Particle Swarm Optimization-based LS-SVM for Building Cooling Load Prediction

Li Xuemei¹², Shao Ming¹, Ding Lixing²
¹. School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China, 510640
². Institute of Built Environment and Control, Zhongkai University of Agriculture and Engineering, Guangzhou, China, 510225
Email: mexmlee@gmail.com

Xu Gang³⁴, Li Jibin⁴
³. School of Mechatronics and Control Engineering, Shenzhen University, Shenzhen, China, 518060
⁴. Shenzhen Key laboratory of mould advanced manufacture, Shenzhen, China, 518060

Abstract—Accurate predicting of building cooling load has been one of the most important issues in the energy-saving building, which provides an approach to integrate and optimize the heating, ventilating, and air-conditioning (HVAC) system cooling supply system efficiently. Because of the remarkable nonlinear mapping capabilities of forecasting, artificial neural networks have played a crucial role in forecasting building cooling load, but suffer from the phenomena of local minimum and over-fitting. This paper investigates the feasibility of using Least Squares Support vector regression (LS-SVR) to forecast building cooling load. LS-SVR is a novel type of learning machine, which has been successfully employed to solve nonlinear regression and time series problems. Due to the importance of parameters optimization in LS-SVR model, particle swarm optimization (PSO) was used to optimize the model parameters. The experiment results show that PSO can quickly obtain the optimal parameters satisfying the precision requirement with a simple calculation, which solves the problem of complex calculation and empiricism in conventional methods. The evaluation on the testing cases shows the SVR model with PSO has a good generalization performance and can be a promising alternative for building cooling load prediction.

Index Terms—building cooling load prediction, LSSVR, particle swarm optimizer, parameter identification, energy-saving building

I. INTRODUCTION

The air-conditioning load prediction, especially the short term prediction, provides an approach to integrate and optimize the heating, ventilating, and air-conditioning (HVAC) system cooling supply system efficiently, based on which the air-conditioning supply matches the demand well[1]. What is more, it is also useful for HVAC operations including adjusting the starting time of cooling to meet start-up loads, minimizing or limiting the electric on-peak demand, optimizing the costs and energy utilization in cool storage systems, and related energy and cost needs in other HVAC systems [1-3].

Accurately predicting the building cooling load is a challenging work. In recent years, some researchers began to focus on the prediction of building cooling load. Several prediction methods have ever been adopted [4-6], such as the admittance and Fourier methods, the transfer function method, the neural networks method, the Monte Carlo simulation method, the Kalman filter methods, and so on. Although these methods have alleviated difficulties in air-conditioning load modeling and prediction to some extent, from a careful review we can still find some problems. However, The Kalman filter methods are unsuitable for predicting the cooling load which sample interval is less than 5 min. The nonparametric regression methods need a huge historical database which occupies many memory and takes much time to predict the cooling load. The neural network methods suffer from problems like the existence of local minima and the limited generalization ability. In order to improve prediction performance, the hybrid model combined with wavelet analysis and neural network is proposed, but accompanied with low efficiency due to the inherent theory flaw from neural networks. It is hard and time-consuming to use those professional energy software, therefore, it is very difficult for the common operators to predict building load using them.

Recently, a novel type of learning machine, called support vector machine (SVM), has been receiving increasing attention. SVM was developed by Vapnik and his coworkers in 1995 [7], and it is based on the structure risk minimization (SRM) principle that seeks to minimize an upper bound of the generalization error consisting of the sum of the training error and a confidence interval.
The time-varying properties of SVR applications resemble the time-dependency of cooling load forecasting, combined with many successful results of SVR predictions encourage our research in using SVR for cooling load forecasting modeling. SVM possess great potential and superior performance as is appeared in many previous researches [23], the results guarantee global minima. SVM can solve some flaws of the neural networks, and has many unique advantages in the fields of small samples and high-dimensional nonlinear manifested.

There is also a great deal of researches concentrating on applying SVM regression to building cooling load forecasting [9] and the forecasting accuracy outperforms other forecasting models. The selection of parameters of a SVM model is important to the accuracy of the forecasting. However, most SVM practitioners select these parameters empirically by trying a finite number of values and keeping those that provide the least testing error. This procedure requires a grid search over the space of parameter values and needs to locate the interval of feasible solution and a suitable sampling step. Because of the computational complexity, grid search is only suitable for the adjustment of very few parameters [10].

