regression function is obtained as Eq. (2.52) in the dual space:

$$f(\mathbf{x}) = \sum_{i=1}^{N} (\beta_i^* - \beta_i)K(\mathbf{x}_i, \mathbf{x}) + b, \tag{2.52}$$

where $K(\mathbf{x}_i, \mathbf{x}_j)$ is called the kernel function and the value of the kernel equals the inner product of two vectors, $\mathbf{x}_i$ and $\mathbf{x}_j$, in the feature space $\varphi(\mathbf{x}_i)$ and $\varphi(\mathbf{x}_j)$, respectively; that is, $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \circ \varphi(\mathbf{x}_j)$. Any function that meets Mercer's condition [20] can be used as the kernel function.

There are several types of kernel function. The most used kernel functions are the Gaussian RBF with a width of σ: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-0.5\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2)$, the polynomial kernel with an order of d and constants a_1 and a_2: $K(\mathbf{x}_i, \mathbf{x}_j) = (a_1 \mathbf{x}_i\mathbf{x}_j + a_2)^d$, and $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\mathbf{x}_i^T\mathbf{x}_j - b)$, where b is a constant, in which, if the value of σ is very large, the RBF kernel approximates the use of a linear kernel (polynomial with an order of 1). Till now, it is hard to determine the type of kernel functions for specific data patterns [24, 25]. However, based on Smola et al.'s [26] empirical results, they claim that the Gaussian RBF kernel is not only easier to implement but also capable to nonlinearly map the training data into an infinite-dimensional space; thus, it is suitable to deal with nonlinear relationship problems. Therefore, the Gaussian RBF kernel function is specified in this book. The forecasting process of an SVR model is illustrated as in Fig. 2.5.

