South China Load Forecasting based on BFO

Vanli Hoo, Jie Han
(School of Information Technology, Suzhou University, Suzhou, 215000, China)

Abstract: Short-term forecasting of electric power load is crucial to electric investment, which is the guarantee of the healthy development of electric industry. The artificial neural network (ANN) was employed in this paper for forecasting. However, ANN is easy to be trapped in local minima, and its convergence speed is too slow. The traditional solving method is to train the ANN via gradient searching techniques, nevertheless, the gradient searching is ineffective. Therefore, bacterial foraging optimization (BFO) was adopted to train the ANN. Besides, leave-one-out cross-validation is chosen for the sake of avoiding over-training. Experiments on the 10/2010-10/2011 historical load data of South China indicate that the proposed BFO-NN is superior to GA-NN, SA-NN, PSO-NN, and ABC-NN, when median square error is considered as the evaluation indicator.

Keywords: load forecasting; bacterial foraging optimization; prediction; leave-one-out cross-validation

1 Introduction

Short-term load forecasting (STLF) is aimed at predicting electric loads for a period of minutes, hours, days or weeks. It plays an important role in power system planning and operation [1]. Accurate load forecasting will lead to appropriate scheduling and planning with much lower costs on the operation of power systems. It is estimated that an increase of 1% in forecast error caused an increase of a million pounds in operation costs per year for an electricity utility [2].

Balestrassi et al. [3] applied statistical methodology of Design of Experiments (DOE) to better determine the parameters of an Artificial Neural Network (ANN) in a problem of nonlinear time series forecasting. Instead of the most common trial and error technique for the ANN’s training, DOE was found to be a better methodology. The main motivation for their study was to forecast seasonal nonlinear time series—that was related to many real problems such as short-term electricity loads, daily prices and returns, water consumption, etc. A case study adopting this framework was presented for six time series representing the electricity load for industrial consumers of a production company in Brazil.

Niu et al. [4] used particle swarm optimization (PSO) as a training algorithm to obtain the weights of the single forecasting method to form the combined forecasting method. Firstly, several forecasting methods were used to do middle-long power load forecasting. Then the upper forecasting methods were measured by several indices and the entropy method was used to form a comprehensive forecasting method evaluation index, following which the PSO was used to attain a combined forecasting method (PSOCF) with the best objective function value. They then obtained the final result by adding all the results of every single forecasting method. Taking actual load data of a power grid company in North China as a sample, their results showed that PSOCF model improved the forecasting precision compared to the traditional models.

Wu et al. [5] presented a new load forecasting model based on hybrid particle swarm optimization with Gaussian and adaptive mutation (HAGPSO) and wavelet v-support vector machine (Wv-SVM). Firstly, it was proved that mother wavelet function can build a set of complete base through horizontal floating and form the wavelet kernel function. And then, Wv-SVM with wavelet kernel function was proposed in this paper. Secondly, aiming to the disadvantage of standard PSO, HAGPSO was proposed to seek the optimal parameter of Wv-SVM. Finally, the load forecasting model based on HAGPSO and Wv-SVM was proposed in this paper. Their results of application in load forecasts showed the proposed model is effective and feasible.

Venkatesan et al. [6] evaluated the impact of urban growth in the Las Vegas Valley (LVV), Nevada, USA on salinity of the Colorado River. In the past thirty eight years the LVV population had grown from 273,288 (1970) to 1,986,146 (2008). The wastewater effluents and runoff from the valley were diverted back to the Colorado River through the Las Vegas Wash (LVW). With the growth of the valley, the salinity released from urban areas had increased the level of TDS in the wastewater effluents, ultimately increasing the TDS in the Colorado River. The increased usage of water softeners in residential and commercial locations was a major contributor of TDS in the wastewater effluents. Controlling TDS release to the Colorado River is important because of the 1944 Treaty signed between the USA and Mexico. In addition, the agriculture salinity damage cost for the Colorado River had been estimated to be more than $306 a million per year using 2004 salinity levels. With the expected growth of LVW in coming years the TDS release into Lake Mead would increase over time. For this purpose, it was important to investigate future TDS release into the Colorado in anticipation of potential TDS reducing measures to be adopted. In their research, a dynamic simulation model was developed using system dynamics modeling to carry out water and TDS mass balances over the entire LVW. The dynamic model output agreed with historic data with an average error of 2%. Forecasts revealed that conservation efforts can reduce TDS load by 16% in the year 2035 when compared to the current trend. If total population using water softeners can be limited to...
10% in the year 2035, from the current 30% usage, TDS load in the LVW can be reduced by 7%.