In farther researches, some intelligent algorithms such as evolution algorithms (EA) and genetic algorithms (GA) were employed to choose the parameters of a SVM model, and the improved model offers a superior performance to ordinary regression SVM model[11,11]. The particle swarm optimization (PSO) algorithm [13], a relatively new evolutionary computation (EC) stochastic technique, can also be used as an excellent optimizer which originated as a simulation of the food-searching behavior of birds. Similar to EA and GA, PSO is a population based optimization tool, which search for optima by updating generations. However, unlike GA and EA, PSO has no evolution operators such as crossover and mutation. Compared to GA and EA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. Most versions of PSO have operated in continuous and real-number space.

The remainder of this paper is organized as follows. In Section 2, the LS-SVM is introduced briefly, then a building cooling load prediction model based on the LS-SVM theory is established, where Particle Swarm Optimization algorithm is used to optimize SVR model parameters. The experimental results of applying the proposed forecasting model to the cooling load prediction are presented in Section 3. Finally, conclusion is drawn in section 4.

II. BASIC PRINCIPLE OF SVR AND PSO

A. Least Square Support Vector Machine

Least Square Support Vector Machine (LS-SVM) is a new technique for regression. When LS-SVM is used to model cooling load, the input and output variables should be chosen firstly. Hence, this paper takes three meteorological parameters, such as the outdoor temperature, humidity and solar radiation, as the input parameters of the building cooling load prediction model. Furthermore, considering the delay of air temperature and solar radiation intensity’s influence on the dynamic cooling load, their history values are also selected as the input parameters. The hourly building cooling load is chosen as the model’s output.

Given a training data set \( \{ (x_1, y_1), \ldots, (x_n, y_n) \} \) with input data \( x_i \in \mathbb{R}^n \) and output data \( y_i \in \mathbb{R} \). In order to get the function dependence relation, SVM map the input space into a high-dimension feature space and construct a linear regression in it. The regression function is expressed with

\[
y = f(x) = w^T \varphi(x) + b
\]

with \( \varphi(\cdot): \mathbb{R}^n \rightarrow \mathbb{R}^m \), a function which maps the input space into a so-called higher dimensional (possibly infinite dimensional) feature space, \( w \) and \( b \) are the regression parameters to be solved.

LS-SVM regression estimation involves primal and dual model formulations. Given the training data set \( \{ (x_1, y_1), \ldots, (x_n, y_n) \} \), the goal is to estimate the model (1), where \( y \) is parameterized as in (4), we can formulate the following optimization scheme to infer our parameters

\[
\min_{w,b}\sum_{i=1}^{n}\left[\epsilon_i^+(y_i-w^T\varphi(x)+b)+\epsilon_i^-(y_i-w^T\varphi(x)+b)\right]
\]

\[
s.t.\quad y_i=w^T\varphi(x)+b+\epsilon_i^+,\quad i=1,\ldots,n
\]

where error variables \( \epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)^T, \epsilon_i \in \mathbb{R} \), the regularization constant \( \gamma > 0 \) is included to control the bias-variance trade-off. The above statement is in fact the same formulation as is used in case of ridge regression [21] in the feature space defined by \( \varphi(\cdot) \). Note that in some cases \( w \) becomes infinite dimension, and the above problem formulation cannot be used to solve the problem. Therefore, we perform the computations in another space, called the dual space of Lagrangian multipliers after applying Mercer’s theorem. Consider the Lagrangian of (2) given by

\[
L_\alpha(w,b,e,\alpha) = \frac{1}{2}w^T\Phi(w) + \frac{\gamma}{2} \sum_{i=1}^{n} e_i^2 - \sum_{i=1}^{n} \alpha_i(\epsilon_i^+(y_i-w^T\varphi(x)+b)+\epsilon_i^-(y_i-w^T\varphi(x)+b))
\]

Here \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T, \alpha_i \in \mathbb{R} \) are Lagrangian multipliers. The first order conditions for optimality are given by:

\[
\begin{align*}
\frac{\partial L_\alpha}{\partial w} &= 0 \Rightarrow \sum_{i=1}^{n} \alpha_i \varphi(x_i), \\
\frac{\partial L_\alpha}{\partial b} &= 0 \Rightarrow 0 = \sum_{i=1}^{n} \alpha_i, \\
\frac{\partial L_\alpha}{\partial e_i} &= 0 \Rightarrow \alpha_i = \gamma e_i, i = 1, 2, \ldots, n, \\
\frac{\partial L_\alpha}{\partial \alpha_i} &= 0 \Rightarrow y_i = w^T\varphi(x_i) + b + e_i, i = 1, 2, \ldots, n
\end{align*}
\]

That is:
Note that in the case of RBF kernels, one has only two additional tuning parameters $\sigma$ in Eq.(9) and $\gamma$ in Eq.(5).

