energies

MDPI

Article

Short-Term Load Forecasting Using Adaptive Annealing Learning Algorithm Based Reinforcement Neural Network

Cheng-Ming Lee [1] and Chia-Nan Ko [2,*]

[1] Department of Digital Living Innovation, Nan Kai University of Technology, Tsaotun, Nantou 542, Taiwan; t104@nkut.edu.tw

[2] Department of Automation Engineering, Nan Kai University of Technology, Tsaotun, Nantou 542, Taiwan

* Correspondence: t105@nkut.edu.tw

Academic Editor: Javier Contreras

Received: 28 September 2016; Accepted: 18 November 2016; Published: 25 November 2016

Abstract: A reinforcement learning algorithm is proposed to improve the accuracy of short-term load forecasting (STLF) in this article. The proposed model integrates radial basis function neural network (RBFNN), support vector regression (SVR), and adaptive annealing learning algorithm (AALA). In the proposed methodology, firstly, the initial structure of RBFNN is determined by using an SVR. Then, an AALA with time-varying learning rates is used to optimize the initial parameters of SVR-RBFNN (AALA-SVR-RBFNN). In order to overcome the stagnation for searching optimal RBFNN, a particle swarm optimization (PSO) is applied to simultaneously find promising learning rates in AALA. Finally, the short-term load demands are predicted by using the optimal RBFNN. The performance of the proposed methodology is verified on the actual load dataset from the Taiwan Power Company (TPC). Simulation results reveal that the proposed AALA-SVR-RBFNN can achieve a better load forecasting precision compared to various RBFNNs.

Keywords: short-term load forecasting; radial basis function neural network; support vector regression; particle swarm optimization; adaptive annealing learning algorithm

1. Introduction

Load forecasting is a crucial issue in power planning, operation, and control [1–4]. A short-term load forecasting (STLF) can be used for power maintenance scheduling, security assessment, and economic dispatch. Thus, in order to strengthen the performance of the power system, improving the load forecasting accuracy is very important [5]. An accurate forecast can reduce costs and maintain security of a power system.

In recent years, various mathematical and statistical methods have been applied to improve the accuracy of STLF. These models are roughly classified as traditional approaches and artificial intelligence (AI) based methods. Traditional approaches include exponential smoothing [6], linear regression methods [7], Box-Jenkins ARMA approaches [8], and Kalman filters [9]. In general, the traditional methods cannot correctly indicate the complex nonlinear behavior of load series. Gouthamkumar et al. [10] proposed a non-dominated sorting disruption-based gravitational search algorithm to solve fixed-head and variable-head short-term economical hydrothermal scheduling problems. With the development in AI techniques, fuzzy logic [11], PSO [12], SVM [13], and singular spectrum analysis and nonlinear multi-layer perceptron network [14] have been successfully used for STLF. AI methods have an excellent approximation ability on nonlinear functions. Therefore, it can deal with nonlinear and complex functions in the system. However, a single method cannot predict STLF efficiently.

Hybrid methods are developed to utilize the unique advantages of each approach. Adaptive ANNs for short-term load forecasting are proposed in [15], in which a PSO algorithm is employed to adjust the network's weights in the training phase of the ANNs. A modified version of the ANN already proposed for the aggregated load of the interconnected system is employed to improve the forecasting accuracy of the ANN [16]. A strategy using support vector regression machines for short-term load forecasting is proposed in [17]. A STLF algorithm based on wavelet transform, extreme learning machine (ELM) and modified artificial bee colony (MABC) algorithm is presented in [18].

As RBFNN has a single hidden layer and fast convergence speed, RBFNN has been successfully used for STLF [19,20]. When using RBFNN, one must determine the hidden layer nodes, the initial kernel parameters, and the initial network weights. A systematic approach must be established to determine the initial parameters of RBFNN. Typically, these parameters are obtained according to the designer experience, or just a random choice. However, such improper initialization usually results in slow convergence speed and poor performance of the RBFNN. An SVR method with Gaussian kernel function is adopted to determine the initial structure of the RBFNN for STLF [21].

In the training procedure, learning rates serve as an important role in the procedure of training RBFNN. The learning rate would depend on the characteristic state of inputs and outputs, in which the learning rate would be increased or decreased to match training data. Through trial and error, the learning rate is chosen to be a time-invariant constant [22,23]. However, there also exist several unstable or slow convergence problems. Many studies have been dedicated to improving the stability and convergence speed of the learning rates [24]. However, the deterministic methods for exploring appropriate learning rates are often tedious.

Recently, efficient learning algorithms for RBFNN have been developed. Besides, researchers have proposed sequential learning algorithms for resource allocation networks to enhance the convergence of the training error and computational efficiency [25]. A reinforcement learning method based on adaptive simulated annealing has been adopted to improve a decision making test problem [26]. In the literature, the learning algorithms for reduction of the training data sequence with significant information generates less computation time for a minimal network and achieves better performance. Motivated by the these learning methodologies, an adaptive learning algorithm is applied to the annealing learning procedure to promote the performance of RBFNNs. Adaptive annealing learning algorithm-based robust wavelet neural networks have been used to approximate function with outliers [27]. In [28], the author proposed time-varying learning algorithm based neural networks to identify nonlinear systems. Ko [29] proposed an adaptive annealing learning algorithm (AALA) to push forward the performance of RBFNN.

In this research, AALA is adopted to train the initial structure of RBFNN using an SVR method. In AALA, PSO approach is applied to simultaneously determine a set of suitable learning rates to improve the training RBFNN performance of STLF.

This paper is organized as follows. Section 2 introduces the architecture of RBFNN. In Section 3, the proposed algorithm, AALA-SVR-RBFNN with an adaptive annealing learning algorithm is introduced. In Section 4, simulation results for three different cases are presented and discussed. Section 5 provides conclusions for the proposed AALA-SVR-RBFNN.

