This article can be cited as J. Saadat, P. Moallem and H. Koofigar, Training Echo Estate Neural Network Using Harmony Search Algorithm, International Journal of Artificial Intelligence, vol. 15, no. 1, pp. 163-179, 2017.

Training Echo Estate Neural Network Using Harmony Search Algorithm

Javad Saadat[1], Payman Moallem[2] and **Hamidreza Koofigar[3]**

[1]Department of Electrcial Engineering, Faculty of Engineering, University of Isfahan, Hazarjerib Street, Isfahan, Iran
Email: saadatjavad90@yahoo.com

[2]Department of Electrcial Engineering, Faculty of Engineering, University of Isfahan, Hazarjerib Street, Isfahan, Iran
Email: p_moallem@eng.ui.ac.ir
(Corresponding author)

[3]Department of Electrcial Engineering, Faculty of Engineering, University of Isfahan, Hazarjerib Street, Isfahan, Iran
Email: koofigar@eng.ui.ac.ir

ABSTRACT

Echo State Networks (ESN) are a special form of recurrent neural networks (RNNs), which allow for the black box modeling of nonlinear dynamical systems. A unique feature of an ESN is that a large number of neurons (the "reservoir"), whose synaptic connections are generated randomly, is used in such that only the connections from the reservoir to the output modified by learning. The computation of optimal weights can then be achieved by a simple linear regression in an offline manner. ESNs have been applied to a variety of tasks from time series prediction to dynamic pattern recognition with great success. In many tasks, however, an online adaptive learning of the output weights is required. Harmony Search (HS) algorithm shows good performance when the search space is large. Here we propose HS algorithm for training echo state network in an online manner. In our simulation experiments, the ESNs are trained for predicting of three different time series including Mackey-Glass, Lorenz chaotic and Rossler chaotic time series with four different algorithms including Recursive Least Squares (RLS-ESN), Particle Swarm Optimization (PSO-ESN), and our proposed methods (HS-ESN and HS-RLS-ESN). Simulation results show that HS-ESN is significantly the fastest algorithm for training ESN whereas can effectively meet the requirements of the output precision. HS-RLS-ESN algorithm firstly uses HS to close to solution region then it uses RLS to obtain less error. HS-RLS-ESN is slower than HS-ESN and faster than RLS-ESN, but its generality power is very close to RLS-ESN.

Key words: echo state network, harmony search algorithm, Recursive Least Squares, chaotic time series.

Mathematics Subject Classification: 92B20, 68T05, 62M10

Computing Classification System: I.2.6

1. INTRODUCTION

In traditional implementations of Recurrent Neural Networks (RNNs), all weights were trained to adjust the output. It means that the standard training procedures for RNNs have a high computational complexity and sometimes, could only find local minimum, therefore, the size of these RNNs was usually limited to 3 to 30 internal units. In the Echo State Neural Network (ESN), which was proposed and developed by Herbert Jaeger (2001), the reservoir is generated randomly and only the readout is trained (Lukosevicius et al. 2007, Jaeger et al. 2007). In contrast to traditional RNNs, ESNs are therefore usually quite large, with hundreds or even thousands of internal units (Lukosevicius et al. 2009).

The ESNs have drawn great interest in theoretical and practical perspectives. In theoretical development of ESN, studying the various ESN schemes (Rodan et al. 2011, Gallicchio et al. 2013, Najibi et al. 2015), improving the reservoir (Schrauwen et al. 2008, Hongyan et al. 2014), ESN with wavelet neurons (Niu et al. 2012, Cui et al. 2013), using different reservoir units (Jaeger et al. 2007), scrutinizing the output units (Holzmann et al. 2010, Chatzis et al. 2011), have been carried out. Moreover, different definition of Echo State Property (ESP) has been explained in (Manjunath et al. 2013, Yildiz et al. 2012), the influence of the memory length on predictive abilities of ESNs has been studied in (Babinec et al. 2011), and the interaction between the driving output feedback and the internal reservoir dynamics in ESNs has been investigated in (Lohmann et al. 2012).

Although the theoretical research of ESN is still at an early stage, it have been successfully applied to various practical tasks, e.g., chaotic time series prediction (Han et al. 2015), ionosphere disturbances behaviour modelling (Massinas et al. 2013), communication channel equalization, noise modelling (Jaeger et al. 2004), dynamical pattern recognition (Ozturk et al. 2007), gene regulatory network modelling (Zhang et al. 2008), speech recognition (Skowronski et al. 2006), reinforcement learning (Bush et al. 2005), language modelling (Tong et al. 2007), prediction of telephone calls load (Bianchi et al. 2015), prediction of blast furnace gas flow (Zhang et al. 2016), and image restoration (Duan and Wang 2016). Furthermore, a practical guide to applying ESNs was presented in (Lukosevicius 2012).

Some applications require online model adaptation, which means online ESN training; in such cases one can use Least Mean Square (LMS) algorithm for training ESN output, although its convergence performance is unfortunately severely impaired by large eigenvalue spreads of cross-correlation matrix of internal network states (Liebald 2004). An alternative to LMS, the Recursive Least Squares (RLS) algorithm, is insensitive to the detrimental effects of eigenvalue spread and boasts a much faster convergence. The downside is that RLS is computationally more expensive and notorious for numerical stability issues (Jaeger 2007).

On the other hand, training of an ESN can be considered as a general optimization problem. However, various optimization approaches have been recently used in many practical and real applications, while evolutionary methods have attracted more attentions (Martin et al.

2009, Precup et al. 2012, Moallem and Razmjooy 2012, Ghosn et al. 2016, Solos et al. 2016). Different evolutionary-based optimization algorithms like Genetic Algorithm (GA) (Dongming et al. 2005), evolution and learning (Chatzidimitriou and Mitkas, 2013), Artificial Fish Swarm Algorithm (ASFA) (Wang and Ping Guo, 2014), and Particle Swarm Optimization algortihm (PSO) (Song et al. 2009, Heshan et al. 2015) are further options for training ESN output connections, such algorithms don't require initial values and uses a random search instead of a gradient-based search, so derivative information is unnecessary.