From the training LS-SVM problem, one can see that there are two free parameters, viz. kernel width parameter $\sigma$ and regularization parameter $\gamma$, which may affect LS-SVM generalization performance. So these parameters need to be properly optimized to minimize the generalization error. In this paper, these parameters are automatically tuned using the PSO in the training phase.

**Basic principle of PSO**

PSO is a stochastic optimization technique introduced recently by Kennedy and Eberhart, which is inspired by social behavior of bird flocking and fish schooling [14, 15]. Similar to other evolutionary computation algorithms such as genetic algorithms, PSO is a population-based search method that exploits a population of individuals to probe promising region of the search space. In this context, the population is called swarm and the individuals are called particles. During the search process in the $d$-dimensional solution space, each particle (i.e., candidate solution) will adjust its flying velocity and position according to its own flying experience as well as the experiences of the other companion particles of the swarm. The global variant of PSO the best position ever attained by all individuals of the swarm is communicated to all the particles.

During last decade many studies focused on this method and almost all of them, strongly confirmed the abilities of this newly proposed optimization technique [14-20]. Abilities such as fast convergence, finding global optimum in presence of many local optima, simple programming and adaptability with constrained problems. PSO has showed to be promising for solving various engineering problems such as automatic control [18], antenna design [19], and inverse problems [20]. The general principles for the PSO algorithm are stated as follows:

Let us consider a swarm of size $n$. Each particle $P_i$ ($i = 1, 2, \ldots, n$) from the swarm is characterized by: 1) its current position $X_i \in \mathbb{R}^d$, which refers to a candidate solution of the optimization problem at iteration $k$; 2) its velocity $V_i \in \mathbb{R}^d$; and 3) the best position $P_{best}(k) \in \mathbb{R}^d$ that is identified during its past trajectory. Let $\gamma \in \mathbb{R}^d$ be the best global position found over all trajectories that are traveled by the articles of the swarm. Each of $n$ particles fly through the $d$-dimensional search space $\mathbb{R}^d$ with a velocity $V_i(k)$, which is dynamically adjusted according to its personal previous best solution $P_{best}(k)$ and the previous global solution $G_{best}(k)$ of the entire swarm. The velocity updates are calculated as a linear combination of position and velocity vectors. The particles interact and move according to the following equations

$$V_i(k+1) = wV_i(k) + C_1 \cdot R_1(k) \cdot (P_{best}(k) - X_i(k)) + C_2 \cdot R_2(k) \cdot (G_{best}(k) - X_i(k))$$

$$X_i(k+1) = X_i(k) + V_i(k+1)$$

Note that in the case of RBF kernels, one has only two additional tuning parameters $\sigma$ in Eq.(9) and $\gamma$ in Eq.(5).

From the training LS-SVM problem, one can see that there are two free parameters, viz. kernel width parameter $\sigma$ and regularization parameter $\gamma$, which may affect LS-SVM generalization performance. So these parameters need to be properly optimized to minimize the generalization error. In this paper, these parameters are automatically tuned using the PSO in the training phase.

**Basic principle of PSO**

PSO is a stochastic optimization technique introduced recently by Kennedy and Eberhart, which is inspired by social behavior of bird flocking and fish schooling [14, 15]. Similar to other evolutionary computation algorithms such as genetic algorithms, PSO is a population-based search method that exploits a population of individuals to probe promising region of the search space. In this context, the population is called swarm and the individuals are called particles. During the search process in the $d$-dimensional solution space, each particle (i.e., candidate solution) will adjust its flying velocity and position according to its own flying experience as well as the experiences of the other companion particles of the swarm. The global variant of PSO the best position ever attained by all individuals of the swarm is communicated to all the particles.