2. Architecture of RBFNN

Generally, an RBFNN has a feed forward architecture with three layers: input layer, hidden layer, and output layer. The basic structure of an RBFNN is shown in Figure 1. The output of the RBFNN can be expressed as follows

$$\hat{y}_j(t+1) = \sum_{i=1}^{L} G_i \omega_{ij} = \sum_{i=1}^{L} \omega_{ij} \exp\left(-\frac{\| x - x_i^c \|^2}{2w_i^2} \right) \text{ for } j = 1, 2, \cdots, p \quad (1)$$

where $x(t) = [x_1(t) \cdots x_m(t)]^T$ is the input vector, $\hat{y}(t) = \left[\hat{y}_1(t) \cdots \hat{y}_p(t)\right]^T$ is the output vector of RBFNN, ω_{ij} is the vector of the synaptic weights in the output layer, G_i is the vector of the Gaussian function, denote the RBFNN activation function of the hidden layer, x_i^c and w_i are the vector of the centers and widths in G_i, respectively, and L is the number of neurons in the hidden layer.

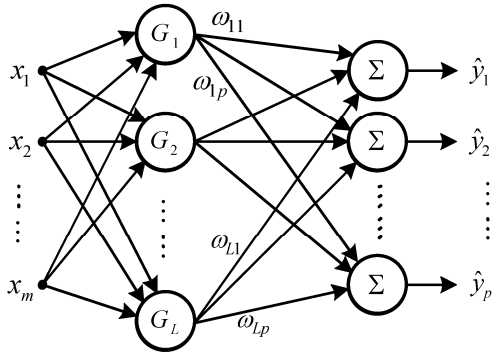

Figure 1. The structure of RBFNN.

In the training procedure, the initial values of parameters in (1) must be selected first. Then a training method is used to adjust these values iteratively to obtain their optimal combination. However, there is no way to systematically select the initial value of parameters. In the following section, an SVR is applied to perform this work.

3. AALA-SVR-RBFNN for STLF

3.1. SVR-Based Initial Parameters Estimation of RBFNN

An SVR-based algorithm is able to approximate an unknown function from a set of data, $(x^{(k)}, y^{(k)})$, $k = 1, 2, \cdots, N$.

In SVR, the Gaussian function is adopted as the kernel function [30]. Therefore, the approximating function can be rewritten as

$$f(x, \lambda) = \sum_{l=1}^{N_{SV}} \alpha_l \exp\left(-\frac{\| x - x_l \|^2}{2{\sigma_l}^2}\right) + b \tag{2}$$

where N_{SV} is the number of *support vectors* (*SV*s) and x_l are *SV*s. Comparing (2) with (1), N_{SV}, l, α_l, σ_l, and x_l in (2) can be considered to be the L, i, ω_{ij}, w_i, and x_i^c in (1), respectively. From the above derivation, the parameters of RBFNN in (1) can be obtained through the above $\varepsilon -$ SVR method.

3.2. AALA-Based SVR-RBFNN

When computing the initial parameters of the SVR-RBFNN, one must establish a learning algorithm to get the optimal parameters of SVR-RBFNN. Based on time-varying learning rates, an AALA is used to train the SVR-RBFNN for conquering the drawbacks of low convergence and local minimum faced by the back-propagation learning method [31]. A cost function for the AALA is defined as

$$\Gamma_j(h) = \frac{1}{N}\sum_{k=1}^{N} \rho\left[e_j^{(k)}(h); \xi(h)\right] \text{ for } j = 1, 2, \cdots, p \tag{3}$$

where

$$e_j^{(k)}(h) = y_j^{(k)} - \hat{f}_j(x^{(k)}) = y_j^{(k)} - \sum_{i=1}^{L} \omega_{ij} \exp\left(-\frac{\| x^{(k)} - x_i^c \|^2}{2{w_i}^2}\right) \tag{4}$$

h denotes the epoch number, $e_j^{(k)}(h)$ denotes the error between the jth desired output and the jth output of RBFNN at epoch h for the kth input-output training data, $\xi(h)$ denotes a deterministic annealing schedule acting like the cut-off point, and $\rho(\cdot)$ denotes a logistic loss function given by

$$\rho\left[e_j^{(k)};\xi\right] = \frac{\xi}{2}\ln\left[1+\frac{\left(e_j^k\right)^2}{\xi}\right] \text{ for } j=1,2,\cdots,p \tag{5}$$

The root mean square error (*RMSE*) is adopted to assess the performance of training RBFNN, given by

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(e_j^{(k)}\right)^2} \text{ for } j=1,2,\cdots,p \tag{6}$$

According to the gradient-descent learning algorithms, the synaptic weights of ω_{ij}, the centers of x_i^c, and the widths of w_i in Gaussian function are adjusted as

$$\Delta\omega_{ij} = -\gamma_\omega\frac{\partial\Gamma_j}{\partial\omega_{ij}} = -\frac{\gamma_\omega}{N}\sum_{k=1}^{N}\phi_j\left(e_j^{(k)};\xi\right)\frac{\partial e_j^{(k)}}{\partial\omega_{ij}} \tag{7}$$

$$\Delta x_i^c = -\gamma_c\frac{\partial\Gamma_j}{\partial x_i^c} = -\frac{\gamma_c}{N}\sum_{j=1}^{p}\sum_{k=1}^{N}\phi_j\left(e_j^{(k)};\xi\right)\frac{\partial e_j^{(k)}}{\partial x_i^c} \tag{8}$$

$$\Delta w_i = -\gamma_w\frac{\partial\Gamma_j}{\partial w_i} = -\frac{\gamma_w}{N}\sum_{j=1}^{p}\sum_{k=1}^{N}\phi_j\left(e_j^{(k)};\xi\right)\frac{\partial e_j^{(k)}}{\partial w_i} \tag{9}$$

$$\phi_j\left(e_j^{(k)};\xi\right) = \frac{\partial\rho\left(e_j^{(k)};\xi\right)}{\partial e_j^{(k)}} = \frac{e_j^{(k)}}{1+\left(e_j^{(k)}\right)^2/\xi(h)} \tag{10}$$

where γ_ω, γ_c, and γ_w are the learning rates for the synaptic weights ω_{ij}, the centers x_i^c, and the widths w_i, respectively; and $\phi(\cdot)$ is usually called the influence function. In ARLA, the annealing schedule $\xi(h)$ has the capability of progressive convergence [32,33].