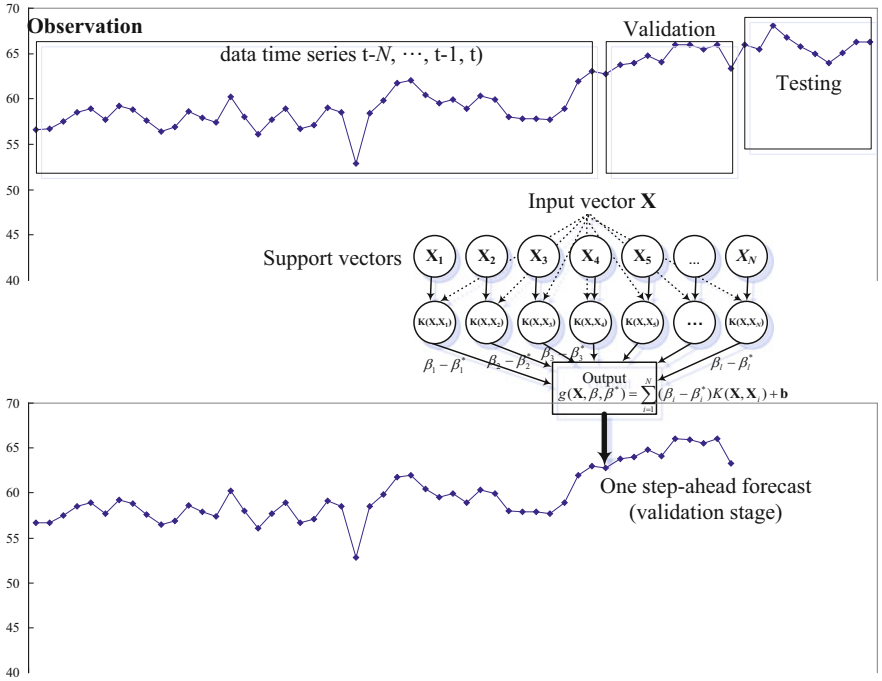

Fig. 2.5 The forecasting process of an SVR model

2.7.3 The Role of Evolutionary Algorithms

It is well known that the forecasting accuracy of an SVR model depends on a good setting of hyper parameters C, ε, and the kernel parameters (σ). For example, parameter C is considered to specify the trade-off between the model flatness and the degree of the training errors larger than ε which are tolerated in Eq. (2.48) (i.e., the empirical risk). If C is too large (approximated to infinity), then the objective is only to minimize the empirical risk, $\Theta_{\varepsilon}(\mathbf{y}, f(\mathbf{x}))$, without model flatness in the optimization formulation, Eq. (2.48). Parameter ε controls the width of the ε-insensitive loss function, that is, the number of support vectors (SVs) employed by the regression [20]. Larger ε-value results in fewer SVs employed; thus, the regression function is more flat (simple). Parameter σ controls the Gaussian function width, which reflects the distribution range of $\mathbf{x}$-values of training data. Therefore, the three parameters affect model constructing in different ways. There is no structural method or any shortage opinions on efficient setting of SVR parameters. Although numerous publications in the literature have given some recommendations on appropriate setting of SVR parameters [27], however, those approaches do not simultaneously consider the interaction effects among the three parameters. Thus, the determination of these three parameters' selection is further an important issue.

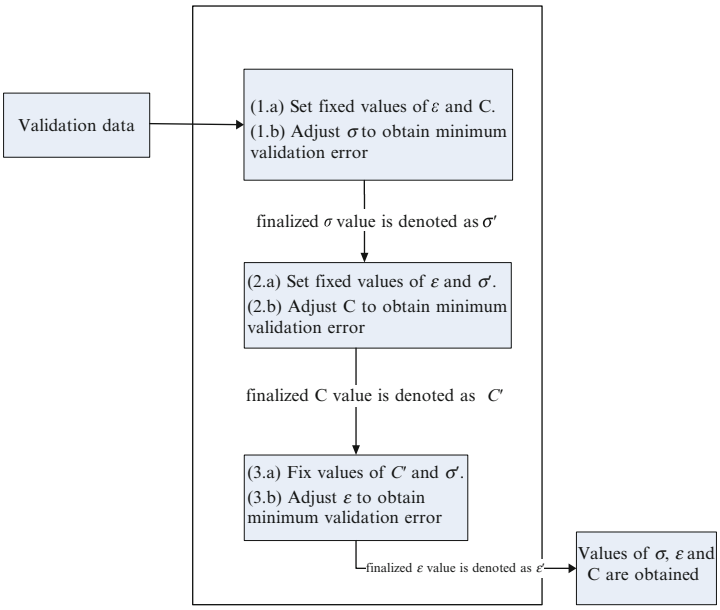

Fig. 2.6 The traditional determining processes of three parameters in an SVR model

The traditional determination procedure in determining suitable values of these three parameters is conducted in following steps and shown as Fig. 2.6:

Step 1: Set fixed values of the parameters ε and C. Then, adjust the value of σ till a minimum testing error is achieved. The finalized σ value is denoted as σ'.

Step 2: Set a fixed value of the parameter ε, and the value of σ is set to σ'. Then, adjust the value of C to achieve a minimum testing error. The finalized C is defined as C'.

Step 3: Values of σ and C are set to σ' and C'. Then, adjust ε till a minimum testing error is obtained. The finalized ε is defined as ε'. Therefore, values of σ, ε, and C are obtained as σ', ε', and C'.

Some numerical examples are used to illustrate step by step the traditional determination procedure. For the first example, please refer Figs. 2.7, 2.8, and 2.9. First, the fixed values, 0.5 and 50, are set to the parameters σ and C, respectively. Then, adjust the value of ε till a minimum testing error (NRMSE) is reached. It is found that there exists only one local minimum value of NRMSE (0.008049) when $\varepsilon = 0.27$, shown in Fig. 2.7. Therefore, set finalized ε-value as 0.27. Second, set $\varepsilon = 0.27$ and $C = 50$, respectively. It is found that there also exists only one local minimum value of NRMSE (0.007584) when $\sigma = 0.45$, presented in Fig. 2.8. Finally, the third step will focus on searching the third parameter, set to $\sigma = 0.45$ and $\varepsilon = 0.27$; it is found that there exists only one local minimum value of NRMSE (0.006446) when $C = 60$, presented in Fig. 2.9. Therefore, the three parameters are estimated as follows: $\sigma = 0.45$, $C = 60$, and $\varepsilon = 0.27$.