During last decade many studies focused on this method and almost all of them, strongly confirmed the abilities of this newly proposed optimization technique [14-20]. Abilities such as fast convergence, finding global optimum in presence of many local optima, simple programming and adaptability with constrained problems. PSO has showed to be promising for solving various engineering problems such as automatic control [18], antenna design [19], and inverse problems [20]. The general principles for the PSO algorithm are stated as follows:

Let us consider a swarm of size $n$. Each particle $P_i$ ($i = 1, 2, \ldots, n$) from the swarm is characterized by: 1) its current position $X_i \in \mathbb{R}^d$, which refers to a candidate solution of the optimization problem at iteration $k$; 2) its velocity $V_i(k) \in \mathbb{R}^d$; and 3) the best position $P_{best}(k) \in \mathbb{R}^d$ that is identified during its past trajectory. Let $\gamma \in \mathbb{R}^d$ be the best global position found over all trajectories that are traveled by the articles of the swarm. Each of $n$ particles fly through the $d$-dimensional search space $\mathbb{R}^d$ with a velocity $V_i(k)$, which is dynamically adjusted according to its personal previous best solution $P_{best}(k)$ and the previous global solution $G_{best}(k)$ of the entire swarm. The velocity updates are calculated as a linear combination of position and velocity vectors. The particles interact and move according to the following equations

$$V_i(k+1) = wV_i(k) + C_1 \cdot R_1(k) \cdot (P_{best}(k) - X_i(k)) + C_2 \cdot R_2(k) \cdot (G_{best}(k) - X_i(k))$$

$$X_i(k+1) = X_i(k) + V_i(k+1)$$

Note that in the case of RBF kernels, one has only two additional tuning parameters $\sigma$ in Eq.(9) and $\gamma$ in Eq.(5).
Where $V(k+1)$ is the velocity of $(k+1)^{th}$ iteration of $i^{th}$ individual, $V(k)$ is the velocity of $k^{th}$ iteration of $i^{th}$ individual, $w(k)$ is the inertial weight used as a tradeoff between global and local exploration capabilities of the swarm. Small values of $w$ result in more rapid convergence usually on a suboptimal position, while a too large value may prevent divergence. Typical implementations of the PSO adapt the value of $w$ during the training stage, e.g., linearly decreasing it from 1.0 to near 0 over the execution. $C_1$, $C_2$ are two acceleration constants regulating the relative velocities with respect to the best global and local positions, respectively. $R_1(\cdot)$ and $R_2(\cdot)$ are random variables that are drawn from a uniform distribution in the range $[0, 1]$ to provide a stochastic weighting of the different components participating in the particle velocity definition.

The inertia weight is set to the following equation:

$$w(k) = w_{\text{max}} - \frac{w_{\text{max}} - w_{\text{min}}}{k_{\text{max}}} \cdot k \quad (16)$$

where $w_{\text{max}}$ is the initial weight, $w_{\text{min}}$ is the final weight, $k_{\text{max}}$ is the maximum number of iterations or generation, and $k$ is the current iteration number.

Eq.(14) allows the computation of the velocity at iteration $k+1$ for each particle in the swarm by combining linearly its current velocity (at iteration $k$) and the distances that separate the current particle position from its best previous position and the best global position, respectively. The updating of the particle position is performed with (14). Both (14) and (15) are iterated until convergence of the search process is reached. Typical convergence criteria are based on the iterative behavior of the best value of the adopted fitness function(s) and/or simply on a user-defined maximum number of iterations.

### III. SVR-PSO Based Building Cooling Load Prediction Modeling

#### A. Problem formulation

According to the changes of time series of the building cooling load, the current building cooling load is certainly linked with that of several hours ago. Thus the previous building cooling load sequence data can be used to predict the future building cooling load.

Assuming $v_i(t)$ is the building cooling load value in $t$ moment $i$, $v_i(t-1)$ is the building cooling load value in $t-1$ moment on road section $i$. The building cooling load value of current and previous $s$ time period can be used to forecast the building cooling load value in the future time period. Let $v_1(t), v_2(t-1), \ldots, v_s(t-s)$ be the samples input vector in $t$ moment as $x_i$, $v_i(t+1)$ be the samples output value $y_i$.