Recently, Harmony Search (HS) optimization algorithm which is inspired by music phenomenon was proposed. Geem et al. (2001) developed a new HS meta-heuristic algorithm that was conceptualized using the musical process of searching for a perfect state of harmony. Although the HS algorithm is a comparatively simple method, it has been successfully applied to various optimization problems. It has been showed that for problems with large dimensions, HS optimization is more effective and faster than some other evalutionary methods (Wang et al. 2015). In this paper, we fistly use HS algorithm for fast ESN online training. Then, we combine classical RLS with HS algorithm for ESN online training to establish a tradeoff between training speed and output accuracy.

The remainder of this paper is organized as follows. In Section 2, we present a brief review on the ESN model, the echo state property and offline training algorithm. This is followed by review on online training algorithms in Section 3. In Section 4, we present a brief review on the harmony search algorithm which we used for ESN training. Our proposed online training algorithms for ESN, including HS-ESN and HS-RLS-ESN are presented in section 5. Simulation results of different online learning of ESN on different time series prediction and discussions about obtained results are given in Sections 6 and 7, which demonstrates the performance of our proposed HS-ESN and HS-RLS-ESN algorithms in compassion with other online learning algorithms. Finally, we briefly conclude our work in Section 8.

2. ECHO STATE NETWORK

ESN as a special form of recurrent discrete-time neural network, which is shown schematically in Figure 1, is fully characterized by its weight matrices and activation functions. In general, the classical ESN consists of K input units $\mathbf{u}(n)=(u_1(n),\ldots,u_k(n))^T$, N internal units $\mathbf{x}(n)=(x_1(n),\ldots,x_N(n))^T$, L output units $\mathbf{y}(n)=(y_1(n),\ldots,y_L(n))^T$, an $N\times K$ input weight matrix $\mathbf{W^{in}}$, an $N\times N$ internal weight matrix $\mathbf{W}$ (reservoir), an $L\times(K+N+L)$ output weight matrix $\mathbf{W^{out}}$, possibly an $N\times L$ back projection weight matrix $\mathbf{W^{back}}$, an activation function f (usually *tanh* or another sigmoid function) and an output function f^{out} (usually the identity).

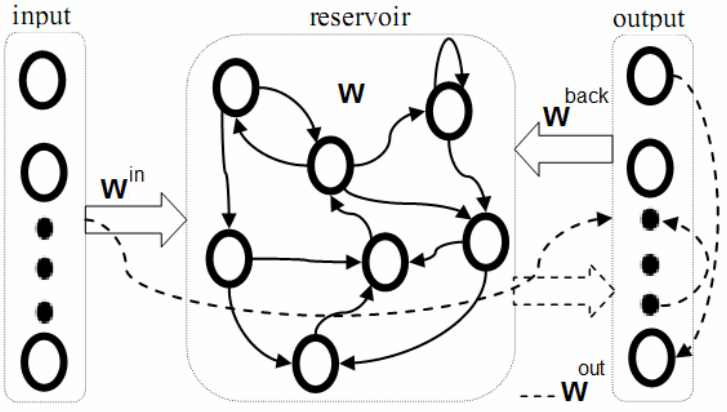

Figure 1. The basic ESN architecture where solid arrows indicate fixed and random connections, and dotted arrows indicate trainable connections.

Input signals are fed into the input units and propagate into the internal units. The activation of internal units is updated according to:

$$\mathbf{x}(n+1) = f(\mathbf{W}^{\mathbf{in}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{\mathbf{back}}\mathbf{y}(n)) \quad (1)$$

The output is computed according to :

$$\mathbf{y}(n+1) = f^{out}\left(\mathbf{W}^{\mathbf{out}}\left[\mathbf{u}(n+1) \quad \mathbf{x}(n+1) \quad \mathbf{y}(n)\right]\right) \quad (2)$$

$[\mathbf{u}(n+1) \ \ \mathbf{x}(n+1) \ \ \mathbf{y}(n)]$ is the concatenation of the input, internal, and previous output activation vectors.

In order to correctly working the ESN approach, the reservoir should satisfy the so-called Echo State Property (ESP): the state of the reservoir $\mathbf{x}(n)$ should be uniquely defined by the fading history of the input $\mathbf{u}(n)$. It means that for a long enough input $\mathbf{u}(n)$, the reservoir state $\mathbf{x}(n)$ should not depend on the initial conditions of the input. For most practical purposes, the ESP can be easily satisfied by merely ensuring that the reservoir weight matrix $\mathbf{W}$ is contractive, i.e., by scaling the reservoir weight matrix so that it's spectral radius $\rho(\mathbf{W})$, which is defined as its largest absolute eigenvalue, is less than one (Venayagamoorthy et al. 2009).

As shown in Figure 1, the only trainable connections are $\mathbf{W}^{\mathbf{out}}$, which is leading from the internal units ($\mathbf{x}(n)$) to the output units, while all other connections remain fixed. This gives us the possibility to employ each of arbitrary fast linear regression algorithms for training. The training error to be minimized can be expressed by Mean Square Error (MSE) as:

$$MSE = \frac{1}{T}\sum_{n=1}^{T}\|\mathbf{d}(n) - \mathbf{y}(n)\|^2 \quad (3)$$

where T is the number of training data, $\mathbf{d}(n)$ and $\mathbf{y}(n)$ are the desired and output vectors, respectively.

