Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

EA-LSTM: Evolutionary attention-based LSTM for time series prediction

Youru Li [a,b], Zhenfeng Zhu [a,b,*], Deqiang Kong [c], Hua Han [d,e], Yao Zhao [a,b]

[a] Institute of Information Science, Beijing Jiaotong University, Beijing, 100044, China
[b] Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing, 100044, China
[c] Microsoft Multimedia, Beijing, 100080, China
[d] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
[e] CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai, 200031, China

A R T I C L E   I N F O

Article history:
Received 15 November 2018
Received in revised form 28 February 2019
Accepted 18 May 2019
Available online 27 May 2019

Keywords:
Evolutionary computation
Deep neural network
Time series prediction

A B S T R A C T

Time series prediction with deep learning methods, especially Long Short-term Memory Neural Network (LSTM), have scored significant achievements in recent years. Despite the fact that LSTM can help to capture long-term dependencies, its ability to pay different degree of attention on sub-window feature within multiple time-steps is insufficient. To address this issue, an evolutionary attention-based LSTM training with competitive random search is proposed for multivariate time series prediction. By transferring shared parameters, an evolutionary attention learning approach is introduced to LSTM. Thus, like that for biological evolution, the pattern for importance-based attention sampling can be confirmed during temporal relationship mining. To refrain from being trapped into partial optimization like traditional gradient-based methods, an evolutionary computation inspired competitive random search method is proposed, which can well configure the parameters in the attention layer. Experimental results have illustrated that the proposed model can achieve competetive prediction performance compared with other baseline methods.

1. Introduction

A time series is a series of data points that are indexed in chronological order. Effective prediction of time series makes better use of existing information for analysis and decision making. Its wide range of applications include, but are not limited to, clinical medicine [1], financial projections [2], traffic flow prediction [3], human behavior prediction [4], and other areas. Unlike other predictive modeling tasks, time series increases the complexity of sequence dependencies among input variables. Therefore, how to build a suitable predictive model for real time prediction tasks by making full use of complex sequence dependencies is a key issue.

The study of time series prediction begins with a regression equation [5] that predicts the number of sunspots in a year in data analysis. Auto Regressive Moving Average model (ARMA) and Auto Regressive Integrated Moving Average model (ARIMA) [6] show that time series prediction modeling based on regression method is becoming more and more popular. Therefore, these models have also become the most basic and important models in time series prediction. However, due to the high complexity, irregularity, randomness and non-linearity of actual data, it is difficult to achieve high-precision prediction through complex models. By using machine learning methods, one can build non-linear prediction models based on a large amount of historical data. In fact, through repeated training iterations and learning approximations, machine learning models can obtain more accurate predictions than traditional statistical-based models. Typical methods include support vector regression [7] or kernel-based classification and artificial neural multi-order (ANN) [8] with strong nonlinear function approximation and tree-based ensembele learning methods, such as gradient-enhanced regression or decision tree (GBRT, GBDT) [9,10]. However, since the above method lacks efficient processing of sequence dependencies between input variables, the effect is limited in time series prediction tasks [11].

As we know, recurrent neural networks (RNN) [12] are often seen as the most efficient method of time series prediction. Actually, RNN is an artificial neural network in which nodes are connected in a loop, and the internal state of the network can exhibit dynamic timing behavior. However, as the length of the processing time series increases, problems such as gradient disappearance often occur during training of RNNs using conventional activation functions, such as tanh or sigmoid functions, which limit the prediction accuracy of the RNN. The Long

* Corresponding author at: Institute of Information Science, Beijing Jiaotong University, Beijing, 100044, China.
E-mail address: zhfzhu@bjtu.edu.cn (Z. Zhu).

https://doi.org/10.1016/j.knosys.2019.05.028
0950-7051/© 2019 Elsevier B.V. All rights reserved.

and Short-Term Memory Unit (LSTM) [13] is based on a simple RNN that solves the problems of memory and forgetting by adding some multi-threshold gates. Therefore, LSTM and Gated Cycle Unit (GRU) [14] address the limited ability to handle long-term dependencies to some extent. These methods have been successfully applied to many sequence learning problems such as machine translation [15]. In general, LSTM is considered to be one of the state-of-the-art methods for dealing with time series prediction problems. Inspired by cognitive neuroscience, some researchers have introduced attention mechanisms into the coding-decoding framework [16]. Attention mechanisms can better select input sequences and encode semantics in long-term memory to improve the information processing capabilities of neural multi-order. Recently, attention mechanisms have been widely used and perform well in many different types of deep learning tasks, such as image captioning [17], visual question answering [18] and speech recognition [19]. Specifically, most research work [11,20] is usually done by introducing a layer of attention into the encoding–decoding framework.