In the annealing schedule, complex modifications to the sampling method have been proposed, which can use a higher learning rate to reduce the simulation cost [34–37]. Based on this concept, the non-uniform sampling rates of AALA are used to train BRFNN in this work. According to the relative deviation of the late epochs, the annealing schedule $\xi(h)$ can be adjusted. The annealing schedule is updated as

$$\xi(h) = \Psi\Delta\cdot\frac{h}{h_{\max}-1} \text{ for epoch } h \tag{11}$$

$$\Delta = -2\log\left(\frac{S}{\overline{RMSE}}\right) \tag{12}$$

$$S = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(RMSE_i-\overline{RMSE}\right)^2} \tag{13}$$

where Ψ is a constant, S is the standard deviations, and $\overline{RMSE}$ is the average of $RMSE$ (6) for m late epochs.

When the learning rates keep constant, choosing an appropriate learning rates γ_ω, γ_c, and γ_w is tedious; furthermore, several problems of getting stuck in a near-optimal solution or slow convergence still exist. Therefore, an AALA is used to overcome the stagnation in the search for a global optimal solution. At the beginning of the learning procedure, a large learning rate is chosen in the search space.

Once the algorithm converges progressively to the optimum, the evolution procedure is gradually tuned by a smaller learning rate in later epochs. Then, a nonlinear time-varying evolution concept is used in each iteration, in which the learning rates γ_ω, γ_c, and γ_w have a high value $\gamma_{\max}$, nonlinearly decreases to $\gamma_{\min}$ at the maximal number of epochs, respectively. The mathematical formula can be expressed as

$$\gamma_\omega = \gamma_{\min} + (epoch\,(h))^{p\omega}\,\Delta\gamma \tag{14}$$

$$\gamma_c = \gamma_{\min} + (epoch\,(h))^{pc}\,\Delta\gamma \tag{15}$$

$$\gamma_w = \gamma_{\min} + (epoch\,(h))^{pw}\,\Delta\gamma \tag{16}$$

$$\Delta\gamma = (\gamma_{\max} - \gamma_{\min}) \tag{17}$$

$$epoch(h) = \left(1 - \frac{h}{epoch_{\max}}\right) \tag{18}$$

where $epoch_{\max}$ is the maximal number of epochs and h is the present number of epochs. During the updated process, the performance of RBFNN can be improved using suitable functions for the learning rates of γ_ω, γ_c, and γ_w. Furthermore, simultaneously determining the optimal combination of $p\omega$, pc, and pw in (14) to (16) is a time-consuming task. A PSO algorithm with linearly time-varying acceleration coefficients will be used to obtain the optimal combination of $(p\omega,\ pc,\ pw)$. A PSO algorithm with linearly time-varying acceleration coefficients for searching the optimal combination of $(p\omega,\ pc,\ pw)$ is introduced in the following section.

3.3. Particle Swarm Optimization

PSO is a population-based stochastic searching technique developed by Kennedy and Eberhart [38]. The searching process behind the algorithm was inspired by the social behavior of animals, such as bird flocking or fish schooling. It is similar to the continuous genetic algorithms, in which it begins with a random population matrix and searches for the optimum by updating generations. However, the PSO has no evolution operations such as crossover and mutation. The potential of this technique makes it a powerful optimization tool which has been successfully applied to many fields. Nowadays, PSO has been developed to be a real competitor with other well-established techniques for population-based evolutionary computation [39,40].

In this paper, the PSO method is adopted to find an optimal combination $(p\omega,\ pc,\ pw)$ of learning rates in (14) to (16). When applying the PSO method, possible solutions must be encoded into particle positions and a fitness function must be chosen. In the optimizing procedure, the goal is to minimize the error between desired outputs and trained outputs, and then root mean square error (*RMSE*) will be defined as the fitness function.

In the PSO, a particle position is represented as

$$\mathbf{P} = [p_1, p_2, p_3] = [p\omega,\ pc,\ pw] \tag{19}$$

At each iteration (generation), the particles update their velocities and positions based on the local best and global best solutions as follows [40]:

$$\mathbf{V}(k+1) = \lambda(k+1)\cdot\mathbf{V}(k) + \mathrm{c}_1(k+1)\cdot\mathrm{r}_1\cdot(\mathbf{P}^{\mathrm{lbest}}(k) - \mathbf{P}(k)) + \mathrm{c}_2(k+1)\cdot\mathrm{r}_2\cdot(\mathbf{P}^{\mathrm{gbest}}(k) - \mathbf{P}(k)) \tag{20}$$

$$\mathbf{P}(k+1) = \mathbf{P}(k) + \mathbf{V}(k+1) \tag{21}$$

where $\mathbf{V}(k)$ and $\mathbf{V}(k+1)$ denote the particle velocities at iterations k and $(k+1)$, respectively, $\mathbf{P}(k)$ and $\mathbf{P}(k+1)$ denote the particle positions at iteration k and $(k+1)$, respectively, $\lambda(k+1)$ denotes the inertia weight at iteration $(k+1)$, r_1 and r_2 are random numbers between 0 and 1, $c_1(k+1)$ is the cognitive parameter, $c_2(k+1)$ is the social parameter at iteration $(k+1)$, $\mathbf{P}^{lbest}(k)$ is the local best solution at iteration k, and $\mathbf{P}^{gbest}$ is the global best solution of the group.

Considering the computational efficiency, a linearly adaptable inertia weight [38] and linearly time-varying acceleration coefficients [40] over the evolutionary procedure of PSO method are adopted in this paper. The inertia weight λ starts with a high value $\lambda_{\max}$ and linearly decreases to $\lambda_{\min}$ at the maximal number of iterations. The cognitive parameter c_1 starts with a high value $c_{1\max}$ and linearly decreases to $c_{1\min}$. Whereas the social parameter c_2 starts with a low value $c_{2\min}$and linearly increases to $c_{2\max}$. Therefore, the inertia weight $\lambda(k+1)$ and the acceleration coefficients $c_1(k+1)$ and $c_2(k+1)$ can be expressed as follows:

$$\lambda(k+1) = \lambda_{\max} - \frac{\lambda_{\max} - \lambda_{\min}}{iter_{\max}} \cdot iter \tag{22}$$

$$c_1(k+1) = c_{1\max} - \frac{c_{1\max} - c_{1\min}}{iter_{\max}} \cdot iter \tag{23}$$

$$c_2(k+1) = c_{2\max} - \frac{c_{2\max} - c_{2\min}}{iter_{\max}} \cdot iter \tag{24}$$

where $iter_{\max}$ is the maximal number of iterations (generations) and $iter$ is the current number of iterations.