Chose $n$ building cooling load samples as initial training set: $S \equiv \{x_i, y_i = (x_i, y_i), i = 1, 2, \ldots, n\}$. According to SVR algorithm, the initial prediction regression function is obtained. Then the prediction value for building cooling load $\hat{y}_i$ is:

$$\hat{y}_i = \sum_{j=1}^{i} (\alpha_j - \alpha_j^*) K(x_i, x) + b \quad (17)$$

#### B. Building Cooling Load Procedures Based on SVR-PSO

While using SVR-PSO model for short term building cooling load forecasting, the history data is divided into two data sets. One is the training data and the other is the testing data. The training data is used to train the LS-SVR for determining model parameters, and the PSO algorithm is employed to determine the free parameters of SVR, then the testing data is used to evaluate the trained LS-SVR.

PSO mentioned above is used to select parameters of LS-SVR: the trade off variable $\gamma$ and kernel parameters $\sigma$, which are two attributes of each particle. In PSO operation, the fitness function of the particles group with test cases was evaluated using the mean absolute percent error (MAPE). The performance metrics is showed by the following equation:

$$E_{\text{MAPE}} = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{y_t - f(x_t)}{y_t} \right|$$

Where $y_t$ is the actual output, $f(x_t)$ is the estimated value in the SVR model, $n$ is the number of sets of test cases.

At the beginning of the algorithm we have ten particles in a group, and the velocity and position of them are created at random. The mean square error is the objective of the particles group. With particles searching, we can get the minimum mean square error of training LS-SVR, and also get the value of $\gamma$ and $\sigma$ at that point. Thus, the computational flow of PSO technique can be described in the following steps.

1. **Phase 1: Training Sample Preprocessing**

   1.1) Determine learning range of SVR-PSO. The SVR model is trained and tested by using the 214 historical data samples and 24 data sample respectively.

   1.2) Preprocessing disorder samples. Subject to random factors (e.g. transmission errors, etc.), it can not be avoided to lose data accuracy such as data errors and data loss, so the data preprocessing need to be implemented to correction errors, of which the threshold test and building cooling load theory based check are two commonly used methods.

   1.3) In order to avoid the influence of difference between factors, the parameters of input and output are normalized as Eq.(19), the learning data are scaled into $[0, 1]$:

$$y_i = \frac{2(y_i - y_{\text{min}})}{y_{\text{max}} - y_{\text{min}}} - 1 \quad (19)$$

2. **Phase 2: PSO Initialization**

   2.1) Set the iteration number $k$ to zero. Initialize randomly the swarm (containing $m$ particles) such that the position $X_0(0)$ of each particle to meet the prescribed conditions. PSO learning factors $C_1=C_2=2$, $w_{\text{max}}=0.90$, $w_{\text{min}}=0.30$, $k_{\text{max}}=1000$.

   2.2) Set the best position of each particle with its initial position, i.e., $P_{\text{best}}=P_i$ ($i=1, 2, \ldots, n$).
2.3) Set to zero the velocity vectors $V_i (i=1, 2, \ldots, n)$ that are associated with the $m$ particles.

2.4) For each candidate particle $P_i (i=1, 2, \ldots, n)$, train an SVM estimator on the corresponding augmented training set and with the estimates of the regularization and kernel parameters that are conveyed by $P_i$. Then, compute its fitness function(s) using Eq.(18).

\textbf{Phase 3: PSO Search Process}

3.1) Perform the update of each particle velocity, the best global position $G_{best}$ is chosen as the position exhibiting the minimal value of the considered fitness function over all explored trajectories, i.e.

$$G_{best}(k) \in \{P_{best}(1), P_{best}(2), \ldots, P_{best}(k)\}$$

$$= \min \{f(G_{best}(k)), f(G_{best}(k+1)), \ldots, f(G_{best}(k+n))\}$$

(20)

3.2) Compare the personal best of each particle to its current fitness, and set $P_{best}(k)$ to the better performance, i.e.

$$P_{best}(k) = \begin{cases} P_{best}(k-1), & \text{if } f(P_{best}(k)) \leq f(P_{best}(k-1)) \\ P_{best}(k), & \text{if } f(P_{best}(k)) > f(P_{best}(k-1)) \end{cases}$$

(21)

3.3) Update the speed of each particle using (1), if $V_i > V_{max}$ then $V_i = V_{max}$. If $V_i < V_{min}$ then $V_i = V_{min}$.

3.4) Update the position of each particle by means of Eq.(15). If a particle goes beyond the predefined boundaries of the search space, truncate the updating by setting the position of the particle at the space boundary and reverse its search direction. This will permit to forbid the particles from further attempting to go outside the allowed search space.