Time series prediction features are typically obtained by sliding the time window, and the prediction results are affected by the sequence of events. First, we established a multivariate temporal prediction model based on LSTM. Then, inspired by the input information of the human brain's attention mechanism, we introduced a layer of attention in LSTM. The introduced attention mechanism can quantitatively assign importance weights for each specific time step in the sequential features to improve the attentional dispersion defects of the traditional LSTM. Furthermore, our research innovation lies in: Based on the idea of evolutionary computation [21], we propose a competitive random search (CRS) instead of the gradient-based method to solve the attention layer weights. Specifically, when the optimal solution is solved, the random search operator proposed by us can flexibly change the search direction to avoid falling into local optimum [22–24]. Therefore, compared to the traditional gradient-based approach, competitive random search is able to better solve the focus layer weights introduced into the LSTM. Our proposed CRS implements an effective improvement of the selection operator and the crossover operator in the genetic algorithm according to the evolutionary strategy. In particular, the improved crossover operator has integrated more stochastic mechanisms to maintain the differences between the progeny individuals, thereby avoiding premature convergence of the algorithm and being trapped in local optimum. In addition, we use the basic bit mutation operator to specifically perform the mutation operation by randomly inverting one or several gene values at the locus according to the mutation rate on a single encoded string. To verify performance, we performed some experiments on several time series prediction datasets for regression and classification tasks and compared the proposed methods with other baseline methods. The results show that the proposed method can produce higher prediction accuracy than other baseline methods.

2. Preliminaries

In this section, formulation and description of the problem will be displayed. Time series prediction which can be divided into regression or classification problems usually uses a historical sequence of values as the input data. Given sliding-window feature matrix of training series $X = (X_1, X_2, \dots, X_T)$ and $X_t = (x_t^1, x_t^2, \dots, x_t^L)$, where $X_t \in X$. Meanwhile, we define the length of time-step as L. Typically, historical values $y = (y_1, y_2, \dots, y_{T-1})$ are also given. As for classification problems, the historical values y are discrete.

Generally, we learn a nonlinear mapping function by using the history-driven sequence feature X and its corresponding target value y to obtain the predicted value $\tilde{y}_T$ with the following formulation:

$$\tilde{y}_T = f(X, y) \tag{1}$$

where mapping $f(\cdot)$ is the nonlinear mapping function we aim to learn.

3. Methodology

In this section, we will introduce the evolutionary attention based-LSTM and the competitive random search and present how to train this model in detail. In this part, we first give the overview of the model we proposed. Then, we will detail the evolutionary attention-based LSTM. Furthermore, we present the competitive random search and a collaborative training mechanism. A graphical illustration is shown in Fig. 1.

3.1. Overview

The idea of an evolutionary attention-based LSTM is to introduce a layer of attention to the basic LSTM networks. This enables LSTM can not only to handle the long-term dependencies of driving sequences over historical time steps, but also an importance-based sampling. To avoid being trapped, we learn the attention weights by a competitive random search referring to evolutionary computation. To train the model, a collaborative mechanism is proposed. Attention weight that is learnt from the competitive random search is transferred to evolutionary attention-based LSTM for time series prediction. Meanwhile, predicted errors, as the feedback, are sent to direct the searching process.

3.2. Temporal modeling with EA–LSTM

Traditional methods generally model the time series prediction problem with hand-crafted features and make the prediction by well-designed regressors. Recurrent neural network (RNN) is chosen because of its capability to model long-term historical information of temporal sequences. Despite of so many basic LSTM variants for capturing long-time dependencies proposed recently, a large-scale analysis shows that none of them can improve the performance in this issue significantly [13,25]. Therefore, we solve the problem of long-term dependence by replacing the simple RNN unit with the LSTM neuron structure in the recurrent neural network. The LSTM is a special kind of RNN. With its gated structure, including the forget gate, the input gate and the output gate, LSTM can memorize what should be memorized and forget what should be forgot. Especially, the forget gate is the first operator in LSTM to decide what information in last time-step should be dropped with a sigmoid function. It is a key operator in gated structure.

Firstly, we define the attention weights as:

$$W = (W^1, W^2, \dots, W^L) \tag{2}$$

with these attention weights, we can take importance-based sampling for input data with

$$\tilde{X_t} = (x_t^1 W^1, x_t^2 W^2, \dots, x_t^L W^L) \tag{3}$$

Then, $\tilde{X} = (\tilde{X_1}, \tilde{X_2}, \dots, \tilde{X_T})$ is fed into LSTM. Furthermore, we can learn the nonlinear mapping function by these formulations [26] of the calculating process in LSTM cells as follows:

$$i^t = \sigma(W_{xi}\tilde{X_t} + W_{hi}h^{t-1} + W_{ci}c^{t-1} + b_i) \tag{4}$$

$$f^t = \sigma(W_{xf}\tilde{X_t} + W_{hf}h^{t-1} + W_{cf}c^{t-1} + b_f) \tag{5}$$

$$c^t = f^t c^{t-1} + i^t \tanh(W_{xc}\tilde{X_t} + W_{hc}h^{t-1} + b_c) \tag{6}$$