3.4. Procedure of Hybrid Learning Algorithm

The proposed AALA-SVR-RBFNN using PSO can be summarized as follows:

Algorithm 1. AALA-SVR-RBFNN

Step 1: Given a set of input-output data, $(x^{(k)}, y^{(k)})$, $k = 1, 2, \cdots, N$ for STLF.

Step 2: Formulate and solve an SVR problem as described in Section 3.1 to determine the initial structure of the RBFNN in (1) based on the given data obtained in Step 1..

Step 3: Adopt PSO method to generate different optimal sets of $(p\omega,\ pc,\ pw)$.

Step 4: $K = 0$

Step 5: $K = K + 1$

Step 6: Produce initial populations of position and velocity particles randomly within the feasible range.

Step 7: Perform the AALA.

- Step 7-1: Calculate the corresponding errors by (4) for all training data.
- Step 7-2: Determine the values of the AALA schedule $\xi(h)$ in (11) for each epoch.
- Step 7-3: Update the synaptic weights ω_{ij}, the centers x_i^c, and the widths w_i of Gaussian functions iteratively according to (7) through (10) and (14) through (16).
- Step 7-4: Repeat the procedure from Step 7-1 to Step 7-3 until the current number h of epochs reaches $epoch_{\max}$.

Step 8: Evaluate fitness value for each population using the fitness function (6).

Step 9: Select the local best for each particle by ranking the fitness values. If the best value of all current local best solutions is better than the previous global best solution, then update the value of the global best solution.

Step 10: Update the velocity and position of each particle by using linearly the updated inertia weight, the local best particle, and the global best particle.

Step 11: Repeat the procedure in Step 7 through Step 10 until the current number of iterations reaches the maximal iteration number in PSO.

Step 12: If $K < m$ (m is the number of times for performing PSO method) then go to Step 6. Otherwise, determine the average optimal value of $(p\omega,\ pc,\ pw)$ in (14) to (16).

Step 13: Use the learning rates with the average optimal value of $(p\omega,\ pc,\ pw)$ in (14) to (16) to train the proposed RBFNN.

In the Algorithm 1, the solution obtained in Step 13 is considered the average optimal value of $(p\omega,\ pc,\ pw)$ decided by PSO through M times independently; meanwhile the optimal structure of the proposed AALA-SVR-RBFNN is determined.

The flowchart of AALA-SVR-RBFNN using PSO is illustrated in Figure 2. The solution of the average optimal value of $(p\omega,\ pc,\ pw)$ is determined using PSO through m times independently. Then, the optimal structure of the proposed AALA-SVR-RBFNN is obtained.

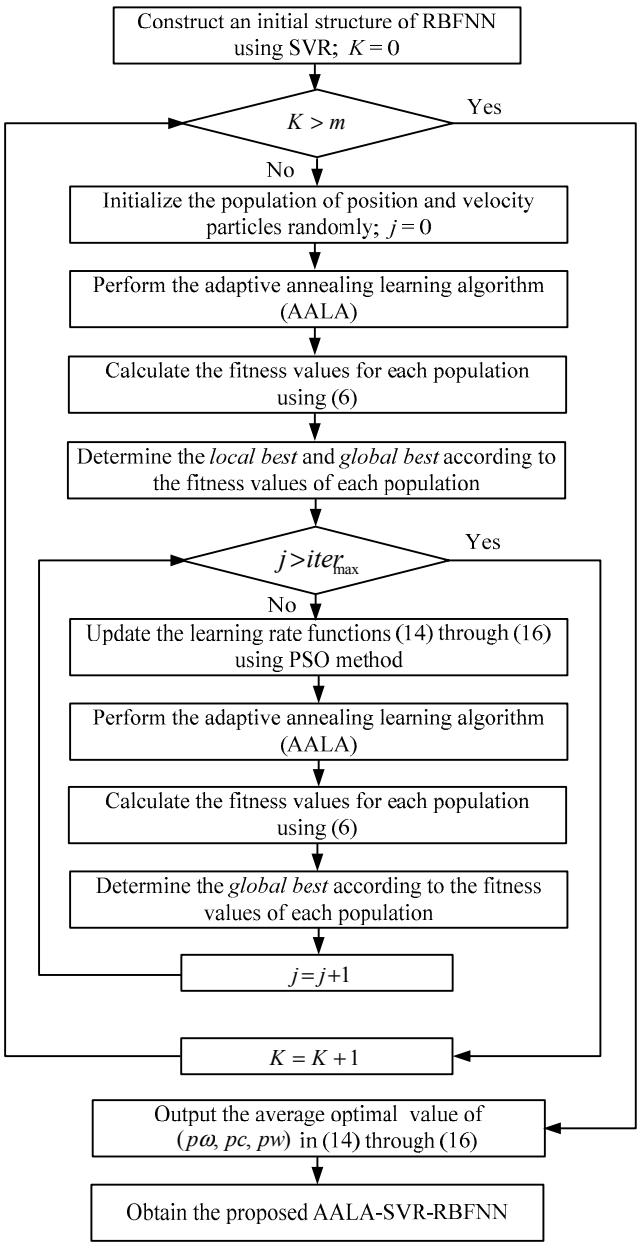

Figure 2. The flowchart of AALA-SVR-RBFNN.

4. Case Studies

The 24-h-ahead forecasting performance of the proposed AALA-SVR-RBFNN is verified on a real-word load dataset from Taiwan Power Company (TPC) in 2007. Three different patterns of load data, such as the working days (Monday through Friday), the weekends (Saturday), and holidays (Sunday and national holiday), are utilized to evaluate the effectiveness of the proposed algorithm. Table 1 lists the periods of the training and testing load data on TPC. The forecasting performance

is also compared to those of DEKF-RBFNN [19], GRD-RBFNN [19], SVR-DEKF-RBFNN [19], and ARLA-SVR-RBFNN. All our simulations are carried out in Matlab 2013a using a personal computer with Intel i3-6100 and 4G RAM, Windows 7 as operating system.