3.5) For each candidate particle $P_i (i=1, 2, \ldots, n)$, train an SVM estimator with the corresponding augmented training set and with the estimates of the regularization and kernel parameters. Compute the fitness function using Eq.(18).

\textbf{Phase 4: Convergence Check}

If the maximal number of iterations is not yet reached, return to Phase 3.

\textbf{Phase 5: Building Cooling Load Forecasting}

5.1) Forecast next time building cooling load. Taking the influencing factor vector of forecasting time into the trained LS-SVR with proper parameters, the building cooling load of next time will be forecasted.

5.2) Correct result. Because building cooling load influenced by a series of uncertain factors, it is hard to get a satisfy result absolutely using historical data. Operators should correct the forecasting result from experience. Sometimes, the change of forecasted building cooling load violates the regular pattern, and then this result is insecure usually, which can be instead by the mean of fore-and-aft loads.

\textbf{C. Models for comparison}

In order to compare the prediction results of LSSVR-PSO model, we construct several conventional building cooling load forecasting models such as BPNN predictor, autoregressive integrated moving average (ARIMA) time series predictor and Gaussian maximum likelihood (GML) approach.

\textbf{BPNN method}

The MLNN-based building cooling load prediction model composed of data processor, input layer, output layer and hidden layer, has been extensively applied for forecasting traffic parameters such as travel speed, travel time and flow. The training of the network is based on back-propagation learning algorithm, where the error calculated at the output of the network is propagated through the layers of neurons to update the weights. It can be seen in the Fig.1

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{fig1.png}
\caption{BPNN-based building cooling load prediction model}
\end{figure}

The formula of the prediction model is as following:

$$Q_i = \left[ \frac{\exp(-\sum_{j=1}^{n} w_i^j) + \mu_{o} \text{error}}{1 + \exp(-\sum_{j=1}^{n} w_i^j v_j - \mu_{o})} \right]$$

(22)

Here, $Q_i$ is the forecasting value, $w_i^j$ is the weight of connection between input layer and hidden layer, $\mu_{o}$ is the threshold of hidden layer unit, $w_i^j$ is the weight of connection between hidden layer and output layer, $\mu_{o}$ is the threshold of output unit.

\textbf{SARIMA}

The autoregressive integrated moving average (ARIMA) is one of the most popular approaches in building cooling load forecasting. Moreover, seasonal ARIMA model was used to forecasting single point short term building cooling load. The SARIMA model [21] is a popular tool in time series forecasting for data with seasonal pattern. The SARIMA $(p, d, q) \times (P, D, Q)$ process generates a time series, $\{X_t\}_{t=1,2,\ldots,n}$, with mean of Box and Jenkins time series model satisfying

$$\phi(B) \psi(B^s)(1-B)^d (1-B)^S (X_t - u)$$

(23)

$$= \theta(B) \Theta(B^s) a_t$$

where $p, d, q, P, D, Q$ are nonnegative integers; $S$ is the seasonal length; $\phi(B) = (1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)$ represents a regular autoregressive operator of order $p$, $\phi(B) = (1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_P B^P)$ is a seasonal autoregressive operator of order $P$. 

© 2010 ACADEMY PUBLISHER
\( \theta(B) = (1-\theta_1B-\theta_2B^2-\cdots-\theta_pB^p) \) denotes a regular moving average operator of order \( q \), and 
\( \Theta(B) = (1-\Theta_1B^q-\Theta_2B^{2q}-\cdots-\Theta_qB^{pq}) \) expresses a seasonal moving average operator of order \( Q \). Additionally, \( B \) indicates the backward shift order, \( d \) denotes the number of regular, \( D \) represents the number of seasonal differences, and \( a_t \) is the forecasted residual at time \( t \).

When fitting a SARIMA model to data, the first task is to estimate values of \( a \) and \( D \), the orders of differencing needed to make the series stationary and to remove most of the seasonality. The suitable values of \( p \), \( P \), \( q \) and \( Q \) can be evaluated by the autocorrelation function (ACF) and partial autocorrelation function (PACF) of the differenced series.