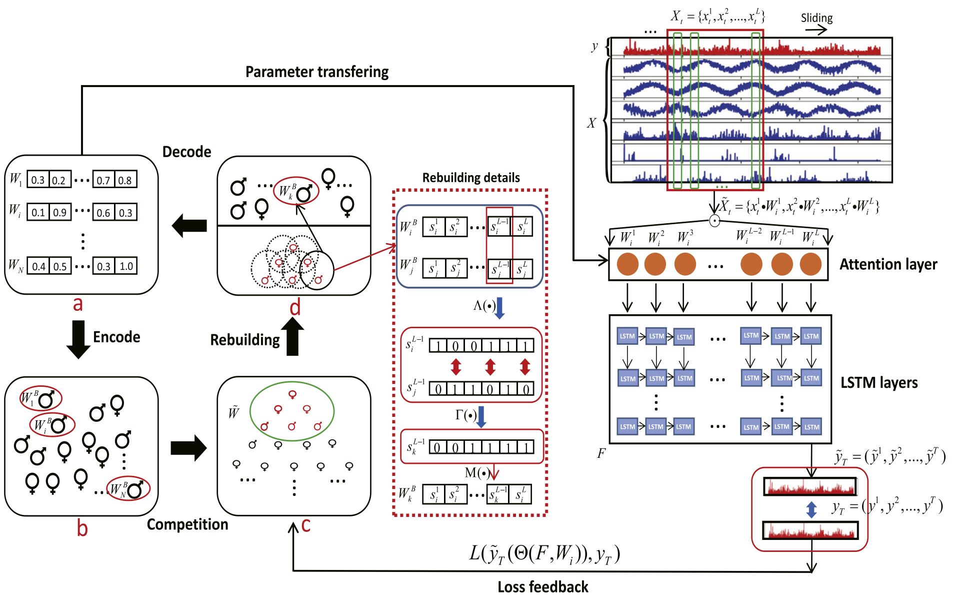

Fig. 1. Graphical illustration of training evolutionary attention-based LSTM with competitive random search. This figure is composed of two parts. The left part displays the process of competitive random search, and the right part the structure of evolutionary attention-based LSTM. On the right, each sample: $X_t = (x_t^1, x_t^2, \ldots, x_t^L)$ in the training set $X = (X_1, X_2, \ldots, X_T)$ multiplies attention weight W_i, the learning result of the left part, producing $\tilde{X}_t = (x_t^1 W_i^1, x_t^2 W_i^2, \ldots, x_t^L W_i^L)$, and $\tilde{X}_t$ are respectively sent to LSTM for training. Finally, the error between the predication result $\tilde{y}_T$ and the real value y_T is obtained in the validation set. The left part consists of a loop where the initial population $W = (W_1, W_2, \ldots, W_N)$ is established in "a", and the individual W_i is encoded into $W^B = (W_1^B, W_2^B, \ldots, W_N^B)$ through binary code and sent to "b". Meanwhile, W_i are respectively transferred to the right network and the corresponding loss evaluation is gained in accordance with the prediction error of the network. Then, the champion individual set $\tilde{W}$ is selected according to the loss situation of W^B in "c", and its subset combination is traversed repeatedly. Finally, the new population is reestablished in the light of operations in the red dotted box and W, the new-generation population, is produced. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$o^t = \sigma(W_{xo}\tilde{X}_t + W_{ho}h^{t-1} + W_{co}c^{t-1} + b_o) \tag{7}$$

$$h^t = o^t \tanh(c^t) \tag{8}$$

where $\sigma(\cdot)$ represents the activation function of sigmoid and W matrices with double subscript the connection weights between the two cells. In addition, i^t represents input gate state, f^t forget gate state, c^t cell state, o^t output gate and h^t the hidden layer output in current time-step. Finally, we can take the last element of output vector h^{t-1} as the predicted value. It can be represented as:

$$\tilde{y}^t = h^{t-1} \tag{9}$$

the final output value can be contacted to a vector:

$$\tilde{y_T} = (\tilde{y}^1, \tilde{y}^2, \ldots, \tilde{y}^T) \tag{10}$$

3.3. Competitive random search

Based on genetic algorithm, competitive random search (CRS) is proposed to generate the optimum parameter combinations in the attention layer of LSTM network. The detailed process of the CRS is elaborated in Fig. 1. The CRS consists of four parts which are introduced as follows.

In Fig. 1, attention weights set, population, $W = (W_1, W_2, \ldots, W_N)$ is given in "a". While being translated into $W^B = (W_1^B, W_2^B, \ldots, W_N^B)$ through binary code and sent into "b", the individual W_i which denotes attention weights are transferred into networks in the right part and produce a corresponding loss value according to predicted error. Then, the champion individual subset $\tilde{W}$ is selected according to the loss of $W^B = (W_1^B, W_2^B, \ldots, W_N^B)$ in "c", and its subset combination is traversed repeatedly. Finally, as is shown in the red dotted box, a new attention wights is rebuilt and W_k^B, the new-generation population, is produced.

Random operators as an improved crossing-over operator based on evolutionary computation are introduced to illustrate how population space is rebuilt in "d". In the red dotted box in Fig. 1, if the selected champion individual combination is W_i^B and W_j^B where each individual is composed of binary strings, they will be evenly divided into L segments in line with L, the time step defined in Section 3.1. Then, the corresponding W_i^B can be expressed by $W_i^B = (S_i^1, S_i^2, \ldots, S_i^L)$ where S_i^1 is a segment of W_i^B. Two important operators are described as follows.