In summary, to use ESN in an application, it is necessry to bulid, train and then finally exploit the ESN, which are explained here:

2.1. Building an untrained ESN

At the first step, it is necessary to build an untrained ESN ($\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{back}$) which has the echo state property:

a. Generate a random weight matrix $\mathbf{W}_0$ that obviously the crucial parameter is N, which is the number of units in the reservoir. Although the reservoir size N is task-dependent. As a rule of thumb, reservoir sizes are usually selected between 1/10 to 1/2 of number of samples in the training data set (Song et al. 2010).
b. Normalize matrix $\mathbf{W}_0$ to matrix $\mathbf{W}_1$ with spectral radius λ_{max} of $\mathbf{W}_0$ as $\mathbf{W}_1=\mathbf{W}_0/\lambda_{max}$, $\mathbf{W}_1$ has now unit spectral radius.
c. Scale matrix $\mathbf{W}_1$ to matrix $\mathbf{W}=\alpha\mathbf{W}_0$ where $\alpha\leq 1$, it means that $\mathbf{W}$ has now a spectral radius of α. One of the most central global parameters of an ESN is spectral radius of the reservoir connection matrix $\mathbf{W}$, the spectral radius determines how fast the influence of an input dies out in a reservoir with time, and how stable the reservoir activations is.
d. Generate random weight matrices $\mathbf{W}^{in}$ and $\mathbf{W}^{back}$. Based on practical experiences, sparse connections tend to give a slightly better performance (Jaeger 2013). So, connect each reservoir unit to a small fixed number of other units (e.g. 10 on average), irrespective of the reservoir size. Exploit this reservoir sparsely results in improving computation speed.

2.2. Training and exploting the ESN

As previously mentioned, to train the ESN dynamics, only the the output wigths $\mathbf{W}^{out}$ should be trained, which can be done in the following offline manner:

a. Initialize arbitrarily the state of the units.
b. Run the ESN by driving it with the training input signal and by applying update equation (1).
c. Collect remaining input and network states row-wise into a matrix $\mathbf{M}$. Usually some initial portion of the states thus collected are discarded to accommodate for a washout of the arbitrary (random or zero) initial reservoir state needed at time the first iteration.
d. Collect simultaneously the remaining training pre-signals into a column vector $\mathbf{T}$.
e. Compute the output weights $\mathbf{W}^{out}$ by multiplying the pseudo-inverse of $\mathbf{M}$ with $\mathbf{T}$, as:

$$\mathbf{W}^{out} = (\mathbf{M}^{-1}\mathbf{T})^{T} \quad (4)$$

In fact, the i^{th} column of $\mathbf{W}^{out}$ contains output weights from all network units to the i^{th} output unit.

After training, the network ($\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{back}$, $\mathbf{W}^{out}$) is now ready for exploitng, it can be driven by an untrained input sequences $\mathbf{u}(n)$, using the update equations (1) and (2), to produce the

output. More details of the offline training algorithm for the ESN can be found in (Jaeger 2013).

3. ONLINE TRAINING ALGORITHMS

The offline algorithm have been applied to a variety of tasks with great success, however, in many tasks that reservoir states have feedback connection from output units or parameters change during the actual task, an online adaptive learning of the output weights is required. For example, when the characteristics of the noise change over time, the ESN needs to modify its internal parameters adaptively.

The most well-known online learning algorithm, the Least Mean Square (LMS) algorithm, however, is difficult to use with ESNs as the cross-correlation matrix of internal states shows a large eigenvalue spread. This leads to a very slow convergence behaviour of the weight vector $\mathbf{W}^{out}$, rendering the LMS algorithm useless with current ESN implementations. So far, ESN tasks that required online learning of weights employed the Recursive Least Square filter algorithm (Song et al. 2011). However, RLS algorithm has a number of disadvantages: It has higher computational cost (quadratic in filter length) and space complexity, it is more difficult to implement and can fall prey to instability (Jaeger 2002).

For ESN, Song and Feng (2009) show that the output connection weight ($\mathbf{W}^{out}$) adaptation problem can be considered as a kind of constrained optimization problem, which results in a sufficient condition for the asymptotic stability. Therefore, they proposed using Particle Swarm Optimization Echo State Network (PSO-ESN) to solve the optimization problem and computing the output connection weight $\mathbf{W}^{out}$.

4. HARMONY SEARCH ALGORITHM

The harmony search algorithm is an optimization technique inspired by music phenomenon. Just as musical instruments are played with certain discrete musical notes based on musician' experiences, or randomness in an improvisation process, so design variables can be assigned with certain discrete values based on computational intelligence or randomness in the optimization process (Lee et al. 2005)(Wang et al. 2015). Just as musicians improve their experiences based on an aesthetic standard, design variables in computer memory can be improved based on objective function.

HS algorithm includes a number of optimization operators, such as the Harmony Memory (HM), the Harmony Memory Size (HMS, number of solution vectors in harmony memory), the Harmony Memory Considering Rate (HMCR), and the Pitch Adjusting Rate (PAR). In the HS algorithm, the harmony memory stores the feasible vectors, which are all in the feasible space. The harmony memory size determines how many vectors it stores. Then a new vector

is generated (next we explain how) and if the new harmony is better than existing worst harmony in the HM, the new harmony is included in the HM and the worst harmony is excluded from the HM. This procedure is repeated until fantastic harmony is found. The optimization procedure of the HS algorithm is shown in Figure 2 and for better understanding, briefly described in the following subsections, the whole steps of the harmony search algorithm could be found in (Geem 2009, Lee et al. 2004) (Wang et al. 2015).

1. Initialize the problem and algorithm parameters: First, the optimization problem is specified, a suitable objective function (f(x)), a set of decision variable (x_i) and the set of possible range of values for each decision variable (X_i). Then the HS algorithm parameters that are required to solve the optimization problem are also specified in this step: HMS (number of solution vectors), $HMCR$, $PAR_{\max}$, $PAR_{\min}$, $bw_{\max}$, $bw_{\min}$ and NI (the maximum number of searches).