The initial parameters in the simulations must be specified first. For the PSO parameters, the population size is set to be 40 and the maximal iteration number is set to be 200. Meanwhile, the number of generating different optimal sets of $(p\omega, pc, pw)$ is set to 10 $(m = 10)$. The values of $(p\omega, pc, pw)$ in learning rate functions (14) to (16) are all set to real numbers in the interval $[0.1, 5]$. Furthermore, the value of $\gamma_{\max}$ is chosen to be 2.0 and the value of $\gamma_{\min}$ is chosen to be 0.05.

Table 1. The periods of the training and testing load data on TPC in 2007.

Case	Data Type	Training Data	Testing Data
1	Weekdays	2 February–15 March	16 March
2	Weekends	12 May–27 October	3 November
3	Holidays	1 July–9 December	16 December

In order to evaluate the forecasting performance of the models, two forecast error measures, such as mean absolute percentage error (*MAPE*), and standard deviation of absolute percentage error (*SDAPE*), which are utilized for model evaluation, and their definitions are shown as follows:

$$MAPE = \frac{1}{N}\sum_{k=1}^{N}\frac{\left|A^{(k)} - F^{(k)}\right|}{A^{(k)}} \times 100 \tag{25}$$

$$SDAPE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\frac{\left|A^{(k)} - F^{(k)}\right|}{A^{(k)}} \times 100 - MAPE\right)^2} \tag{26}$$

where N is the number of forecast periods, $A^{(k)}$ is the actual value, and $F^{(k)}$ is the forecast value. Moreover, the *RMSE* in (6) is employed to verify the performance of training RBFNN.

Case 1: Load prediction of weekdays

The training hourly actual load data are shown in Table 1 and Figure 3. After 1000 training epochs, the initial parameters of RBFNN are determined by using SVR. The value of L in (1) is found to be 5 for parameters $C = 1$ and $\varepsilon = 0.05$ in SVR. Meanwhile, the average optimal learning rate set of $(p\omega, pc, pw)$ is determined by PSO, which is found to be (2.1711, 1.4654, 3.6347).

Table 2 lists the comparison results of *RMSE* in (6) between ARLA and AALA. In Table 2, it can be observed that AALA is superior to ARLA. After training, the proposed approach is evaluated on 24-h-ahead load forecasting on 16 March 2007. From Figure 4, it can be observed that the predicted values of the proposed AALA-SVR-RBFNN are close to the actual values.

The comparisons of *MAPE* and *SDAPE* using the five prediction models are shown in Table 3. From the comparisons, the value of *MAPE* of the proposed method is the smallest among the prediction approaches and has improvements of 48.10%, 51.19%, 31.67%, and 4.65% over DEKF-RBFNN [19], GRD-RBFNN [19], SVR-DEKF-RBFNN [19], and ARLA-SVR-RBFNN, respectively. Moreover, the *SDAPE* value of the proposed AALA-SVR-RBFNN is 0.34%, less than those obtained by the four methods.

Table 2. Results of *RMSE* in (6) of the ARLA ($0.002 \leq \gamma \leq 2$) and AALA after 1000 training epochs for three cases.

Case	ε	AALA	ARLA (γ)									
			2	**1.5**	**1**	**0.5**	**0.1**	**0.05**	**0.02**	**0.01**	**0.005**	**0.002**
1	0.05	**0.0072774**	0.010025	0.009653	0.009227	0.008514	0.008452	0.008386	**0.008379**	0.008426	0.008554	0.008808
2	0.06	**0.0109520**	0.015183	0.014111	0.012962	0.011622	**0.011525**	0.011596	0.011681	0.011708	0.011861	0.012267
3	0.05	**0.0097116**	0.012594	0.011968	0.01132	0.011636	0.010698	0.010365	0.010186	**0.010104**	0.010127	0.01018

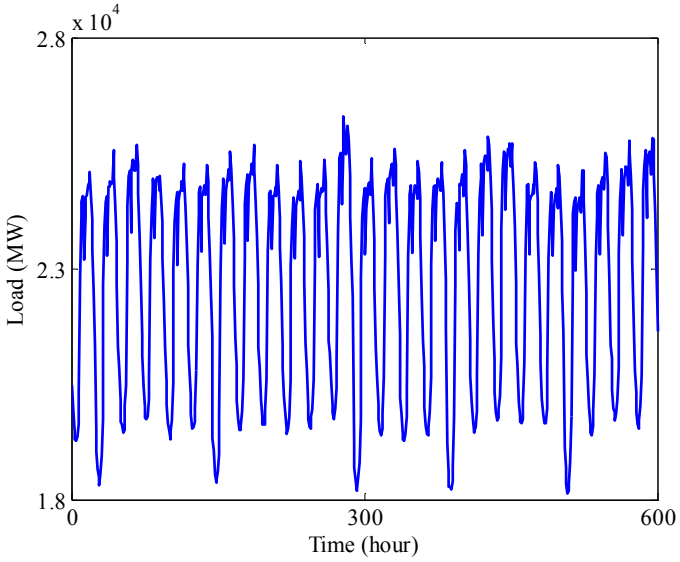

Figure 3. The training hourly actual load data in Case 1.

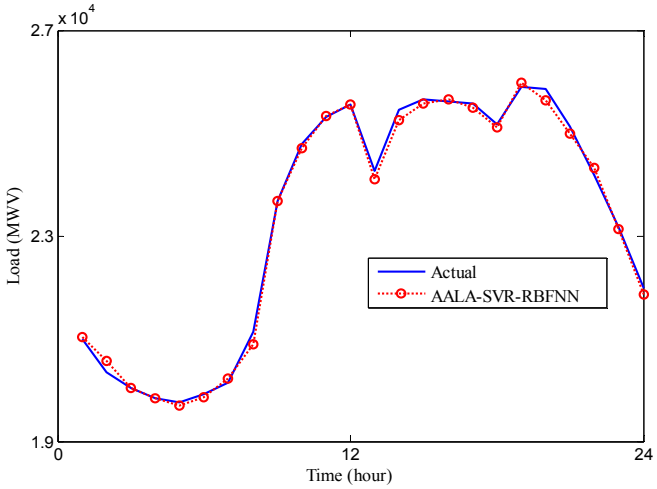

Figure 4. The forecasting results of the proposed AALA-SVR-RBFNN in Case 1.

