Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 504120, 10 pages
http://dx.doi.org/10.1155/2015/504120

Research Article

Particle Swarm Optimization Based Selective Ensemble of Online Sequential Extreme Learning Machine

Yang Liu,[1] Bo He,[1] Diya Dong,[1] Yue Shen,[1] Tianhong Yan,[2] Rui Nian,[1] and Amaury Lendasse[3,4]

[1] *School of Information Science and Engineering, Ocean University of China, 238 Songling Road, Qingdao 266100, China*
[2] *School of Mechanical and Electrical Engineering, China Jiliang University, 258 Xueyuan Street, Xiasha High-Edu Park, Hangzhou 310018, China*
[3] *Department of Mechanical and Industrial Engineering and the Iowa Informatics Initiative, 3131 Seamans Center, The University of Iowa, Iowa City, IA 52242-1527, USA*
[4] *Arcada University of Applied Sciences, 00550 Helsinki, Finland*

Correspondence should be addressed to Bo He; bhe@ouc.edu.cn and Tianhong Yan; thyan@163.com

Received 7 August 2014; Revised 1 November 2014; Accepted 5 November 2014

Academic Editor: Zhan-li Sun

A novel particle swarm optimization based selective ensemble (PSOSEN) of online sequential extreme learning machine (OS-ELM) is proposed. It is based on the original OS-ELM with an adaptive selective ensemble framework. Two novel insights are proposed in this paper. First, a novel selective ensemble algorithm referred to as particle swarm optimization selective ensemble is proposed, noting that PSOSEN is a general selective ensemble method which is applicable to any learning algorithms, including batch learning and online learning. Second, an adaptive selective ensemble framework for online learning is designed to balance the accuracy and speed of the algorithm. Experiments for both regression and classification problems with UCI data sets are carried out. Comparisons between OS-ELM, simple ensemble OS-ELM (EOS-ELM), genetic algorithm based selective ensemble (GASEN) of OS-ELM, and the proposed particle swarm optimization based selective ensemble of OS-ELM empirically show that the proposed algorithm achieves good generalization performance and fast learning speed.

1. Introduction

Feedforward neural network is one of the most prevailing neural networks for data processing in the past decades [1, 2]. However, the slow learning speed limits its applications. Recently, an original algorithm designed for single hidden layer feedforward neural networks (SLFNs) named extreme learning machine (ELM) was proposed by Huang et al. [3]. ELM is a tuning free algorithm for it randomly selects the input weights and biases of the hidden nodes instead of learning these parameters. And, also, the output weights of the network are then analytically determined. ELM proves to be a few orders faster than traditional learning algorithms and obtains better generalization performance as well. It lets the fast and accurate data analytics become possible and has been applied to many fields [4–6].

However, the algorithms mentioned above need all the training data available to build the model, which is referred to as batch learning. In many industrial applications, it is very common that the training data can only be obtained one by one or chunk by chunk. If batch learning algorithms are performed each time new training data is available, the learning process will be very time consuming. Hence online learning is necessary for many real world applications.

An online sequential extreme learning machine is then proposed by Liang et al. [7]. OS-ELM can learn the sequential training observations online at arbitrary length (one by one or chunk by chunk). New arrived training observations are learned to update the model of the SLFNs. As soon as the learning procedure for the arrived observations is completed, the data is discarded. Moreover, it has no prior knowledge about the amount of the observations which will be presented. Therefore, OS-ELM is an elegant online learning algorithm which can handle both the RBF and additive nodes in the same framework and can be used to both the classification and function regression problems. OS-ELM proves to be a very fast and accurate online sequential learning algorithm [8–10], which can provide better generalization performance in faster speed compared with other online learning algorithms such as GAP-RBF, GGAP-RBF, SGBP, RAN, RANEKF, and MRAN.

However, due to the random generation of the parameters for the hidden nodes, the generalization performance of OS-ELM sometimes cannot be guaranteed, similar to ELM. Some ensemble based methods have been applied to ELM to improve its accuracy [11–13]. Ensemble learning is a learning scheme where a collection of a finite number of learners are trained for the same task [14, 15]. It has been demonstrated that the generalization ability of a learner can be significantly improved by ensembling a set of learners. In [16] a simple ensemble OS-ELM, that is, EOS-ELM, has been investigated. However, Zhou et al. [17] proved that selective ensemble is a better choice. We apply this idea to OS-ELM. At first, a novel selective ensemble algorithm, termed as PSOSEN, is proposed. PSOSEN adopts particle swarm optimization [18] to select the individual OS-ELMs to form the ensemble. Benefiting from the fast speed of PSO, PSOSEN is designed to be a new accurate and fast selective ensemble algorithm. It should be noted that PSOSEN is a general selective ensemble algorithm suitable for any learning algorithms.