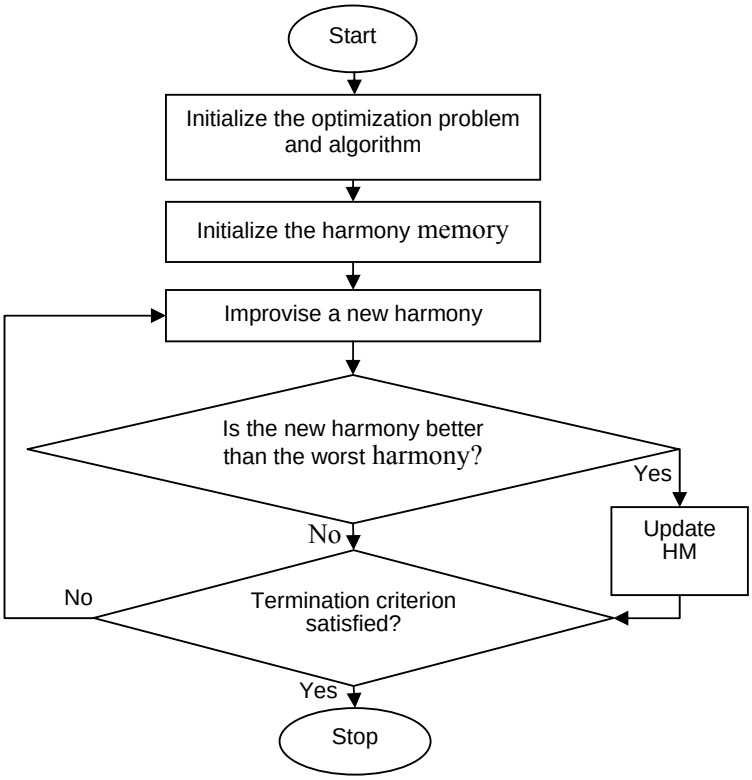

Figure 2. HS algorithm optimization procedure.

2. Initialize the harmony memory: In this step, the harmony memory matrix is filled with many randomly generated solution vectors as the size of the HMS, and then sorted by the values of the objective function f(x).

3. Improvise a new harmony: A new harmony vector $x^t = (x_1^t, ..., x_N^t)$ is generated based on three rules as following:

$$x_i^{'j} \in \begin{cases} \{x_i^1 \quad x_i^2 \quad \ldots \quad x_i^{HMS}\} & \text{with probability } HMCR \\ D & \text{with probability } 1 - HMCR \end{cases} \tag{5}$$

where D is random value between -1 to +1. Every component chosen by harmony consideration (with probablity $HMCR$) is examined for pitch adjustment based on the following rule, pitch adjusting decision given as:

$$x_i^t \leftarrow \begin{cases} \text{yes} & \text{with probability } PAR(gn) \\ \text{no} & \text{with probability } 1 - PAR(gn) \end{cases} \tag{6}$$

The value of (1-$PAR(gn)$) sets the rate of doing nothing. If pitch adjustment decision for x_i^t is yes, x_i^t is modified as:

$$x_i^t = x_i^t + bw(gn) \times rand() \tag{7}$$

where $rand()$ is a random number between 0 and 1 and bw is an arbitrary distance bandwidth which is dynamically changed as a function of generation number gn, as follows:

$$bw(gn) = bw_{\max} \exp(c.gn)$$
$$c = \frac{\ln(\frac{bw_{\min}}{bw_{\max}})}{NI} \tag{8}$$

PAR is also related to the generation number gn, as:

$$PAR(gn) = PAR_{\max} - \frac{PAR_{\max} - PAR_{\min}}{NI} gn \tag{9}$$

4. Update the HM: The new memory is judged in terms of the objective function value (fitness function f(x) and if the new memory is better than the previous memory in the HM, then new harmony memory is included in the HM and, the existing worst harmony is excluded from the HM.

5. Check for stopping criteria: If maximum number of improvisations is reached, then stop, otherwise steps 3 and 4 are repeated.

The common factor in meta-heuristic algorithms is that they combine rules and randomness to imitate natural phenomena. Compared to gradient based mathematical optimization algorithms, the HS algorithm imposes fewer mathematical requirements to solve optimization problems and does not require initial starting values for the decision variables. The HS algorithm uses a stochastic random search based on the $HMCR$ and PAR, which effactively guide a global search rather than a gradient search, so that derivative information is unnecessary. Furthermore, the HS algorithm generates a new vector after considering all of the existing vectors based on the $HMCR$ and the PAR, rather than considering only two

(parents) as in genetic algorithms. These features increase the flexibility of the HS algorithm and produce better solutions.

Moreover, for optimization problems with large dimensions, HS algorithm is more effective and faster than some other evalutionary methods (Wang et al. 2015). As previously mentioned, ESN reservoir is usually large in practical applications, which results in large dimension of Wout. Therefore it is expected that HS algorithm works well for ESN online training.

5. PROPOSED TRAINING ALGORITHMS: HS-ESN AND HS-RLS-ESN

In this paper, we propose two training algorithms for ESN neural networks. The first one is HS-ESN that is similar to PSO-ESN (Mahdavi et al. 2007), but it replaces HS instead of PSO which results in more accurate and faster learning. Our simulation results show that HS-ESN is very faster that RLS-ESN, but its accuracy is lower. The following steps explain our propsoed HS-ESN in details:

1- Build an untrained ESN ($\mathbf{W}^{in}$,$\mathbf{W}$,$\mathbf{W}^{back}$) (subsection 2.1)
2- Consider the training dataset for the ESN ($\mathbf{u}(n)$: training inputs, $\mathbf{d}(n)$:desired outputs)
3- Compute the ESN internal state ($\mathbf{x}(n)$) after applying the training inputs ($\mathbf{u}(n)$) using Equation (1)
4- Initialize the parameters of HS optimization algorithm (which are described in the step 1 of “Harmony Search Algorithm”, section 4). The HS objective function is MSE of ESN training dataset ($\mathbf{u}(n)$,$\mathbf{d}(n)$), which can be computed by Equations (1) through (3).
5- Generate HMS random ESN output matrix ($\mathbf{W}^{out}$), sort them by the values of the objective function MSE, and put them in HM (which are described in the step 2 of “Harmony Search Algorithm”, section 4).
6- Run HS optimization algorithm to compute the best ESN output matrix ($\mathbf{W}^{out}$) (which are described in the steps 3 trough 5 of “Harmony Search Algorithm”, section 4). The stopping criteria for HS algorithm are maximum number of improvisations or achieving to a desired MSE.