Santos et al. [7] considered that methodologies such as artificial neural networks (ANN) had been widely used in the next hour load forecast horizon with satisfactory results. However, those types of approaches had some shortcomings. Usually, the input vector (IV) was defined in an arbitrary way, mainly based on experience, on engineering judgment criteria and on concern about the ANN dimension, always taking into consideration the apparent correlations within the available endogenous and exogenous data. In their paper, a proposal was made of an approach to define the IV composition, with the main focus on reducing the influence of trial-and-error and common sense judgments, which usually were not based on sufficient evidence of comparative advantages over previous alternatives. The proposal included the assessment of the strictly necessary instances of the endogenous variable, both from the point of view of the contiguous values prior to the forecast to be made, and of the past values representing the trend of consumption at homologous time intervals of the past. It also assessed the influence of exogenous variables, again limiting their presence at the IV to the indispensable minimum. A comparison was made with two alternative IV structures previously proposed in the literature, also applied to the distribution sector. Their paper was supported by a real case study at the distribution sector.

In this study, the bacterial foraging optimization (BFO) and neural network (NN) [8-11] were combined together so as to form a novel model named BFO-NN. The structure of this paper is organized as follows. Section 2 is the methodology, giving introductions of the BFO & ANN. In section 3, short-term electric load data of South China City are used as simulation test, and the results show the effectiveness and validity of this proposed BFO-NN approach. Final section 4 is devoted to the conclusion and future work.

2 Methods

2.1 Bacterial Foraging Optimization

BFO is an algorithmic approximation technique mimicking bacteria colony growth. The motion of a single bacterium in two dimensions obeys following assumptions. The path of a bacterium is a sequence of straight-line trajectories; All trajectories have the same constant speed; When a bacterium turns, its choice of the new direction, the angle between two successive trajectories, and the duration of a trajectory are all regulated by a probability distribution; The probability distributions for both the angle and the duration are independent of parameters of the previous trajectory [11-13].

The processing of BFO is explained as follows [14, 15].

Step 1 Initialization. Randomize the positions and velocities v of the bacteria group, and assume their speeds as scalar constant value 1.

Step 2 Calculate the duration of the trajectory τ, the distribution of which satisfies the exponential probability density function (PDF)

\[ P(X = \tau) = \frac{1}{\tau} \exp(-\frac{\tau}{\tau}) \]  

(1)

where the expectation value \( E(X) = \tau \) and the variance \( \text{Var}(X) = \tau^2 \). The time \( \tau \) is given by

\[ \tau = \begin{cases} T_0, & \text{for } f_{pr}/l_{pr} \geq 0 \\ T_0(1+b) \left[ f_{pr}/l_{pr} \right], & \text{for } f_{pr}/l_{pr} < 0 \end{cases} \]  

(2)

Here \( T_0 \) denotes for the minimal mean time; \( f_{pr} \) denotes for the difference between the actual and the previous fitness values; \( l_{pr} \) denotes for the vector pertaining the previous and the actual position in the parameter space; and the final parameter \( b \) is assumed as the dimensionless controlling parameter [16, 17].

Step 3 Calculate the new direction. The PDF of the angle \( \alpha \) between the previous and the new directions is distributed as Gaussian function as follows:

\[ P(X = \alpha, v = \pm \mu) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(\alpha - \mu)^2}{2\sigma^2}\right) \]  

(3)

where the expectation value \( \mu = E(X) \) and variance \( \sigma = \text{Var}(X)\) are given by:

\[ \mu = 62\cdot(1 - \cos \theta); \sigma = 26\cdot(1 - \cos \theta) \text{ for } f_{pr}/l_{pr} < 0 \]  

(4)

\[ \mu = 62\cdot\sigma = 26 \text{ for } f_{pr}/l_{pr} \geq 0 \]

Step 4 Calculate the new position,

\[ x_{new} = x_{old} + n_u \]  

(6)

Here, \( x_{new} \) represents the new position of the bacteria; \( x_{old} \) represents its previous position; \( n_u \) represents the normalized new direction vector; and \( l \) represents the length of the new trajectory.

In summary, the algorithm contains the following parameters to be computed in advance: \( T_0, \tau \) and \( b \).

\[ T_0 = 0.3^{T_0^{1.75}} \]

\[ b = T_0^{-1.5)} \]

\[ \tau = \frac{b}{T_0^{0.31}} \]

(7)

2.2 ANN model

An artificial neural network (ANN) is a mathematical model or computational model that is inspired by the structure and functional aspects of biological neural networks[18]. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation [19-21]. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase [22, 23]. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex
relationships between inputs and outputs or to find patterns in data [24].

A two-layer neural network with 6×3×1 structure was chosen, as shown in Fig. 1. The input layers contain 6 neurons, which correspond to following features extracted from the past actual load.