Different from batch learning, online learning algorithms need to perform learning continually. Therefore the complexity of the learning algorithm should be taken into account. Obviously, performing selective ensemble learning each step is not a good choice for online learning. Thus we designed an adaptive selective ensemble framework for OS-ELM. A set of OS-ELMs are trained online, and the root mean square error (RMSE) will always be calculated. The error will be compared with a preset threshold λ. If RMSE is bigger than the threshold, it means the model is not accurate. Then PSOSEN will be performed and a selective ensemble M is obtained. Otherwise, it means the model is relatively accurate and the ensemble will not be selected. Then the output of the system is calculated as the average of the individuals in the ensemble set. And each individual OS-ELM will be updated recursively.

UCI data sets [19], which contain both regression and classification data, are used to verify the feasibility of the proposed algorithm. Comparisons of three aspects including RMSE, standard deviation and running time between OS-ELM, and EOS-ELM, selective ensemble of OS-ELM (SEOS-ELM) with both GASEN and PSOSEN are presented. The results convincingly show that PSOSEN achieves better generalization accuracy and fast learning speed.

The rest of the paper is organized as follows. In Section 2, previous work including ELM and OS-ELM is reviewed. The novel selective ensemble based on particle swarm optimization is presented in Section 3. An adaptive selective ensemble framework is designed for OS-ELM in Section 4. Experiments are carried out in Section 5 and the comparison results are also presented. In Section 6, further discussion about PSOSEN is provided. We draw the conclusion of the paper in Section 7.

2. Review of Related Work

In this section, both the basic ELM algorithm and the online version OS-ELM are reviewed in brief as the background knowledge for our work.

2.1. Extreme Learning Machine (ELM). ELM algorithm is derived from single hidden layer feedforward neural networks (SLFNs). Unlike traditional SLFNs, ELM assigns the parameters of the hidden nodes randomly without any iterative tuning. Besides, all the parameters of the hidden nodes in ELM are independent of each other. Hence ELM can be seen as generalized SLFNs.

Given N training samples $(x_i, t_i) \in R^n \times R^m$, where x_i is an input vector of n dimensions and t_i is a target vector of m dimensions. Then SLFNs with $\widetilde{N}$ hidden nodes each with output function $G(a_i, b_i, x)$ are mathematically modeled as

$$f_{\widetilde{N}}\left(x_j\right) = \sum_{i=1}^{\widetilde{N}} \beta_i G\left(a_i, b_i, x_j\right) = t_j, \quad j = 1, \ldots, N, \tag{1}$$

where (a_i, b_i) are parameters of hidden nodes, and β_i is the weight vector connecting the ith hidden node and the output node. To simplify, (1) can be written equivalently as

$$H\beta = T, \tag{2}$$

where

$$H\left(a_1, \ldots, a_N, b_1, \ldots, b_{\widetilde{N}}, x_1, \ldots, x_N\right) = \begin{bmatrix} G\left(a_1, b_1, x_1\right) & \cdots & G\left(a_{\widetilde{N}}, b_{\widetilde{N}}, x_1\right) \\ \vdots & \ddots & \vdots \\ G\left(a_1, b_1, x_N\right) & \cdots & G\left(a_{\widetilde{N}}, b_{\widetilde{N}}, x_N\right) \end{bmatrix}_{N \times \widetilde{N}}, \tag{3}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\widetilde{N}}^T \end{bmatrix}_{\widetilde{N} \times m} \qquad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m},$$

H is called the hidden layer output matrix of the neural network, and the ith column of H is the output of the ith hidden node with respect to inputs $x_1, x_2, \ldots, x_N$.

In ELM, H can be easily obtained as long as the training set is available and the parameters (a_i, b_i) are randomly assigned. Then ELM evolves into a linear system and the output weights β are calculated as

$$\widehat{\beta} = H^{\dagger} T, \tag{4}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H.