- **Randomly select.** Firstly, we define this operator as $\Lambda(\cdot)$. Its function is to randomly select subsegments in each champion combination. For instance, in Fig. 1, the subsegment $L-1$ of the two subspaces are selected. It should be noted that the number of selected subsection is not fixed.
- **Recombine.** This operator can be expressed as $\Gamma(\cdot)$. It is defined to recombine the genes in the selected subsegment. The process interchanges the two subsegments expressed by binary codes with the length of 6 and from different subspaces in either even or odd index. s_i^{L-1} and s_j^{L-1} will generate s_k^{L-1} after $\Gamma(\cdot)$. It should also be noted that the figure only displays interchange in the even index, but the index where actual interchanges happen is decided by the random judgment of $\Gamma(\cdot)$.

Due to the improved crossing-over operator has integrated more random mechanisms to maintain the difference among child gens so as to avoid a early algorithm convergence and be trapped in local optimum, compared with the single-point or multi-point crossover and other traditional crossover operators in traditional genetic algorithm.

Furthermore, after the abovementioned two steps, we have adopted the simple mutation operator which achieves mutation operation by randomly assigning, on the individual coded string, a digit or several digits of values on the gene locus based on mutation rate. The simple mutation operator is named as $M(\cdot)$ in our paper. Specifically, the operator $M(\cdot)$ is set to reverse the genotype of the newly generated s_k^{L-1} in a random index. For instance, 0 is reversed to 1. Finally, s_k^{L-1} replaces the corresponding s_i^{L-1} in W_i^B, forming W_k^B which is inserted into W. When rebuilding population, we will repeatedly traverse champion individual subset $\tilde{W}$ until the size of W has reached the default value N. The key factor in the CRS is the error feedback introduced from the right network in Fig. 1. The CRS is demonstrated in the optimizing issue as follows:

$$\min L(\tilde{y_T}(\Theta(F,W)), y_T) \tag{11}$$

where F are entire parameters in LSTM and $\Theta(\cdot)$ is the parameter space needed when obtaining the predicted value $\tilde{y_T}$. The most important operator is the rebuilding process which can determines the performance significantly by controlling the direction of the random searching. Algorithm 1 outlines the competitive random search.

Algorithm 1 Competitive Random Search

```
Input:
  $N$: size of attention weights set, $T$: epochs , $\tilde{W}$: champion attention weights subset with the size of $\tilde{N}$, $L = (L_1, L_2, ..., L_N)$: loss of each $W_i \in W$
Output:
  $W$: attention weights set
1:  while $t < T$ do
2:    if $t = 0$ then
3:      $W \leftarrow (W_1, W_2, ..., W_N)$
4:    else
5:      $\tilde{W} \leftarrow Ranking(W_i|L_i, \tilde{N})$
6:      $W \leftarrow \emptyset$
7:      while $length(W) < N$ do
8:        $W \leftarrow \tilde{W}$
9:        for $(W_i, W_j) \in \tilde{W}$ do
10:         $W_k \leftarrow M(\Gamma(\Lambda(W_i, W_j)))$
11:       end for
12:       $W \leftarrow W_k$
13:     end while
14:   end if
15: end while
```

3.4. Parameters transferring

To train our model, we proposed a collaborative mechanism which combines EA-LSTM with the competitive random search. The idea of collaborative training is to share the parameters and loss feedback between the two components of the model. We use mini-batch stochastic gradient decent (SGD) together with Adam optimizer [27] to train EA-LSTM. Except for attention layer, the other parameter in LSTM can be learned by standard back propagation through time algorithm with mean squared error and cross entropy loss as the objective function. Meanwhile, the attention weights outputted by competitive random search will be fed into attention layer before the LSTM begin to be trained. In addition, the current prediction loss of the LSTM in validation set will be used to rank the individual in population.

4. Experiments

In this section, a description of the data set used in our study is first given. Then we will introduce the parameter settings and display the training results of EA-LSTM. Furthermore, we compare our proposed model to some baseline models such as SVR, GBRT, RNN, GRU and LSTM. In addition, some attention-based methods also serve as competitors to verify the performance of our proposed model.

Table 1
Target value statistics and partition settings for two regression datasets.