In many applications, the accuracy of HS-ESN is sufficient but there may some applications which need more accuracy. The accuracy of RLS-ESN is the best but it is execution time is very high which is not suitable for many online training. Therefore, we also propose another training algorithm, HS-RLS-ESN, which firstly uses HS for a pre-training based on a part of training dataset, and then it complementary trains by RLS based on the remained training dataset, in order to improve its accuracy. The following steps explain our proposed HS-RLS-ESN method, briefly:

1. Pre-train ESN with 80% of training dateset using HS-ESN algorithm.
2. Retrain the pre-trained ESN with the remained 20% of training dataset using RLS algorithm.

Our simulation results show that HS-RLS-ESN establishes a trade off between HS-ESN and RSL-ESN, from points of accuracy and execution time.

6. SIMULATON RESULTS

In this section, three illustrative examples are given to demonstrate the performance of our proposed online ESN HS algorithm that trained ESN (HS-ESN) for prediction of the Mackey-Glass, Lorenz, and Rossler chaotic time series. The prediction performance is measured by the mean squared error on the both train (MSE_{train}) and test (MSE_{test}) sequence as well as execution time during network training. In all experiments, 90% of generated data is considered as train data and, the remained 10% is considered as test data. To investigate the results statistically, the experiments are carried out over 10 runs from different random initial points, and the averages and standard deviations are reported. All the experiments were carried out in MATLAB 2010a environment, by the personal computer with CPU speed of 2.67GHz and RAM size of 4.00GB.

The size of reservoir ($\mathbf{W}$) is chosen as 500×500 for all simulations and, the spectral radius of $\mathbf{W}$ is chosen as 0.95, for the first and third experiments and 0.90 for the second experiment, which will guarantee the reservoir to work in stable regions. The sparsity of $\mathbf{W}$ is also set to 10%.

In ESN-based time series prediction, the direct connection from input to output and $\mathbf{W}^{back}$ are not necessary. Since the ESN with one output is enough, it means that the size of $\mathbf{W}^{out}$ is 1×500. Moreover, for ESN with R input unit, the size of $\mathbf{W}^{in}$ is 500×R, in which R is dependent to necessary delay time and the embedded dimension of chaotic time series. The sparsity of $\mathbf{W}^{in}$ is also set to 5% and the weights of $\mathbf{W}^{in}$ are generated by a uniform random generator in interval of [-1,+1].

The length of training sequence pairs $\{\mathbf{u}(n),\mathbf{d}(n)\}$ is 3500 in which the first 500 samples will be discarded to wash out the initial transient. The zero-mean Gaussian noise is also added to the original time series, and the noise level (ratio between the standard deviation of noise and the signal standard deviation) is 20%. The length of the validation set is 500. It is noted that the validation set is still noisy (noise level is 20%), because the noiseless test data set is not available before the predictor is created.

For all examples presented in this paper, the HS algorithm parameters set to HMS=10, $HMCR$=0.8, PAR_{max}=0.5, PAR_{min}=0.1, bw_{max}=0.4, bw_{min}=0.1, $D\in${-0.4,-0.2,0,0.2,0.4} and NI=500. The weights of all initial $\mathbf{W}^{out}$ for primary HM are generated by a uniform random generator in interval of [-1,+1].

To further demonstrate the performance of the proposed algorithms, HS-ESN and HS-RLS-ESN, two other algorithms, RLS-ESN and PSO-ESN, are used to compare the prediction results. For HS-RLS-ESN training algorithm, in all simulations, 80% of the training datasets

are considered for pre-training by HS and the remained 20% are considered for complementary training by RLS.

6.1. Mackey-Glass time series

The Mackey-Glass time series is derived from a time-delay differential system with the form (Mackey et al. 1997)

$$\frac{dx}{dt} = \beta.x(t) + \frac{\alpha.x(t-\delta)}{1+x(t+\delta)^{10}} \tag{10}$$

where x(t) is the value of time series at time t. The system is chaotic for $\delta \leq 16.8$, and the parameter values are chosen as β=-0.1, α=0.2 and δ=17. The data set is constructed using second-order Runge-Kutta method with a step size of 0.1. The embedded data vector consists of four values of the time series $u_{MG}(n)$={x(n) x(n-6) x(n-2×6) x(n-3×6)}, where the delay time and the embedded dimension for phase space reconstruction are six and four (Kennel et al. 1992). Therefore, for Mackey-Glass chaotic time series, $\mathbf{W}^{in}$ dimension of ESN is 500×4. Table 1 summarizes the averages and standard deviations of MSEs (training and testing) and execution times, over 10 runs, by applying the four different training algorithms for prediction of the Mackey-Glass chaotic time series. In this table, the standard deviations are in parenthesis.

Table 1: Performances of different ESN training algorithms for prediction of Mackey-Glass chaotic time series. The averages and standard deviations of MSE (training and testing) and execution times are reported separately, over 10 runs. The standard deviations are also reported in parenthesis.