The ELM algorithm can be summarized in three steps as shown in Algorithm 1.

```
Input:
A training set $N = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \ldots, N\}$, hidden node output function
$G(a_i, b_i, x)$, and the number of hidden nodes $\widetilde{N}$.
Steps:
(1) Assign parameters of hidden nodes $(a_i, b_i)$ randomly, $i = 1, \ldots, \widetilde{N}$.
(2) Calculate the hidden layer output matrix $H$.
(3) Calculate the output weight $\beta : \hat{\beta} = H^{\dagger} T$, where $H^{\dagger}$ is the Moore-Penrose generalized
inverse of hidden layer output matrix $H$.
```

Algorithm 1

2.2. OS-ELM. In many industrial applications, it is impossible to have all the training data available before the learning process. It is common that the training observations are sequentially inputted to the learning algorithm; that is, the observations arrive one-by-one or chunk-by-chunk. In this case, the batch ELM algorithm is no longer applicable. Hence, a fast and accurate online sequential extreme learning machine was proposed to deal with online learning.

The output weight β obtained from (4) is actually a least-squares solution of (2). Given $\operatorname{rank}(H) = \widetilde{N}$, the number of hidden nodes, $H^{\dagger}$ can be presented as

$$H^{\dagger} = \left(H^T H\right)^{-1} H^T. \tag{5}$$

This can also be called the left pseudoinverse of H for it satisfies the equation $H^{\dagger} H = I_{\widetilde{N}}$. If $H^T H$ tends to be singular, smaller network size $\widetilde{N}$ and larger data number N_0 should be chosen in the initialization step of OS-ELM. Substituting (5) to (4), we can get

$$\widehat{\beta} = \left(H^T H\right)^{-1} H^T T \tag{6}$$

which is the least-squares solution to (2). Then the OS-ELM algorithm can be deduced by recursive implementation of the least-squares solution of (6).

There are two main steps in OS-ELM, initialization step and update step. In the initialization step, the number of training data N_0 needed in this step should be equal to or larger than network size $\widetilde{N}$. In the update step, the learning model is updated with the method of recursive least square (RLS). And only the newly arrived single or chunk training observations are learned, which will be discarded as soon as the learning step is completed.

The two steps for OS-ELM algorithm in general are as follows.

(a) Initialization step: batch ELM is used to initialize the learning system with a small chunk of initial training data $\mathbb{N}_0 = \{(x_i, t_i)\}_{i=1}^{N_0}$ from given training set

$$\mathbb{N} = \left\{(x_i, t_i),\ x_i \in R^n,\ t_i \in R^m,\ i = 1, \ldots\right\}, \quad N_0 \geq \widetilde{N}. \tag{7}$$

(1) Assign random input weights a_i and bias b_i (for additive hidden nodes) or center a_i and impact factor b_i (for RBF hidden nodes), $i = 1, \ldots, \widetilde{N}$.

(2) Calculate the initial hidden layer output matrix:

$$H_0 = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_{\widetilde{N}}, b_{\widetilde{N}}, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_{N_0}) & \cdots & G(a_{\widetilde{N}}, b_{\widetilde{N}}, x_{N_0}) \end{bmatrix}_{N_0 \times \widetilde{N}}. \tag{8}$$

(3) Calculate the initial output weight $\beta^{(0)} = P_0 H_0^T T_0$, where $P_0 = (H_0^T H_0)^{-1}$ and $T_0 = [t_1, \ldots, t_{N_0}]^T$.

(4) Set $k = 0$. Initialization is finished.

(b) Sequential learning step is as follows.

The $(k+1)$th chunk of new observations can be expressed as

$$\mathbb{N}_{k+1} = \left\{(x_i, t_i)\right\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j}, \tag{9}$$

where $\mathbb{N}_{k+1}$ represents the number of observations in the $(k+1)$th chunk newly arrived.

(1) Compute the partial hidden layer output matrix H_{k+1} for the $(k+1)$th chunk:

$$H_{k+1} = \begin{bmatrix} G\left(a_1, b_1, x_{(\sum_{j=0}^{k} N_j)+1}\right) & \cdots & G\left(a_{\widetilde{N}}, b_{\widetilde{N}}, x_{(\sum_{j=0}^{k} N_j)+1}\right) \\ \vdots & \ddots & \vdots \\ G\left(a_1, b_1, x_{\sum_{j=0}^{k+1} N_j}\right) & \cdots & G\left(a_{\widetilde{N}}, b_{\widetilde{N}}, x_{\sum_{j=0}^{k+1} N_j}\right) \end{bmatrix}_{N_{k+1} \times \widetilde{N}}. \tag{10}$$