\textbf{-Gaussian maximum likelihood (GML) approach}

Lin’s GML-based model makes use of both historical and real-time information in an integrated way by using two key variables: cooling load and cooling load increment [22]. Let \( X_i \) (\( i = 0, 1, 2, \ldots, n \)) be consecutive observations of the building cooling load obtained at time \( i \). Let \( Y_i = X_i - X_{i-1} \) be the flow increment. Assuming that these two variables are normally distributed, an estimate for the flow in the next period \( i \) was derived by maximizing the product of the two probability functions of \( X_i \) and \( Y_i \), resulting in the simple model:

\[ \hat{x}_i = \frac{\sigma_{x,i}^2(\mu_{x,i} + \mu_{y,i}) + \sigma_{y,i}^2 \mu_{x,i}}{\sigma_{x,i}^2 + \sigma_{y,i}^2}, \quad i = 1, 2, \ldots, n \quad (24) \]

Where \( \mu_{x,i} \) and \( \sigma_{x,i}^2 \) are the mean and variance of \( X_i \), respectively, and \( \mu_{y,i} \) and \( \sigma_{y,i}^2 \) are the mean and variance of \( Y_i \), respectively. These means and variances are estimated from the empirical data.

Notice in (24) that flow at time \( i \) is predicted based on the observed flow at time \( i-1 \), on the historical mean and variance of the flows at time \( i \), and on the historical mean and variance of the flow increments related to times \( i \) and \( i-1 \). More details of Lin’s GML-based model can be found in [22].

\textbf{IV. EXPERIMENT RESULTS AND ANALYSIS}

\textbf{A. Data Set Description}

The hourly climate data and building cooling load from May to September is considered in this paper. The input parameters of SVM such as dry-bulb temperature, relative humidity and solar radiation intensity are taken from the climate database of Guangzhou in the typical meteorology year. DeST is used to calculate the office building’s hourly cooling loads, which are taken as the basic values to compare with the predicted values from LS-SVR.

The data were sampled per hour in every day. Since building cooling load data are always sampled with noise, the data preprocessing need to be implemented to correct errors, of which the threshold test and building HVAC theory-based check are two commonly used methods. At last, the data should be normalized treatment to improve the efficiency of computation.

The collected data were divided into training sets and testing sets. The sample data gathered for simulation in this paper are composed of 218 samples. We take 204 samples to carry on the SVR-PSO training for determining the SVR-PSO estimator model parameters and use 14 samples of the rest to evaluate the performance of the optimal prediction model before its application. Cross-validation method was applied to obtain the optimal parameters of model.

\textbf{B. SVR model Parameters Identification}

This paper adopts the improved SMO algorithm to train SVM and the Gaussian kernel is selected as the kernel function. In training stage, the training data set are fed into the LSSVM-PSO model running in SIMULINK environment. The parameter selection method is programmed in Python based on regression.py in LIBSVM-2.8. The training of SVR is finished by LIBSVM so does the course of getting the regression function. The structural risk minimization principle is employed to minimize the training error. With the trained SVM model, the estimation result of testing data is gained. The two parameters, \( \gamma \) and \( \sigma \) of SVM model are adjusted by PSO algorithm. While testing errors improvement occurs. Here, the range constrains of the parameters are set as \( \gamma \in [0.1,1000] \), and \( \sigma \in [0.01,10] \). Then, the adjusted parameters with minimum testing error are selected as the most appropriate parameters. The method proposed in this paper is compared with grid search. The result is shown in Tab.1. Obviously, the proposed methodology has better performance not only on searching time but also on the identification accuracy, which provides an effective approach for parameter selection of LS-SVR. The error convergent cure during training process is shown in Fig.2, where X axis is the iteration of training, Y axis is model error. As the iteration number became larger and larger, the error converged quickly. Compared with Grid search approach, the PSO optimization has better performance.
SARIMA model and GML model are also adopted to one-step forecasting. The architecture of BPNN was composed as follows: 8 neurons in the input layer, single hidden layer with 6 neurons and 1 output neuron. Tangent sigmoid function and linear transfer function are used for activation function in the hidden and output nodes. By the statistical package, the appropriate model for building cooling load used in this study is SARIMA(1, 0,1)×(0, 0,1)12.

Fig. 3 shows comparison of the actual and different forecasted values respectively, the SVR-PSO forecasting curve is very closer to the actual data, indicating that the regression result is good. The predictive curve obtained by using the Gaussian standard deviation kernel function differs little from the actual value and so, has predictive ability.