Training algorithm	MSE_{train}	MSE_{test}	Execution time (sec)
PSO-ESN	1.77×10^{-6} (3.94×10^{-6})	1.31×10^{-4} (2.68×10^{-4})	327.0 (7.2)
HS-ESN	4.16×10^{-7} (3.23×10^{-7})	7.91×10^{-6} (1.04×10^{-5})	232.3 (6.9)
RLS-ESN	8.11×10^{-9} (6.54×10^{-9})	1.76×10^{-7} (1.34×10^{-7})	1549.5 (15.5)
HS-RLS-ESN	5.49×10^{-8} (6.61×10^{-8})	3.79×10^{-7} (2.40×10^{-7})	933.4 (7.5)

6.2. Lorenz Chaotic Time Series

In this example, the data is derived from the Lorenz system, which is given by three time-delay differential systems:

$$\begin{aligned}\frac{dx}{dt} &= -a.x(t) + a.y(t)\\ \frac{dy}{dt} &= b.x(t) - y(t) - x(t).z(t)\\ \frac{dz}{dt} &= x(t).y(t) - c.z(t)\end{aligned} \tag{11}$$

where x(t), y(t) and z(t) are the values of time series at time t. A typical choice for the parameter values are as: a=10, b=28 and c=8/2, In this case, the system is chaotic. The data set is constructed by using four-order Runge-Kutta method with the initial value as: x(0)=12, y(0)=2 and z(0)=9 and the step size is chosen as 0.02.

In order to extract the dynamic characteristic of Lorenz system to predict x(n+1), the embedded data vector is chosen as: $u_L(n)$={x(n) x(n-8) x(n-2×8) … x(n-5×8)}, where the delay time and embedded dimension for the phase space reconstruction are eight and six, respectively. Therefore, for Lorenz chaotic time series, $\mathbf{W}^{in}$ dimension of ESN is 500×6. Table 2 summarizes the averages and standard deviations over 10 runs, by applying the four different training algorithms for prediction of the Lorenz chaotic time series by ESN. In this table, the standard deviations are in parenthesis.

Table 2: Performances of different ESN training algorithms for prediction of Lorenz chaotic time series. The averages and standard deviations of MSE (training and testing) and execution times are reported separately, over 10 runs. The standard deviations are also reported in parenthesis.

Training algorithm	MSE_{train}	MSE_{test}	Execution time (sec)
PSO-ESN	1.36×10^{-6} (2.34×10^{-6})	2.13×10^{-4} (2.93×10^{-4})	334.2 (6.0)
HS-ESN	6.74×10^{-7} (7.61×10^{-7})	3.03×10^{-5} (1.56×10^{-5})	235.3 (4.7)
RLS-ESN	4.12×10^{-9} (2.84×10^{-9})	1.52×10^{-7} (2.38×10^{-7})	1530.5 (6.5)
HS-RLS-ESN	4.85×10^{-8} (2.27×10^{-8})	6.34×10^{-7} (5.24×10^{-7})	926.8 (3.7)

6.3. Rossler Chaotic Time Series

The sequence of the Rossler time series (Precup et al. 2014) is generated from the differential systems, as:

$$\frac{dx}{dt} = -z(t) - y(t)$$
$$\frac{dy}{dt} = x(t) + d.y(t) \quad (12)$$
$$\frac{dz}{dt} = e + z(t).(x(t) - f)$$

For the series, a typical choice for the parameter values are as d=0.15, e=0.2 and f=10. In this case, the system is chaotic and the step size in the four-order Runge-Kutta method is 0.01. To predict the desired output x(n+1), the delay time and embedded dimension are chosen as five and four. It means that the embedded data vector is chosen as u_R(n)={x(n) x(n-5) x(n-2×5) x(n-3×5) x(n-4×5)}. Therefore, for Rossler chaotic time series, $\mathbf{W}^{in}$ dimension of ESN is 500×5. Table 3 show the simulation results of prediction Rossler chaotic time series with ESN that trained by four different algorithms.

Table 3: Performances of different ESN training algorithms for prediction of Rossler chaotic time series. The averages and standard deviations of MSE (training and testing) and execution times are reported separately, over 10 runs. The standard deviations are in parenthesis.

Training algorithm	MSE_{train}	MSE_{test}	Execution time (sec)
PSO-ESN	1.37×10^{-6} (2.73×10^{-6})	2.61×10^{-4} (3.42×10^{-4})	317.3 (4.1)
HS-ESN	2.94×10^{-7} (3.12×10^{-7})	4.02×10^{-5} (2.16×10^{-5})	226.1 (4.6)
RLS-ESN	4.01×10^{-9} (7.53×10^{-9})	1.43×10^{-7} (2.54×10^{-7})	1514.9 (6.1)
HS-RLS-ESN	4.98×10^{-8} (9.01×10^{-8})	7.81×10^{-7} (6.49×10^{-7})	918.5 (3.6)

7. DISCUSSION

Experimental results in Tables 1, 2 and 3 show that HS-ESN algorithm is the fastest training algorithms, about 30% and 85% faster than PSO-ESN and RLS-ESN, respectively. The accuracy of HS-ESN demonstrates improvement about one order, regarding to PSO-ESN, both in training and testing. The accuracy of RLS-ESN shows highest value, but in testing, which shows generality power and is the most important parameter of a training algorithm, HS-RLS-ESN and RLS-ESN are close to each other. Moreover, HS-RLS-ESN is about 40% faster than RLS-ESN. It means that HS-RLS-ESN can be used in online applications which need high accuracy. If the accuracy of HS-ESN is sufficient, it can be used as fastest training algorithm.

The HS parameters used in HS-ESN and HS-RLS-ESN are primary selected based on the suggestions in the literatures (Geem 2009, Lee et al. 2004, Wang et al. 2015) and then tuned by trial and error. Our experiments also show that the results are also robust against about 10% variations in the HS parameters.

8. CONCLUSIONS

Echo State Networks have been used successfully in a broad range of applications. In fact, their simplicity and ease of use, paired with their underlying mathematical power make them an ideal choice in many black-box modelling tasks. For many applications, however, it is mandatory to learn and adjust the ESN parameters online. In this paper the harmony search (HS) meta-heuristic search algorithm has been used for training ESN readout online. Simulation results show that the first proposed algorithm, HS-ESN, spent much less time than the other compared algorithms during training network whereas can effectively meet the requirements of the output precision.