Dataset	Sensors	Train/Valid	Test	mean	std	min	max
PM2.5	8	28,032/7,008	8,760	94.01	92.25	0	994
SML	16	2,880/720	537	20.15	3.31	11.08	28.55

4.1. Datasets description and setting

For the performance of the proposed model on different types of time series prediction tasks, the data sets used in our experiments are as follows:

- **Beijing PM2.5 Data**[1] (PM2.5). The dataset [28] contains PM2.5 data from the US Embassy in Beijing and an hourly sampling rate from January 1, 2010 to December 31, 2014. At the same time, it also includes meteorological data of Beijing Capital International Airport. Its sensor data such as current time, PM2.5 concentration, dew point, temperature, pressure, wind direction, wind speed, hours, rain hours. The PM2.5 concentration is the target value predicted in the experiment.
- **SML2010**[2] (SML). It is a uci open dataset [29] for indoor temperature prediction. The dataset was collected from a surveillance system installed in the house. It corresponds to approximately 40 days of monitoring data. The data is sampled every minute with a calculation and upload time of 15 min. The sensor data we use includes current time, weather forecast temperature, carbon dioxide, relative humidity, lighting, rain, sun dusk, wind, sun light in west facade, sun light in east facade, sun light in south facade, sun irradiance, Enthalpic motor 1 and 2, Enthalpic motor turbo, outdoor temperature, outdoor relative humidity, and day of the week. The room temperature is the target value to predict in our experiments.
- **MSR Action3D Dataset**[3] (MSR). MSR Action3D dataset contains twenty actions: high arm wave, horizontal arm wave, hammer, hand catch, forward punch, high throw, draw x, draw tick, draw circle, hand clap, two hand wave, side-boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing, pick up and throw. There are 10 subjects, each subject performs each action 2 or 3 times. There are 567 depth map sequences in total. The resolution is 320×240. The data was recorded with a depth sensor similar to the Kinect device.

The settings for the PM2.5 and SML dataset are given in Table 1. In addition, for MSR dataset, we follow the standard settings provided in [4] and calculate the average accuracy for comparison.

4.2. Parameter settings and sensitivity

There are three parameters in the basic LSTM model, i.e., the number of time steps L and the size of hidden units for each layers in LSTM m (we set the same hidden units for each layer in LSTM) and the batchsize b in training process. We carefully tuned the parameters L (time-steps), m (hidden units number) and b (batch-size) for our basic model. To approximate the best performance of the model, we conducted a grid search over $L \in \{3, 6, 12, 18, 24\}$, $m \in \{16, 32, 64, 128, 256\}$, $b \in \{64, 128, 256, 512, 1024\}$ in the **Beijing PM2.5** dataset and $L \in \{6, 12, 18, 24, 36\}$, $m \in$

1 http://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data.
2 http://archive.ics.uci.edu/ml/datasets/SML2010.
3 http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/.

Table 2
Hyperparameters for LSTM in each dataset.

Dataset	Time-Steps	Units	Batchsize
Beijing PM2.5	18	128	256
SML 2010	24	128	128
MSR Action3D	13	128	16

$\{16, 32, 64, 128, 256\}$, $b \in \{64, 128, 256, 512, 1024\}$ in the **SML2010** dataset and $m \in \{16, 32, 64, 128\}$, $b \in \{8, 16, 32, 64\}$ in the **MSR Action3D** dataset. It should be noted that in MSR dataset we set the number of frames in each sample as the L. When one parameter varies, the others are fixed. Finally, we achieved the hyperparameter with the best performance on the validation set to fix the basic model structure. The box plot drawn in Fig. 2 is used to show the sensitivity of the parameters to the two datasets used for the regression task.

The root means squared error for the time series task with one box-whisker (showing middle value, 25% and 75% quantiles, minimum, maximum and outliers) for five testing results of the basic model we proposed. After grid searching, we define the hyperparameters used in EA-LSTM with the best ones. The hyperparameters of LSTM in different datasets are given in Table 2.

Furthermore, there are four hyperparameters in competitive random search, i.e., the size of attention weights set N, the encoding length for each attention weights, the size of champion attention weights subset $\tilde{W}$ and the number of epochs T. To balance the solving efficiency, we defined size of optimization space as 36, encoding length for each subspace as 6 which varies from 0.016 to 1.000, the size of $\tilde{W}$ as 6, and the number of epochs T as 20.

4.3. Evaluation metrics

To evaluate the performance, we take the Root Mean Squared Errors (RMSE) [30] and Mean Absolute Errors (MAE) as the evaluation metrics. They are calculated by the following.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\tilde{y}_t^i - y_t^i)^2} \tag{12}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\tilde{y}_t^i - y_t^i| \tag{13}$$

where $\tilde{y}_t^i$ is prediction, y_t^i is real value and N is the number of testing samples.

4.4. Training attention layer

We trained the EA-LSTM with Competitive random search for 20 epochs. Training processes are visualized in Fig. 3. In addition, the points drawn in Fig. 3 indicate the error and accuracy corresponding to the weights selected in champion individual subset $\tilde{W}$. We can find that training with the CRS, attention weights in optimization space continuously improve the performance and not be trapped. Meanwhile, to better understand importance-based sampling of input series within time steps, the most suitable attention weights are visualized by heat map and showed in Fig. 4. In addition, varied scale of attention distribution of input driving series within multiple time steps over each datasets are showed as well. By solving attention weights which can better suits for the characteristics across different tasks, we improve the performance of the LSTM and get better prediction results. In addition, we can also find that the proposed method effectively utilize local information within one sampling window according to varied scale of attention distribution in Fig. 4. It is crucial to make a soft feature selection in multiple time steps time series prediction.