(2) Set $T_{k+1} = [t_{(\sum_{j=0}^{k} N_j)+1}, \ldots, t_{\sum_{j=0}^{k+1} N_j}]^T$. And we have

$$\begin{aligned} K_{k+1} &= K_k + H_{k+1}^T H_{k+1}, \\ \beta^{(k+1)} &= \beta^{(k)} + K_{k+1}^{-1} H_{k+1}^T \left(T_{k+1} - H_{k+1} \beta^{(k)}\right). \end{aligned} \tag{11}$$

To avoid calculating inverse in the iterative procedure, K_{k+1}^{-1} is factored as the following according to Woodbury formula:

$$\begin{aligned} K_{k+1}^{-1} &= \left(K_k + H_{k+1}^T H_{k+1}\right)^{-1} \\ &= K_k^{-1} - K_k^{-1} H_{k+1}^T \left(I + H_{k+1} K_k^{-1} H_{k+1}^T\right)^{-1} H_{k+1} K_k^{-1}. \end{aligned} \tag{12}$$

Let $P_{k+1} = K_{k+1}^{-1}$.

(3) Calculate the output weight $\beta^{(k+1)}$, according to the updating equations:

$$P_{k+1} = P_k - P_k H_{k+1}^T \left(I + H_{k+1} P_k H_{k+1}^T\right)^{-1} H_{k+1} P_k, \quad \beta^{(k+1)} = \beta^{(k)} + P_{k+1} H_{k+1}^T \left(T_{k+1} - H_{k+1}\beta^{(k)}\right). \tag{13}$$

(4) Set $k = k + 1$. Go to step (b).

3. Particle Swarm Optimization Selective Ensemble

In this section, a novel selective ensemble method referred to as particle swarm optimization selective ensemble (PSOSEN) is proposed. PSOSEN adopts particle swarm optimization to select the good learners and combine their predictions. Detailed procedures of the PSOSEN algorithm will be introduced in this section.

A remarkable superiority of PSOSEN is its speed over other selective ensemble algorithms. Another popular selective ensemble learning method is based on genetic algorithm. Compared with GASEN, PSOSEN achieves faster convergence to optimal solution due to the omission of crossover and mutation operations used in GASEN. GASEN is actually quite complicated for the requirement of encode, decode, and other genetic operations. For instance, GASEN only works with binary encoding, while PSOSEN is available for any forms of values based on their current positions and velocity vectors in the corresponding hyperspace. For PSOSEN, there is no need for overmuch parameter adjustment, thus easy to implement. Although using simple method, PSOSEN is still capable of obtaining high accuracy of prediction and reaching the optima earlier than GASEN. Furthermore, PSO is less influenced by changes in problem dimensionality or modality of problems compared with GA, which also proves to be robust in most situations [20].

As selective ensemble is usually more time-consuming than original algorithm, a faster optimization method might be preferable. For this purpose, PSOSEN might be more appropriate to be adopted to search for the optimal ensemble of ELM models efficiently.

Zhou et al. [17] have demonstrated that ensembling many of the available learners may be better than ensembling all of those learners in both regression and classification. The detailed proof of this conclusion will not be presented in this paper. However, one important problem for selective ensemble is how to select the good learners in a set of available learners.

The novel approach selective ensemble algorithm is proposed to select good learners in the ensemble. PSOSEN is based on the idea of heuristics. It assumes each learner can be assigned a weight, which could characterize the fitness of including this learner in the ensemble. Then the learner with the weight bigger than a preset threshold λ could be selected to join the ensemble.

We will explain the principle of PSOSEN from the context of regression. We use ω_i to denote the weight of the ith component learner. The weight should satisfy the following equations:

$$0 \le \omega_i \le 1, \quad \sum_{i=1}^{N} \omega_i = 1. \tag{14}$$

Then the weight vector is

$$\omega = \left(\omega_1, \omega_2, \ldots, \omega_N\right). \tag{15}$$

Suppose input variables $x \in R^m$ according to the distribution $p(x)$, the true output of x is $d(x)$, and the actual output of the ith learner is $f_i(x)$. Then the output of the simple weighted ensemble on x is

$$\hat{f}(x) = \sum_{i=1}^{N} \omega_i f_i(x). \tag{16}$$

Then the generalization error $E_i(x)$ of the ith learner and the generalization error $\hat{E}(x)$ of the ensemble are calculated on x, respectively:

$$E_i(x) = \left(f_i(x) - d(x)\right)^2, \quad \hat{E}(x) = \left(\hat{f}(x) - d(x)\right)^2. \tag{17}$$