In most practical cases, the prediction error using the proposed method are acceptable, however, the error can be reduced by using the second proposed algorithm, HS-RLS-ESN, which is firstly used HS-ESN and then its accuracy improves by HS-RLS. The generality of HS-RLS-ESN is close to HS-RLS algorithm which demonstrates the highest accuracy among all compared algorithm. The execution time of HS-RLS-ESN is lower than HS-RLS which means that both HS-RLS-ESN and HS-ESN are more suitable for online training while the generality and execution time of HS-RLS-ESN is higher than HS-ESN.

REFERENCES

Babinec, S., Pospichal, J., 2011, Modular echo state neural networks in time series prediction. *Computing and Informatics*, **30**, 321-334.

Basterrech, S., Rubino, G., 2013, Echo State Queueing Network: A new reservoir computing learning tool. In: Proc. IEEE 10th Consumer Communications and Networking Conference (CCNC), Las Vegas, VN, 11-14 Jan, 118-123.

Bianchi, F.M., Scardapane, S, Uncini, A.,Rizzi, A., Sadeghian, A., 2015, Prediction of telephone calls load using Echo State Network with exogenous variables. *Neural Networks*, **71**, 204-213.

Buehner, M., Young, P., 2006, A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, **17**, 820-824.

Bush, K., Anderson, C., 2005, Modeling reward functions for incomplete state representations via echo state networks. In: Proc. Int. Joint Conf. Neural Networks, Montreal, Canada, 2995-3000.

Chatzidimitriou, K.C., Mitkas, P.A., 2013, Adaptive reservoir computing through evolution and learning. *Neurocomputing*, **103**,198-209.

Chatzis, S., Demiris, Y., 2011, Echo state Gaussian process. *IEEE Transactions on Neural Networks*, **22**, 1435-1445.

Cui, H., Feng, C., Liu, Y., 2013, Analysis of prediction performance in wavelet minimum complexity echo state network. *The Journal of China Universities of Posts and Telecommunications*, **20**, 59-66.

Deng, Z., Zhang, Y., 2007, Collective behavior of a small-world recurrent neural system with scale-free distribution. *IEEE Transactions on Neural Networks*, **18**, 1364-1375.

Dongming, X., Jing, L., Principe, J.C., 2005, Direct adaptive control: an echo state network and genetic algorithm approach, In: Proc. IEEE Int. Joint Conf. Neural Networks, Montreal, Canada, 31 July - 4 Aug., **3**, 1483-1486.

Duan, H., Wang, X., 2016, Echo state networks with orthogonal pigeon-inspired optimization for Image restoration, *IEEE Transactions on Neural Networks and Learning Systems*, **27**, 2413-2425.

Gallicchio, C., Micheli, A., 2013, Tree echo state networks. *Neurocomputing*, **101**, 319–337.

Geem, Z.W., 2009, *Music-inspired harmony search algorithm theory and applications*. Studies in Computational Intelligence. **191**, Springe-Verlag.

Geem, Z.W., Kim, J.H., Loganathan, G.V., 2001, A new heuristic optimization algorithm: Harmony search. *Simulation*, **76**, 60-68.

Ghosn, S.B., Drouby, F., Harmanani, H.M., 2016, A parallel genetic algorithm for the open-shop scheduling problem using deterministic and random moves. *International Journal of Artificial Intelligence*, **14**, 130-144.

Han, M., Xu, M., 2015, Predicting multivariate time series using subspace echo state network. *Neural Processing Letters*, **41,** 201-209.

Heshan, W., Xuefeng, Y., 2015, Optimizing the echo state network with a binary particle swarm optimization algorithm. *Knowledge-Based Systems*, **86**, 182–193.

Holzmann, G., Hauser, H., 2010, Echo state networks with filter neurons and a delay & sum readout. *Neural Networks*, **23**, 244-256.

Hongyan, C., Chen, F., Yuan, C., Ren, L., Yunjie, L., 2014, Effect of hybrid circle reservoir injected with wavelet-neurons on performance of echo state network. *Neural Networks*,**57**, 141–151.

Hongyan, C., Xiang, L., Lixiang, L., 2012, The architecture of dynamic reservoir in the echo state network. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **22**, 033127.

Jaeger, H., 2001, *The 'echo state' approach to analysing and training recurrent neural networks*. Technical Report, Jacobs University.

Jaeger, H., 2013, *A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the 'echo state network' approach*. Technical Report, International University Bremen, Fifth revision.

Jaeger, H., 2002, *Short term memory in echo state networks*. Technical Report, Jacobs University.

Jaeger, H., 2007, *Discovering multiscale dynamical features with hierarchical echo state networks*, Technical Report, Jacobs University.

Jaeger, H., Hass, H., 2004, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication. *Science*, **304**, 78-80.

Jaeger, H., Lukosevicius, M., Popovici, D., Siewert, U., 2007, Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, **20**, 335-352.

Jaeger, H., Maass, W., Príncipe, J. C., 2007, Special issue on echo state networks and liquid state machines. *Neural Networks*, **20**, 287-289.

Kennel, M.B., Brown, R., Abarbanel, H. D. I., 1992, Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A*, **45**, 3403-3411.

Lee, K. S., Geem, Z. W., 2005, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, **194**, 3902-3933.

Lee, K. S., Geem, Z. W., 2004, A new structural optimization method based on the harmony search algorithm. *Computers and Structures*, **82**, 781-798.

Liebald, B., 2004, Exploration of effects of different network topologies on the ESN signal cross correlation matrix spectrum. Bachelor's Thesis, Jacobs University.

Lohmann, D., Butz, M. V., 2012, Balanced echo state networks. *IEEE Transactions on Neural Networks*, **36**, 35-45.

Lukosevicius, M., 2012, A practical guide to applying echo state neural networks, Tricks of the Trade, *Lecture Notes on Computer Science*, **7700**, 659-686.