Table 3. *MAPE* (%) and *SDAPE* (%) results for prediction methods in Case 1.

Method	*MAPE*	*SDAPE*
DEKF-RBFNN [19]	0.79	0.6
GRD-RBFNN [19]	0.84	0.6
SVR-DEKF-RBFNN [19]	0.6	0.37
ARLA-SVR-RBFNN	0.43	0.34
AALA-SVR-RBFNN	0.41	0.34

Case 2: Load prediction of weekends

The training hourly load data are shown in Table 1 and Figure 5. After 1000 training epochs, the initial parameters of RBFNN are obtained by using SVR. The value of L in (1) is found to be 7 for parameters $C = 1$ and $\varepsilon = 0.06$ in SVR. Meantime, the average optimal learning rate set of $(p\omega, pc, pw)$ is determined by PSO, which is found to be (2.6007, 0.7976, 4.0978).

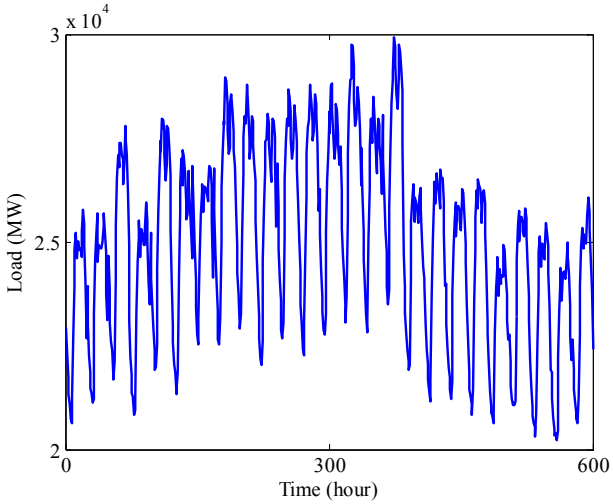

Figure 5. The training hourly actual load data in Case 2.

Table 2 shows the *RSME* in (6) ARLA and AALA. As seen from Table 2, the AALA can obtain an encouraging result than ARLA. After training, the proposed approach is evaluated on 24-h-ahead load forecasting. Figure 6 shows that the predicted values of the proposed method are very close to the actual values.

The comparison results of *MAPE* and *SDAPE* using the five prediction models are shown in Table 4. From the comparisons, the proposed AALA-SVR-RBFNN has the minimum value of *MAPE*. It proves the *MAPE* of AALA-SVR-RBFNN over DEKF-RBFNN [19], GRD-RBFNN [19], SVR-DEKF-RBFNN [19], and ARLA-SVR-RBFNN by 58.76%, 62.63%, 44.33%, and 24.53%. Moreover, the value of *SDAPE* of the proposed algorithm is smaller than those of other four methods.

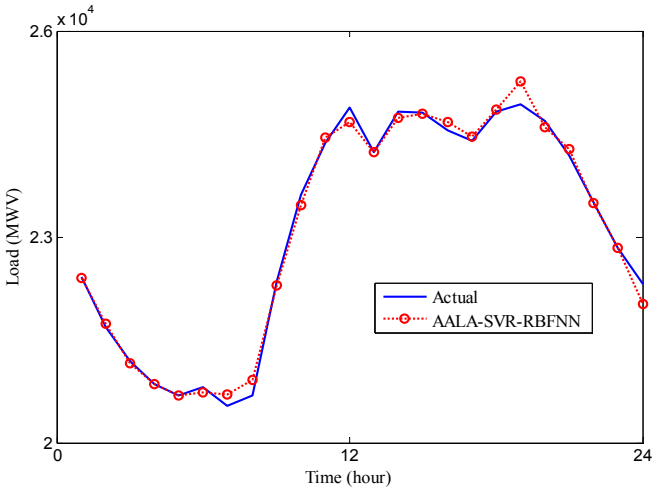

Figure 6. The forecasting results of the proposed AALA-SVR-RBFNN) in Case 2.

Table 4. *MAPE* (%) and *SDAPE* (%) results for prediction methods in Case 2.

Method	*MAPE*	*SDAPE*
DEKF-RBFNN [19]	0.97	0.6
GRD-RBFNN [19]	1.07	0.54
SVR-DEKF-RBFNN [19]	0.72	0.44
ARLA-SVR-RBFNN	0.53	0.5
AALA-SVR-RBFNN	0.4	0.4

Case 3: Load prediction of Holidays

The training hourly load data are shown in Table 1 and Figure 7. After 1000 training epochs, the initial parameters of RBFNN are determined by using SVR. The value of L in (1) is found to be 11 for $C = 1$ and $\varepsilon = 0.05$ in SVR. Meanwhile, the average optimal learning rate set of $(p\omega, pc, pw)$ is determined by PSO, which is found to be (1.9640, 0.8327, 4.9085).

Table 2 illustrates the comparison results of *RSME* in (6) between ARLA and AALA. As seen from Table 2, the AALA can produce results superior to ARLA. After training, the proposed approach is tested on 24-h-ahead load forecasting. Figure 8 shows that the predicted values of the proposed AALA-SVR-RBFNN are quite close to the actual values.

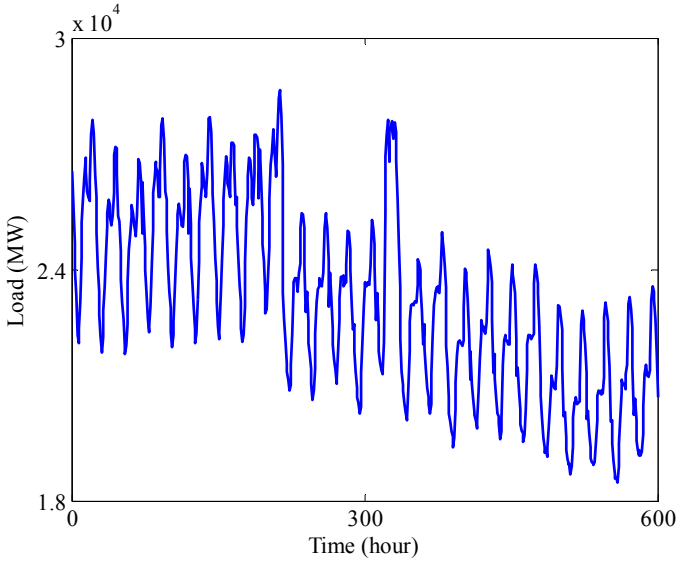

Figure 7. The training hourly actual load data in Case 3.