<table>
<thead>
<tr>
<th>Neuron Index</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Day Type (D)</td>
<td>0 denotes for Tuesday, Wednesday, Thursday, and Friday. 1 denotes for Monday and other workdays after holiday. 2 denotes for Saturday, Sunday, and holidays.</td>
</tr>
<tr>
<td>2</td>
<td>Load (L(t-1))</td>
<td>The last actual load</td>
</tr>
<tr>
<td>3</td>
<td>Load (L(t-2))</td>
<td>The penultimate actual load</td>
</tr>
<tr>
<td>4</td>
<td>Load (L(t-T))</td>
<td>The actual load at the same time last day. If the time interval is 1h, then (T=24). If the time interval is 15m, then (T=24<em>60/15=96). If the time interval is 5m, then (T=24</em>60/5=288).</td>
</tr>
<tr>
<td>5</td>
<td>Min. Temp.</td>
<td>The minimum forecast temperature of current day.</td>
</tr>
<tr>
<td>6</td>
<td>Max. Temp.</td>
<td>The maximum forecast temperature of current day.</td>
</tr>
</tbody>
</table>

The hidden layer contains 3 neurons and the output layer contains only 1 neuron indicating the predicted load. The outputs of the neurons in the hidden layer are calculated as

\[ y_j = f_H(\text{net}_j), \quad j = 1, 2, 3 \]  

Here, \(\text{net}_j\) is the activation value of the \(j\)th node in the hidden layer. The \(f_H\) is called the transfer function (activation function) of the hidden neurons, usually a sigmoid function [25].

\[ f_H(x) = \frac{1}{1 + \exp(-x)} \]  

Meanwhile, the output neuron is calculated as

\[ o = f_O(\text{net}_o) \]  

Here, \(\text{net}_o\) is the activation value of the output neuron. The \(f_O\) is the transfer function of the output neuron, usually a line function. MSE (mean squared error) [26, 27] is selected as the search fitness function, and it is detailed as follow:

\[ \text{MSE} = \frac{1}{N} \sum_{k=1}^{N} [d(k) - o(k)]^2 \]  

Here \(d(k)\) represents the authentic values which are already known to users, \(o(k)\) is the output values of the neural network after BFO training, and \(N\) represents the number of samples. Our goal is to minimize the \(\text{MSE}\) of the network through BFO.

### 2.3 Cross validation

In case of the problem of over-training, we use cross-validation to check for the presence of overtraining and optimally select hyper-parameters such as to minimize the generalization error [28, 29]. It is a technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice [30, 31]. One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on training set, and validating the analysis on the validation set. To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds [32, 33].

In this paper, we chose the leave-one-out cross-validation (LOOCV) [34] which involves using a single observation from the original sample as the validation data, and the remaining observations as the training data. This is repeated such that each observation in the sample is used once as the validation data. This is the same as a K-fold cross-validation with \(K\) being equal to the number of observations in the original sample. Leave-one-out cross-validation is computationally expensive because it requires many repetitions of training [35].

### 3 Experiments and discussions

The experiments were carried out on the platform of P4 IBM with 2GHz main frequency and 1G memory, running under Windows XP operating system. The algorithm was in-house developed via the neural network toolbox of Matlab 2011b. The load and weather data was performed on the 10/2010-10/2011 historical load data of South China. The time interval of the load data is 5 mins. The weather data includes the maximum and minimum
temperatures of every day. We chose Genetic Algorithm, Simulated Annealing [36], Particle Swarm Optimization [37], and Artificial Bee Colony [38] as the comparative algorithms. Their results are listed in in Tab.2 with respect to different day types.

Tab.2 indicates that the median square errors of BFO-NN are the least among all algorithms, which demonstrates that the BFONN is the most accurate for forecasting loads. Besides, in common, the errors of day type 0 are less than the errors of day type 1, and the errors of day type 1 are less than the errors of day type 2.

4 Conclusions

A novel BFO-NN for short term load forecast was presented in this paper. We demonstrate that the BFONN is superior to GA-NN, SA-NN, PSO-NN, ABC-NN, and BFO-NN, when the median square error is considered as the evaluation indicator. In the future, we shall focus on applying the BFO-NN to other fields including image registration [39], compressed sensing [40], and protein folding [41].

References


<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Workdays not after holiday</th>
<th>Workdays after holiday</th>
<th>Holidays</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA-NN</td>
<td>0.72</td>
<td>0.77</td>
<td>0.91</td>
</tr>
<tr>
<td>SA-NN</td>
<td>0.89</td>
<td>0.88</td>
<td>0.99</td>
</tr>
<tr>
<td>PSO-NN</td>
<td>0.74</td>
<td>0.72</td>
<td>0.85</td>
</tr>
<tr>
<td>ABC-NN</td>
<td>0.78</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>BFO-NN</td>
<td>0.63</td>
<td>0.65</td>
<td>0.68</td>
</tr>
</tbody>
</table>