Table 3
Performance of different baseline methods compared in two datasets for Regression Tasks.

Model	Datasets			
	Beijing PM2.5		SML2010	
	MAE	RMSE	MAE	RMSE
SVR	2.6779	2.8623	0.0558	0.0652
GBRT	0.9909	1.0576	0.0253	0.0327
RNN	0.8646	0.9621	0.0261	0.0367
GRU	0.6733	0.7433	0.0231	0.0288
LSTM	0.6168	0.7026	0.0178	0.0234
Attention-LSTM	0.2324	0.3619	0.0190	0.0225
DA-RNN [11]	–	–	0.0150	0.0197
EA-LSTM	**0.1902**	**0.2755**	**0.0103**	**0.0154**

4.5. Performance comparison

To evaluate the performance of the EA-LSTM training with competitive random search in time series prediction, we compared comparative experiments with other baseline methods, including traditional machine learning methods and deep learning methods. In the experiment, the SVR, GBRT, RNN, LSTM and GRU as competitors were carefully tuned respectively. In addition, all of the baseline methods we compared were trained and tested five times, and the final predictions were shown in Table 3 to reduce random errors on average. We can see that the proposed method effectively improves the performance of the baseline counterparts in the public open benchmarking dataset used in time series prediction.

Furthermore, we also compared the proposed method with DA-RNN [11] in our public testing dataset: **SML 2010**. It should be noted that aiming to make a more fair comparison to DA-RNN, an unopen-sourced model, we can only test it on our public dataset. Specifically, DA-RNN, which is similar to the traditional attention-based model, is a time series predictive model trained by solving the network parameters together with attention-layer parameters. As a matter of fact, this model obtained the state-of-the-art performance by constructing a more complex attention mechanism. With the dataset identically classified into sets for training, validating, and testing, the experimental results show that the EA-LSTM can get a higher predicted precision. We can also see that there is the feasibility to enhance attention-based model by improving training method for attention layer not only by introducing a more complex attention mechanism.

In addition, we compared the proposed method with the same method whose optimization method is replaced with gradient descent which named "Attention-LSTM" to clearly highlight the benefit of using the competitive random search, instead of gradient descent. Specifically, an input-attention layer whose weights are learned together with other parameters is introduced to LSTM. The experimental results clearly highlight the benefit of using the evolutionary computation inspired competitive random search to refrain from being trapped into partial optimization effectively, instead of gradient descent.

Moreover, we also add a comparison between our proposed model and some baseline ones by human action recognition experiments, a typical time series prediction task for testing the ability to take temporal modeling of different methods. Specifically, the MSR Action3D dataset is a benchmark dataset widely used in human action recognition tasks in the field of computer vision. We quote several state of the art methods proposed in

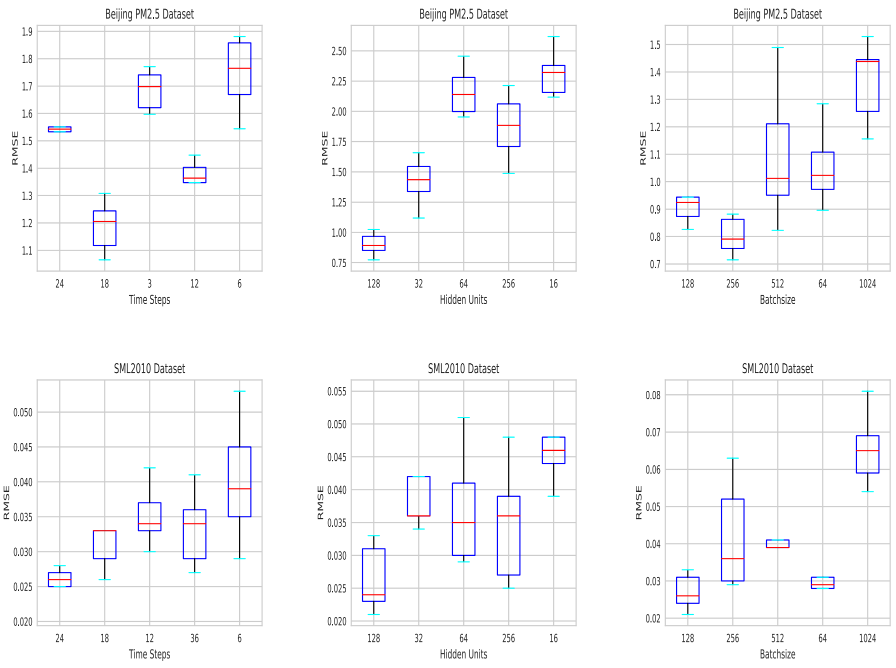

Fig. 2. Parameter Sensitivity of Beijing PM2.5 Dataset and SML2010.