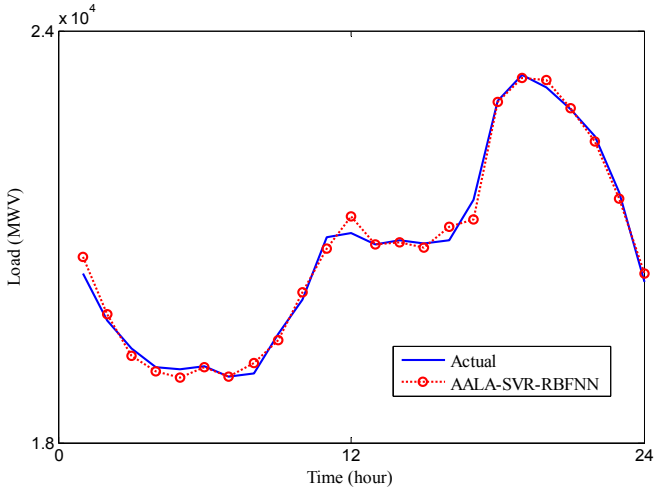

Figure 8. The forecasting results of the proposed AALA-SVR-RBFNN in Case 3.

The comparisons of *MAPE* and *SDAPE* using the five prediction models are shown in Table 5. Comparing DEKF-RBFNN [19], GRD-RBFNN [19], SVR-DEKF-RBFNN [19], and ARLA-SVR-RBFNN with the proposed AALA-SVR-RBFNN, the error of *MAPE* is reduced by 44.94%, 68.79%, 12.50%, and 20.97%, respectively. Moreover, the *SDAPE* value of the proposed algorithm is 0.38%, smaller than those obtained by using the four approaches. These results verify the superiority of the proposed AALA-SVR-RBFNN over other prediction methods.

Table 5. *MAPE* (%) and *SDAPE* (%) results for prediction methods in Case 3.

Method	*MAPE*	*SDAPE*
DEKF-RBFNN [19]	0.89	0.52
GRD-RBFNN [19]	1.57	1.04
SVR-DEKF-RBFNN [19]	0.56	0.52
ARLA-SVR-RBFNN	0.62	0.51
AALA-SVR-RBFNN	0.49	0.38

5. Conclusions

In this paper, a reinforcement neural network of AALA-SVR-RBFNN is developed for predicting STLF accurately. An SVR method is first used to find the initial parameters of RBFNN. After initializations, the parameters of RBFNN are adjusted by using AALA to obtain an optimal combination. When performing the AALA, the optimal nonlinear learning rates are simultaneously determined by using PSO. Meantime, the stagnation of training RBFNN can be overcome during the adaptive annealing learning procedure. Once the optimal RBFNN is established, the 24-h-ahead load forecasting is performed. Three different patterns of load are considered. Simulation results indicate that the proposed AALA-SVR-RBFNN can yield excellent forecasting results over DEKF-RBFNN, GRD-RBFNN, SVR-DEKF-RBFNN, and ARLA-SVR-RBFNN.

Acknowledgments: This work was supported in part by the National Science Council, Taiwan, R.O.C., under grants MOST 105-2221-E-252-002.

Author Contributions: Cheng-Ming Lee and Chia-Nan Ko conceived and designed the experiments; Cheng-Ming Lee performed the experiments; Cheng-Ming Lee and Chia-Nan Ko analyzed the data; and Chia-Nan Ko wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bozic, M.; Stojanovic, M.; Stajic, Z.; Tasic, D. A New Two-Stage Approach to Short Term Electrical Load Forecasting. *Energies* **2013**, *6*, 2130–2148. [CrossRef]
2. Hernandez, L.; Baladrón, C.; Aguiar, J.M.; Carro, B.; Sanchez-Esguevillas, A.J.; Lloret, J. Short-Term Load Forecasting for Microgrids Based on Artificial Neural Networks. *Energies* **2013**, *6*, 1385–1408. [CrossRef]
3. Amjady, N.; Keynia, F. A New Neural Network Approach to Short Term Load Forecasting of Electrical Power Systems. *Energies* **2011**, *4*, 488–503. [CrossRef]
4. Wang, J.; Jiang, H.; Wu, Y.; Dong, Y. Forecasting solar radiation using an optimized hybrid model by Cuckoo Search algorithm. *Energy* **2015**, *81*, 627–644. [CrossRef]
5. Shahidehpour, M.; Yamin, H.; Li, Z. *Market Operations in Electric Power Systems: Forecasting, Scheduling, and Risk Management*; John Wiley & Sons: Hoboken, NJ, USA, 2002.
6. Christiaanse, W.R. Short term load forecasting using general exponential smoothing. *IEEE Trans. Power Appar. Syst.* **1971**, *90*, 900–911. [CrossRef]
7. Dudek, G. Pattern-based local linear regression models for short-term load forecasting. *Electr. Power Syst. Res.* **2016**, *130*, 139–147. [CrossRef]
8. Darbellay, G.A.; Slama, M. Forecasting the short-term demand for electricity. *Int. J. Forecast.* **2000**, *16*, 71–83. [CrossRef]
9. Zheng, T.; Girgis, A.A.; Makram, E.B. A hybrid wavelet-Kalman filter method for load forecasting. *Electr. Power Syst. Res.* **2000**, *54*, 11–17. [CrossRef]
10. Gouthamkumar, N.; Sharma, V.; Naresh, R. Non-dominated sorting disruption-based gravitational search algorithm with mutation scheme for multi-objective short-term hydrothermal scheduling. *Electr. Power Compon. Syst.* **2016**, *44*, 990–1004.
11. Mori, H.; Kobayashi, H. Optimal fuzzy inference for short-term load forecasting. *IEEE Trans. Power Syst.* **1996**, *11*, 390–396. [CrossRef]