The generalization error E_i of the ith learner and that of the ensemble $\hat{E}$ is calculated on $p(x)$, respectively:

$$E_i = \int dx p(x) E_i(x), \quad \hat{E} = \int dx p(x) \hat{E}(x). \tag{18}$$

We then define the correlation between the ith and the jth component learner as follows:

$$C_{ij} = \int dx p(x) \left(f_i(x) - d(x)\right)\left(f_j(x) - d(x)\right). \tag{19}$$

Obviously C_{ij} satisfies the following equations:

$$C_{ii} = E_i, \quad C_{ij} = C_{ji}. \tag{20}$$

Considering the equations defined above, we can get

$$\hat{E}(x) = \left(\sum_{i=1}^{N} \omega_i f_i(x) - d(x)\right)\left(\sum_{j=1}^{N} \omega_j f_j(x) - d(x)\right), \tag{21}$$

$$\hat{E} = \sum_{i=1}^{N}\sum_{j=1}^{N} \omega_i \omega_j C_{ij}. \tag{22}$$

To minimize the generalization error of the ensemble, according to (22), the optimum weight vector can be obtained as

$$\omega_{\text{opt}} = \underset{\omega}{\operatorname{argmin}} \left(\sum_{i=1}^{N}\sum_{j=1}^{N} \omega_i \omega_j C_{ij}\right). \tag{23}$$

The kth variable of ω_{opt}, that is, $\omega_{\text{opt}\cdot k}$, can be solved by Lagrange multiplier:

$$\frac{\partial\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_i\omega_j C_{ij} - 2*\lambda\left(\sum_{i=1}^{N}\omega_i - 1\right)\right)}{\partial\omega_{\text{opt}\cdot k}} = 0. \quad (24)$$

The equation can be simplified to

$$\sum_{\substack{j=1\\ j\neq k}}^{N}\omega_{\text{opt}\cdot k}C_{kj} = \lambda. \quad (25)$$

Taking (2) into account, we can get

$$\omega_{\text{opt}\cdot k} = \frac{\sum_{j=1}^{N} C_{kj}^{-1}}{\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_i\omega_j C_{ij}^{-1}}. \quad (26)$$

Equation (26) gives the direct solution for ω_{opt}. But the solution seldom works well in real world applications. Due to the fact that some learners are quite similar in performance, when a number of learners are available, the correlation matrix C_{ij} may be irreversible or ill-conditioned.

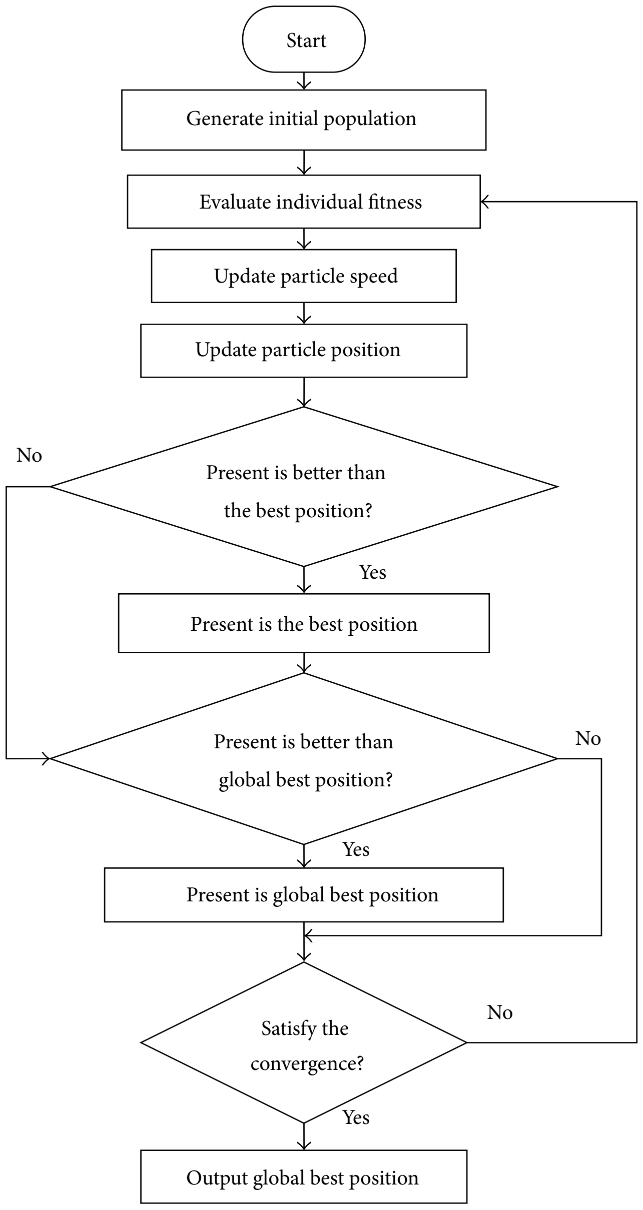
Although we cannot obtain the optimum weights of the learner directly, we can approximate them in some way. Equation (23) can be viewed as an optimization problem. As particle swarm optimization has been proved to be a powerful optimization tool, PSOSEN is then proposed. The basic PSO algorithm is shown in Figure 1.