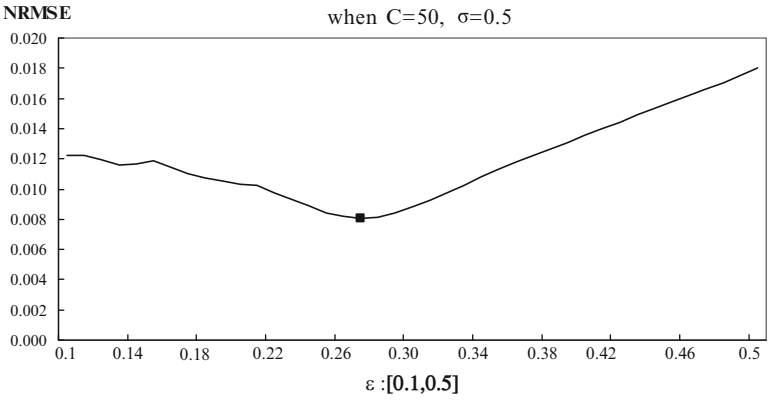

Fig. 2.7 Fixed values of C and σ to adjust ε

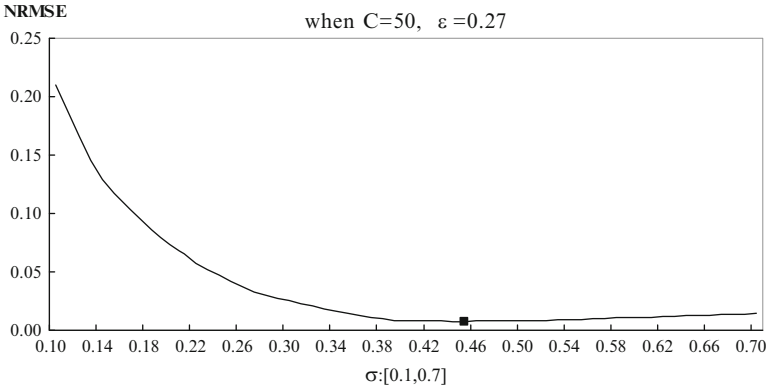

Fig. 2.8 Fixed values of C and ε to adjust σ

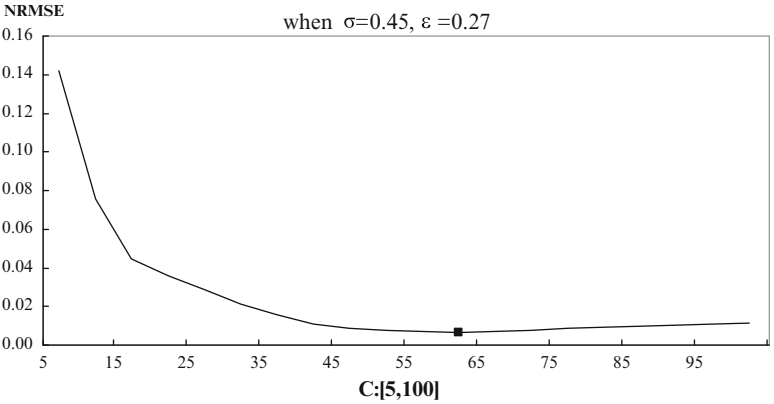

Fig. 2.9 Fixed values of σ and ε to adjust C

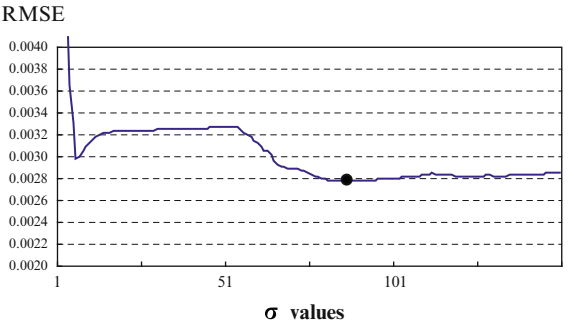

Fig. 2.10 Fixed values of C and ε to adjust σ values

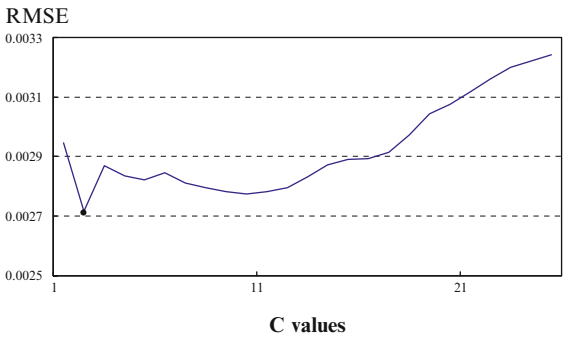

Fig. 2.11 Fixed values of σ and ε to adjust C values

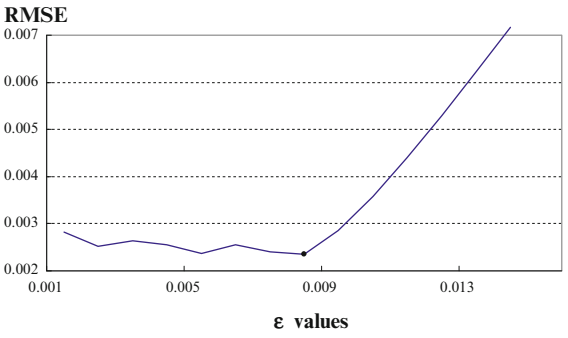

Fig. 2.12 Fixed values of σ and C to adjust ε-values

For the second example, please refer Figs. 2.10, 2.11, and 2.12. First, set the values of the parameters ε and C fixed at 0 and 10, respectively. Then, adjust the value of σ till a minimum testing error (RMSE) is reached. In Fig. 2.10, when the finalized σ value equaled to 87, the minimum value of RMSE is achieved. Therefore, set finalized σ value as 87. Secondly, fix the values of the parameters σ and ε at 87 and 0, respectively. Then, adjust the value of C (Fig. 2.11). The finalized value of C is 2. The minimum value of RMSE is obtained. Therefore, the finalized C value is 2. Finally, set the values of the parameters σ and C fixed at

87 and 2, respectively. Then, adjust the value of ε (Fig. 2.12). The ε-value equals to 0.008 when the minimum value of RMSE is achieved. Therefore, the values of three parameters (σ, C, and ε) are 87, 2, and 0.008, correspondingly. The minimum value of RMSE is 0.002345.

The traditional determination of these three parameters is not only time-consuming but also unable to receive satisfied forecasting accuracy level. This is because it is difficult to set up more suitable initial values of parameters ε and C in the initial step and it cannot efficiently find out the near-optimal solution for large-scale data set; particularly, while simultaneously considering the interaction effects among the three parameters, the computing complexity will exhaust the limited decision time. Therefore, it is feasible to employ optimization solving procedure to obtain suitable parameter combination, such as minimizing the objective function describing the structural risk mentioned above. Evolutionary algorithms, such as genetic algorithm, simulated annealing algorithms, immune algorithms, particle swarm optimization, and tabu search, are the very candidates to be employed to determine appropriate parameter values. The author will start a series exploration by employing different evolutionary algorithms to determine suitable parameters of an SVR model in Chap. 3.