Lukosevicius, M., Jaeger, H., 2009, Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, **3**, 127-149.

Lukosevicius, M., Jaeger, H., 2007, *Overview of reservoir recipes*. Technical Report, Jacobs University.

Mackey, MC., Glass, L., 1977, Oscillation and chaos in physiological control systems. *Science*, **197**, 287-289.

Mahdavi, M., Fesanghary, M., Damangir, E., 2007, An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, **188**, 1567-1579.

Manjunath, G., Jaeger, H., 2013, Echo state property linked to an input: exploring a fundamental characteristic of recurrent neural networks. *Neural Computation*, **25**, 671-697.

Martin, D., Toro, R.D., Haber, R., Dorronsoro, J., 2009, Optimal tuning of a networked linear controller using multi-objective genetic algorithm and its application to one complex electromechanical process. *International Journal of Innovative Computing, Information and Control*, **5**, 3405-3414.

Massinas, B.A., Doulamis, A., Doulamis, N., Paradissis, D., 2013, An echo state network for ionospheric disturbances behavior modeling on spaceborne interferometric synthetic aperture radar, In: Proc. AIAA Space 2013 Conference and Exposition, San Diego, USA, 10-12 Sep.

Moallem, P., Razmjooy, N., 2012, A multi layer perceptron neural network trained by invasive weed optimization for potato color image segmentation, *Trends in Applied Sciences Research* **7**, 445-455.

Najibi, E., Rostami, H., 2015, SCESN, SPESN, SWESN: Three recurrent neural echo state networks with clustered reservoirs for prediction of nonlinear and chaotic time series. *Applied Intelligence*, **43**, 460-472.

Niu, D., Ji, L., Wang, Y., Liu, D., 2012, Echo state network with wavelet in load forecasting. *Kybernetes*, **41**, 1557-1570.

Ozturk, M.C., Príncipe, J.C., 2007, An associative memory readout for ESNs with applications to dynamical pattern recognition. *Neural Networks*, **20**, 377-390.

Precup, R.-E., Tomescu, M.-L., Dragos, C.-A., 2014, Stabilization of Rossler chaotic dynamical system using fuzzy logic control algorithm. *International Journal of General Systems*, **43**, 413-433.

Precup, R.-E., Tomescu, M.-L., Radac, M.-B., Petriu, E.M., Preitl, S., Dragos, C.-A., 2012, Iterative performance improvement of fuzzy control systems for three tank systems. *Expert Systems with Applications*, **39**, 8288-8299.

Qianli, M., Weibiao, C., Jia, W., Zhiwen, Y., 2014, Direct model of memory properties and the linear reservoir topologies in echo state networks. *Applied Soft Computing*, **22**, 622–628.

Rodan, A., Tino, P., 2011, Minimum complexity echo state network. *IEEE Transaction on Neural Networks*, **22**, 131-144.

Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J., Stroobandt, D., 2008, Improving reservoirs using intrinsic plasticity. *Neurocomputing*, **71**, 1159-1171.

Skowronski, M., Harris, J., 2006, Minimum mean squared error time series classification using an echo state network prediction model. In: Proc. IEEE Int. Symp. Circuits Syst, Island of Kos, Greece, 21-24 May, 3153-3156.

Solos, I.P., Tassopoulos, I.X., Beligiannis, G.N., 2016, Optimizing shift scheduling for tank trucks using an effective stochastic variable neighbourhood approach. *International Journal of Artificial Intelligence*, **14**, 1-26.

Song, Q., Feng, Z., 2009, Stable trajectory generator-echo state network trained by particle swarm optimization. In: Proc. IEEE Int. Symp. Computational Intelligence in Robotics and Automation, Daejeon, China, 15-18 Dec, 21-26.

Song, Q., Feng, Z., 2010, Effects of connectivity structure of complex echo state network on its prediction performance for nonlinear time series. *Neurocomputing*, **73**, 2177–2185.

Song, Q., Zhao, X., Feng, Z., Song, B., 2011, Recursive least squares algorithm with adaptive forgetting factor based on echo state network. In: Proc. World Congress on Intelligent Control and Automation, Taipei, Taiwan, 21-25 June, 295-298.

Steil, J., 2007, Online reservoir adaptation by intrinsic plasticity for backpropagation decorrelation and echo state learning. *Neural Networks*, **20**, 353-364.

Tong, M. H., Bicket, A., Christiansen, E., Cottrell, G., 2007, Learning grammatical structure with echo state network. *Neural Networks*, **20**, 424-432.

Venayagamoorthy, G.K., Shishir, B., 2009, Effects of spectral radius and settling time in the performance of echo state networks. *Neural Networks*, **22**, 861–863.

Wang, J.Sh., Ping Guo, Sh.H., 2014, Echo state networks based predictive model of vinyl chloride monomer convention velocity optimized by artificial fish swarm algorithm. *Soft Computing*, **18**, 457-468.

Wang, S., Yang, X., Wei, C. J., 2006, Harnessing non-linearity by sigmoid-wavelet hybrid echo state networks (SWHESN). In: Proc. 6th World Congress on Intelligent Control and Automation, Dalian, China, June 21-23.

Wang, X., Gao, X.-Zh., Zenger, K., 2015, *An Introduction to Harmony Search Optimization Method*, Springer Briefs in Computational Intelligence, Springer International Publishing.

Yildiz, I. B., Jaeger, H., Kiebel, S. J., 2012, Re-visiting the echo state property. *Neural Networks*, **35**, 1-9.

Zhang, B., Wang, Y., 2008, Echo state networks with decoupled reservoir states. In: Proc. IEEE Workshop Mach. Learn. Signal Process, Cancun, Mexico, 16-19 Oct, 444-449.

Zhang, L., Hua, C., Tang, Y., Guan, X., 2016, Ill-posed echo state network based on L-curve method for prediction of blast furnace gas flow. *Neural Processing Letters*, **43**, 97-113.