The mean average percent error (MAPE) values of these forecasts are shown in Tab. 2. See in Tab. 2 that BPNN and ARIMA models clearly presented higher MAPE values in different time tests. It can also be noticed in Tab. 2 that, the GML approach presented the higher overall prediction accuracy with an average MAPE of 5.6%, which supports the finds of previous studies [22]. The SVR-PSO has the second best overall prediction accuracy with an average MAPE of 5.8%.

V. CONCLUSION

Accurate forecast of building air-conditioning load, especially the short term cooling load prediction, provides an approach to integrate and optimize the heating, ventilating, and air-conditioning (HVAC) system cooling supply system efficiently. Many forecasting methodologies have been proposed to deal with nonlinearity of building cooling load. This study presented a LSSVR-PSO model for forecasting building cooling load in the Science Building at University town of Guangzhou. In this paper, LSSVR was adopted to establish the building cooling load prediction model. And due to the importance of parameters optimization in SVR model, the PSO algorithm was used to obtain the free

<table>
<thead>
<tr>
<th>Optimize method</th>
<th>SVM type</th>
<th>Kernel Func.</th>
<th>γ</th>
<th>σ2</th>
<th>b</th>
<th>SVs</th>
<th>Time</th>
<th>APE</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid search</td>
<td>LS-SVR</td>
<td>RBF</td>
<td>94</td>
<td>15.83</td>
<td>-2.941</td>
<td>27</td>
<td>9’23</td>
<td>2.364%</td>
<td>1.607%</td>
</tr>
<tr>
<td>PSO</td>
<td>LS-SVR</td>
<td>RBF</td>
<td>315.4</td>
<td>23.62</td>
<td>-2.762</td>
<td>26</td>
<td>2’58</td>
<td>1.054%</td>
<td>0.926%</td>
</tr>
</tbody>
</table>

Tab.1 Performance comparison between PSO and Grid search

![Fig.3 Actual and predicted cooling load value of different predictor](image)

Tab.2 Performance Comparison of different estimator

<table>
<thead>
<tr>
<th>No.</th>
<th>SVR-PSO</th>
<th>BPNN</th>
<th>ARIMA</th>
<th>GML</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.8</td>
<td>5.1</td>
<td>5.4</td>
<td>5.2</td>
</tr>
<tr>
<td>2</td>
<td>5.2</td>
<td>5.7</td>
<td>5.9</td>
<td>5.1</td>
</tr>
<tr>
<td>3</td>
<td>7.2</td>
<td>7.6</td>
<td>8.0</td>
<td>6.8</td>
</tr>
<tr>
<td>4</td>
<td>4.6</td>
<td>5.0</td>
<td>5.4</td>
<td>4.3</td>
</tr>
<tr>
<td>5</td>
<td>6.4</td>
<td>6.9</td>
<td>7.2</td>
<td>6.2</td>
</tr>
<tr>
<td>6</td>
<td>8.6</td>
<td>9.1</td>
<td>9.4</td>
<td>8.2</td>
</tr>
<tr>
<td>7</td>
<td>3.8</td>
<td>4.3</td>
<td>4.5</td>
<td>4.0</td>
</tr>
<tr>
<td>8</td>
<td>5.5</td>
<td>5.9</td>
<td>6.3</td>
<td>5.2</td>
</tr>
<tr>
<td>Mean</td>
<td>5.8</td>
<td>6.2</td>
<td>6.5</td>
<td>5.6</td>
</tr>
</tbody>
</table>
parameters. PSO optimization requires only simple mathematical operators. This algorithm is simple to implement and effective, and is inexpensive in terms of memory and time required. This approach provides solutions with better quality within a reasonable time limit. The experimental results indicate that the SVR-PSO model outperformed the BPNN and ARIMA models in terms of forecasting accuracy. The test results demonstrated the proposed forecasting method could provide a considerable improvement of the forecasting accuracy for building cooling load prediction.

Future research should look into some other advanced optimization algorithms for online update SVR parameters can be considered to improve the forecasting efficiency and real time performance.

REFERENCES

[23] Liang Zhao, Fei-Yue Wang, Short-term traffic flow prediction based on ratio-median lengths of intervals two-factors high-order fuzzy time series, Proc. of Conf. on Vehicular Electronics and Safety, Dec. 2007, pp.1–7