12. El-Telbany, M.; El-Karmi, F. Short-term forecasting of Jordanian electricity demand using particle swarm optimization. *Expert Syst. Appl.* **2008**, *78*, 425–433. [CrossRef]
13. Hong, W.C. Electric load forecasting by support vector model. *Appl. Math. Model.* **2009**, *33*, 2444–2454. [CrossRef]
14. Niu, M.F.; Sun, S.L.; Wu, J.; Yu, L.; Wang, J. An innovative integrated model using the singular spectrum analysis and nonlinear multi-layer perceptron network optimized by hybrid intelligent algorithm for short-term load forecasting. *Appl. Math. Model.* **2016**, *40*, 4079–4093. [CrossRef]
15. Bashir, Z.A.; El-Hawary, M.E. Applying Wavelets to Short-Term Load Forecasting Using PSO-Based Neural Networks. *IEEE Trans. Power Syst.* **2009**, *24*, 20–27. [CrossRef]
16. Panapakidis, I.P. Application of hybrid computational intelligence models in short-term bus load forecasting. *Expert Syst. Appl.* **2016**, *54*, 105–120. [CrossRef]
17. Ceperic, E.; Ceperic, V.; Baric, A. A strategy for short-term load forecasting by support vector regression machines. *IEEE Trans. Power Syst.* **2013**, *28*, 4356–4364. [CrossRef]
18. Li, S.; Wang, P.; Goel, L. Short-term load forecasting by wavelet transform and evolutionary extreme learning machine. *Electr. Power Syst. Res.* **2015**, *122*, 96–103. [CrossRef]
19. Ko, C.N.; Lee, C.M. Short-term load forecasting using SVR (support vector regression)-based radial basis function neural network with dual extended Kalman filter. *Energy* **2013**, *49*, 413–422. [CrossRef]
20. Lin, Z.; Zhang, D.; Gao, L.; Kong, Z. Using an adaptive self-tuning approach to forecast power loads. *Neurocomputing* **2008**, *71*, 559–563. [CrossRef]
21. Wang, J.; Li, L.; Niu, D.; Tan, Z. An annual load forecasting model based on support vector regression with differential evolution algorithm. *Appl. Energy* **2012**, *94*, 65–70. [CrossRef]
22. Attoh-Okine, N.O. Analysis of learning rate and momentum term in backpropagation neural network algorithm trained to predict pavement performance. *Adv. Eng. Softw.* **1999**, *30*, 291–302. [CrossRef]
23. Nied, A.; Seleme, S.I., Jr.; Parma, G.G.; Menezes, B.R. On-line neural training algorithm with sliding mode control and adaptive learning rate. *Neurocomputing* **2007**, *70*, 2687–2691. [CrossRef]
24. Li, Y.; Fu, Y.; Li, H.; Zhang, S.-W. The improved training algorithm of back propagation neural network with self-adaptive learning rate. In Proceedings of the International Conference on Computational Intelligence and Natural Computing, Wuhan, China, 6–7 June 2009; pp. 73–76.
25. Suresh, S.; Savitha, R.; Sundararajan, N. Sequential learning algorithm for complex-valued self-regulating resource allocation network-CSRAN. *IEEE Trans. Neural Netw.* **2011**, *22*, 1061–1072. [CrossRef] [PubMed]
26. Atiya, A.F.; Parlos, A.G.; Ingber, L. A reinforcement learning method based on adaptive simulated annealing. In Proceedings of the 2003 IEEE 46th Midwest Symposium on Circuits and Systems, Cairo, Egypt, 27–30 December 2004; Volume 1, pp. 121–124.
27. Kuo, S.S.; Ko, C.N. Adaptive annealing learning algorithm-based robust wavelet neural networks for function approximation with outliers. *Artif. Life Robot.* **2014**, *19*, 186–192. [CrossRef]
28. Ko, C.N. Identification of nonlinear systems using RBF neural networks with time-varying learning algorithm. *IET Signal Process.* **2012**, *6*, 91–98. [CrossRef]
29. Ko, C.N. Reinforcement radial basis function neural networks with an adaptive annealing learning algorithm. *Appl. Math. Model.* **2013**, *221*, 503–513. [CrossRef]
30. Vapnik, V. *The Nature of Statistic Learning Theory*; Springer: New York, NY, USA, 1995.
31. Chuang, C.C.; Jeng, J.T.; Lin, P.T. Annealing robust radial basis function networks for function approximation with outliers. *Neurocomputing* **2004**, *56*, 123–139. [CrossRef]
32. Fu, Y.Y.; Wu, C.J.; Jeng, J.T.; Ko, C.N. Identification of MIMO systems using radial basis function networks with hybrid learning algorithm. *Appl. Math. Comput.* **2009**, *213*, 184–196. [CrossRef]
33. Chuang, C.C.; Su, S.F.; Hsiao, C.C. The annealing robust backpropagation (BP) learning algorithm. *IEEE Trans. Neural Netw.* **2000**, *11*, 1067–1077. [CrossRef] [PubMed]
34. Alrefaei, M.H.; Diabat, A.H. A simulated annealing technique for multi-objective simulation optimization. *Appl. Math. Comput.* **2009**, *215*, 3029–3035. [CrossRef]
35. Ingber, L. Very fast simulated re-annealing. *Math. Comput. Model.* **1989**, *12*, 967–983. [CrossRef]
36. Ingber, L. *Adaptive Simulated Annealing (ASA)*; Lester Ingber Research: McLean, VA, USA, 1993.
37. Shieh, H.L.; Kuo, C.C.; Chiang, C.M. Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification. *Appl. Math. Comput.* **2011**, *218*, 4365–4383. [CrossRef]

38. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
39. Elbeltagi, E.; Hegazy, T.; Grierson, D. Comparison among five evolutionary-based optimization algorithms. *Adv. Eng. Inform.* **2005**, *19*, 43–53. [CrossRef]
40. Ratnaweera, A.; Halgamuge, S.K.; Watson, C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Trans. Evol. Comput.* **2004**, *8*, 240–255. [CrossRef]