References

1. Moghram I, Rahman S (1989) Analysis and evaluation of five short-term load forecasting techniques. IEEE Trans Power Syst 4:1484–1491. doi:10.1109/59. 41700
2. El-Hawary ME, Mbamalu GAN (1990) Short-term power system load forecasting using the iteratively reweighted least squares algorithm. Electric Power Syst Res 19:11–22. doi:10.1016/0378-7796(90)90003-L
3. Papalexopoulos AD, Hesterberg TC (1990) A regression-based approach to short-term system load forecasting. IEEE Trans Power Syst 5:1535–1547. doi:10.1109/59.99410
4. Grady WM, Groce LA, Huebner TM, Lu QC, Crawford MM (1991) Enhancement, implementation and performance of an adaptive short-term load forecasting algorithm. IEEE Trans Power Syst 6:1404–1410. doi:10.1109/59. 116982
5. Al-Fuhaid AS, EL-Sayed MA, Mahmoud MS (1997) Cascaded artificial neural networks for short-term load forecasting. IEEE Trans Power Syst 12:1524–1529. doi:10.1109/59.627852
6. Hong WC (2009) Hybrid evolutionary algorithms in a SVR-based electric load forecasting model. Int J Electr Power Energ Syst 31:409–417. doi:10.1016/j.ijepes.2009.03.020
7. Hong WC (2009) Chaotic particle swarm optimization algorithm in a support vector regression electric load forecasting model. Energy Convers Manage 50:105–117. doi:10.1016/j.enconman.2008.08.031
8. Hong WC (2009) Electric load forecasting by support vector model. Appl Math Model 33:2444–2454. doi:10.1016/j.apm.2008.07.010
9. Hong WC (2010) Application of chaotic ant swarm optimization in electric load forecasting. Energ Policy 38:5830–5839. doi:10.1016/j.enpol.2010.05.033
10. Pai PF, Hong WC (2005) Forecasting regional electric load based on recurrent support vector machines with genetic algorithms. Electric Power Syst Res 74:417–425. doi:10.1016/j.epsr.2005.01.006

11. Pai PF, Hong WC (2005) Support vector machines with simulated annealing algorithms in electricity load forecasting. Energy Convers Manage 46:2669–2688. doi:10.1016/j.enconman.2005.02.004
12. Hong WC, Dong Y, Zhang WY, Chen LY, Panigrahi BK (2013) Cyclic electric load forecasting by seasonal SVR with chaotic genetic algorithm. Int J Electr Power Energ Syst 44:604–614. doi:10.1016/j.ijepes.2012.08.010
13. Box GEP, Jenkins GM (1970) Time series analysis, forecasting and control. Holden-Day, San Francisco, CA
14. Abu-El-Magd MA, Sinha NK (1982) Short-term load demand modeling and forecasting: a review. IEEE Trans Syst Man Cybern 12:370–382. doi:10.1109/TSMC.1982.4308827
15. Soliman SA, Persaud S, El-Nagar K, El-Hawary ME (1997) Application of least absolute value parameter estimation based on linear programming to short-term load forecasting. Int J Electr Power Energ Syst 19:209–216. doi:10.1016/S0142-0615(96)00048-8
16. Holt CC (1957) Forecasting seasonal and trends by exponentially weighted averages. Carnegie Institute of Technology, Pittsburgh, PA
17. Winters PR (1960) Forecasting sales by exponentially weighted moving averages. Manag Sci 6:324–342. doi:10.1287/mnsc.6.3.324
18. Specht DA (1991) A general regression neural network. IEEE Trans Neural Netw 2:568–576. doi:10.1109/72.97934
19. Cao L, Gu Q (2002) Dynamic support vector machines for non-stationary time series forecasting. Intell Data Anal 6:67–83
20. Vapnik V (1995) The nature of statistical learning theory. Springer, New York, NY
21. Cortes C, Vapnik V (1995) Support vector networks. Mach Learn 20:273–297. doi:10.1023/A:1022627411411
22. Haykin S (1999) Neural networks: a comprehensive foundation. Prentice-Hall, Upper Saddle River, NJ
23. Shawe-Taylor J, Bartlett PL, Williamson RC, Anthony M (1998) Structural risk minimization over data-dependent hierarchies. IEEE Trans Inf Theory 44:1926–1940. doi:10.1109/18.705570
24. Amari S, Wu S (1999) Improving support vector machine classifiers by modifying kernel functions. Neural Netw 12:783–789. doi:10.1016/S0893- 6080(99)00032-5
25. Vojislav K (2001) Learning and soft computing—support vector machines, neural networks and fuzzy logic models. The MIT Press, Cambridge, MA, 2001
26. Smola AJ, Schölkopf B, Müller KR (1998) The connection between regularization operators and support vector kernels. Neural Netw 11:637–649. doi:10.1016/S0893-6080(98)00032-X
27. Cherkassky V, Ma Y (2004) Practical selection of SVM parameters and noise estimation for SVM regression. Neural Netw 17:113–126. doi:10.1016/S0893-6080(03)00169-2