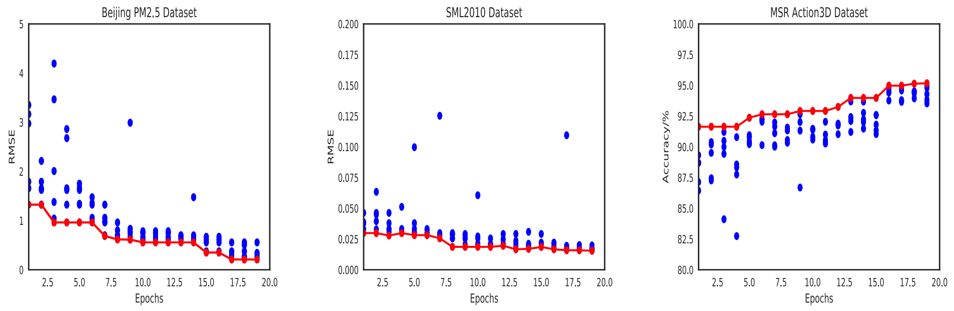

Fig. 3. The training process of Competitive Random Search. The points plotted in the figure represent the error calculation results for each weight in the champion individual subset W for each epoch. There are six champion individuals with blue spot, and the best individuals have red spots in each of the figures. It should be noted that the points in the third sub-picture show the accuracy curve of the EA-LSTM. In addition, we can see that with the iteration of the search, the prediction accuracy gradually increases and shows good convergence. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

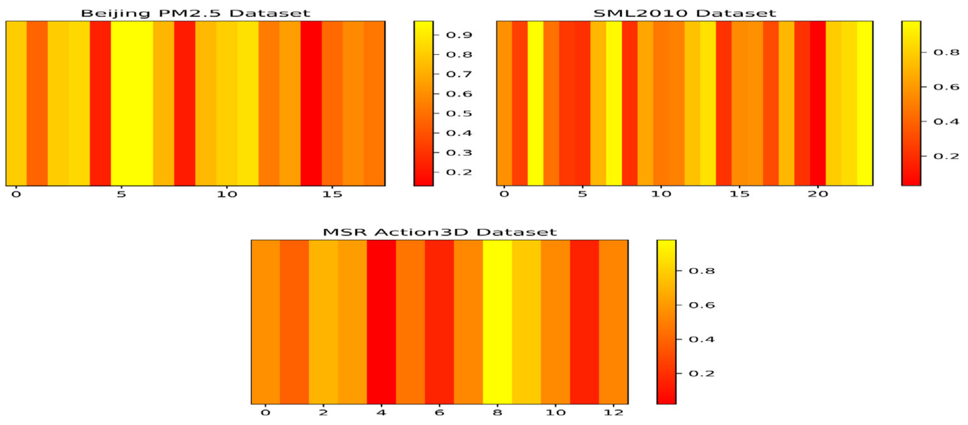

Fig. 4. The attentional distribution map for each experimental dataset, where the coordinates represent the time step of the input driving series and the colors represent the weight of the attention (lighter color, greater weight). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Experimental results on the MSR Action3D dataset.

Methods	Accuracy/%
[31]	91.26
[32]	92.46
HBRNN [4]	94.49
LSTM	90.67
Attention-LSTM	92.58
EA-LSTM	**95.20**

CVPR to verify that our proposed model can maintain a high level of performance in more fields. The experimental results testify that the proposed method also delivers robust performance even in classification prediction tasks (see Table 4).

5. Conclusion

This paper proposes an evolutionary attention-based LSTM model (EA-LSTM), which is trained with competitive random search for time series prediction. During temporal relationship mining, the parameters of attention layer for importance-based sampling in the proposed EA-LSTM can be confirmed. Therefore, the network is able to correctly resolve local feature relationships within the time step. As a hard optimization problem that approximates the best attention weights of the actual input driving series data, our proposed CRS method can avoid being trapped during parameter solving. Experiments have shown that EA-LSTM can provide competitive predictive performance compared to the state-of-the-art methods. These results show that the use of competitive random search to train evolutionary attention-based LSTM not only helps to capture long-term dependence in time series prediction, but also can effectively utilize local information in a sampling window according to different scales of attention distribution. In addition, taking genetic algorithm as an example, this paper introduces evolutionary computation into deep neural networks, and has achieved good performance. For future work, parallel computing will be introduced to increase efficiency, and more studies inspired by biological rules will also be employed to improve the performance of deep neural networks.

Acknowledgments

The work was supported in part by National Key Research and Development of China (NO. 2016YFB0800404), National Natural Science Foundation of China (No. 61572068, No. 61532005), Special Program of Beijing Municipal Science & Technology Commission (No. Z181100000118002), the Strategic Priority Research Program of Chinese Academy of Science (No. XDB32030200), and the Fundamental Research Funds for the Central Universities of China (No. 2018YJS032).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Luchen Liu, Jianhao Shen, Ming Zhang, Zichang Wang, Jian Tang, Learning the joint representation of heterogeneous temporal events for clinical endpoint prediction, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. 109–116.
[2] Wei Cao, Liang Hu, Longbing Cao, Deep modeling complex couplings within financial markets, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 2518–2524.
[3] Pierre Hulot, Daniel Aloise, Sanjay Dominik Jena, Towards station-level demand prediction for effective rebalancing in bike-sharing systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 378–386.
[4] Yong Du, Wei Wang, Liang Wang, Hierarchical recurrent neural network for skeleton based action recognition, in: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1110–1118.
[5] G. Udny Yule, On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers, Phil. Trans. R. Soc. Lond. 226 (226) (1927) 267–298.
[6] G.E.P. Box, Davida Pierce, Distribution of residual autocorrelations in autoregressive-integrated moving average time series models, Publ. Am. Stat. Assoc. 65 (332) (1968) 1509–1526.
[7] Harris Drucker, Christopher J.C. Burges, Linda Kaufman, Alexander J. Smola, Vladimir Vapnik, Support vector regression machines, in: Proceedings of Conference on Neural Information Processing Systems 1996, 1996, pp. 155–161.
[8] Kristina Davoian, Wolfram-Manfred Lippe, Time series prediction with parallel evolutionary artificial neural multi-order, in: Proceedings of the 2007 IEEE International Conference on Data Mining, 2007, pp. 10–15.
[9] Xia Li, Ruibin Bai, Freight vehicle travel time prediction using gradient boosting regression tree, in: Proceedings of the 2016 IEEE International Conference on Machine Learning and Applications, 2016, pp. 1010–1015.
[10] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu, Lightgbm: A highly efficient gradient boosting decision tree, in: Proceedings of Conference on Neural Information Processing Systems 2017, 2017, pp. 3149–3157.
[11] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, Garrison W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: Proceedings of the twenty-sixth International Joint Conference on Artificial Intelligence, 2017, pp. 2627–2633.
[12] David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[13] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780.
[14] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, Yoshua Bengio, On the properties of neural machine translation: Encoder–decoder approaches, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 103–111.
[15] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1724–1734.
[16] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, Neural machine translation by jointly learning to align and translate, 2014, arXiv:1409.0473.
[17] Jiasen Lu, Caiming Xiong, Devi Parikh, Richard Socher, Knowing when to look: Adaptive attention via a visual sentinel for image captioning, in: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3242–3250.
[18] Zhou Yu, Jun Yu, Jianping Fan, Dacheng Tao, Multi-modal factorized bilinear pooling with co-attention learning for visual question answering, in: Proceedings of the 2015 IEEE International Conference on Computer Vision, 2017, pp. 1839–1848.
[19] Suyoun Kim, Takaaki Hori, Shinji Watanabe, Joint ctc-attention based end-to-end speech recognition using multi-task learning, in: Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, 2017, pp. 4835–4839.
[20] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, Yu Zheng, Geoman: Multi-level attention multi-order for geo-sensory time series prediction, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018, pp. 3428–3434.
[21] John H. Holland, Genetic algorithms and the optimal allocation of trials, SIAM J. Comput. 2 (2) (1973) 88–105.
[22] X. Zhang, J. Clune, K.O. Stanley, On the relationship between the OpenAI evolution strategy and stochastic gradient descent, 2017, arXiv:1712.06564.
[23] E. Conti, V. Madhavan, F. Petroski Such, J. Lehman, K.O. Stanley, J. Clune, Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents, 2017, arXiv:1712.06560.
[24] Joel Lehman, Jay Chen, Jeff Clune, Kenneth O. Stanley, Safe mutations for deep and recurrent neural multi-order through output gradients, 2017, arXiv:1712.06563.
[25] Klaus Greff, RupeshKumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber, LSTM: A search space odyssey, IEEE Trans. Neural Netw. Learn. Syst. 28 (10) (2017) 2222–2232.

[26] Alex Graves, Supervised sequence labelling with recurrent neural multi-order, in: Studies in Computational Intelligence, Vol. 385, Springer, 2012.

[27] Diederik P. Kingma, Jimmy Ba Adam, A method for stochastic optimization, 2014, arXiv:1412.6980.

[28] Xuan Liang Huang, Tao Zou, Bin Guo, Shuo Li, Haozhe Zhang, Shuyi Zhang, Hui Huang, Xi Chen Song, Assessing Beijing's PM2.5 pollution: severity, weather impact, APEC winter heating, Proc. R. Soc. A Math. Phys. Eng. Sci. 471 (2182) (2015) 20150257.

[29] F. Zamora-Martínez, P. Romeu, P. Botella-Rocamora, J. Pardo, On-line learning of indoor temperature forecasting models towards energy efficiency, Energy Build. 83 (2014) 162–172.

[30] Mark Plutowski, Garrison W. Cottrell, Halbert White, Experience with selecting exemplars from clean data, Neural Multi-order 9 (2) (1996) 273–294.

[31] Mohammad Abdelaziz Gowayyed, Marwan Torki, Mohamed Elsayed Hussein, Motaz El-Saban, Histogram of oriented displacements (HOD): describing trajectories of human joints for action recognition, in: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, 2013, pp. 1351–1357.

[32] Raviteja Vemulapalli, Felipe Arrate, Rama Chellappa, Human action recognition by representing 3d skeletons as points in a lie group, in: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 588–595.