PSOSEN randomly assigns a weight to each of the available learners at first. Then it employs particle swarm optimization algorithm to evolve those weights so that the weights can characterize the fitness of the learners in joining the ensemble. Finally, learners whose weight is bigger than a preset threshold λ are selected to form the ensemble. Note that if all the evolved weights are bigger than the threshold λ, then all the learners will be selected to join the ensemble.

PSOSEN can be applied to both regression and classification problems for the purpose of the weights evolving process which is only to select the component learners. In particular, the outputs of the ensemble for regression are combined via simple averaging instead of weighted averaging. The reason is that previous work [17] showed that using the weights both in selection of the component learners and in combination of the outputs tends to suffer the overfitting problem.

In the process of generating population, the goodness of the individuals is evaluated via validation data bootstrap sampled from the training data set. We use $\widehat{E}_{\omega}^{V}$ to denote the generalization error of the ensemble, which corresponds to individual ω on the validation data V. Obviously $\widehat{E}_{\omega}^{V}$ can describe the goodness of ω. The smaller $\widehat{E}_{\omega}^{V}$ is, the better ω is. So, PSOSEN adopts $f(\omega) = 1/\widehat{E}_{\omega}^{V}$ as the fitness function.

The PSOSEN algorithm is summarized as follows. $S_1, S_2, \ldots, S_T$ are bootstrap samples generated from original training data set. A component learner N_t is trained from each S_T. And a selective ensemble N^* is built from $N_1, N_2, \ldots, N_T$. The output is the average output of the ensemble for regression or the class label who receives the most number in voting process for classification (see Algorithm 2).



Start
Generate initial population
Evaluate individual fitness
Update particle speed
Update particle position
Present is better than the best position?
No
Yes
Present is the best position
Present is better than global best position?
No
Yes
Present is global best position
Satisfy the convergence?
No
Yes
Output global best position

Figure 1: Flowchart for particle swarm optimization algorithm.

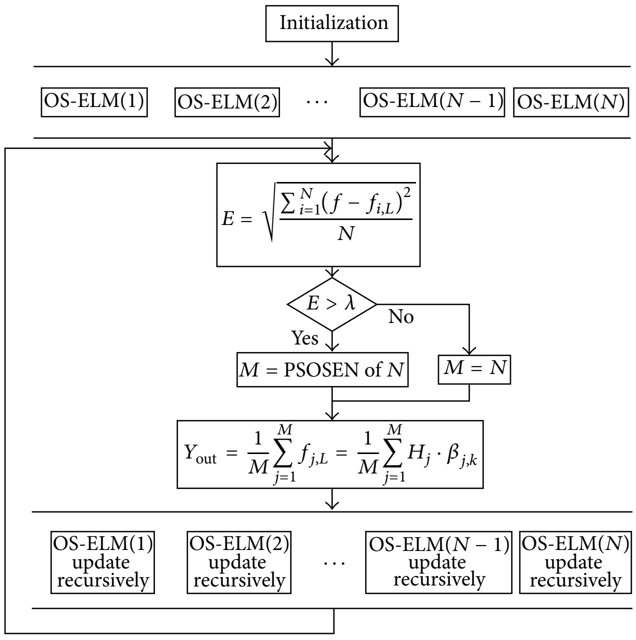
4. Particle Swarm Optimization Based Selective Ensemble of Online Sequential Extreme Learning Machine

In this section, PSOSEN is applied to the original OS-ELM to improve the generalization performance. In order to reduce the complexity and employ PSOSEN flexibly, an adaptive framework is then designed. The flowchart of the framework is shown as in Figure 2.

Online learning is necessary in many industrial applications. In these situations, training data can only be obtained sequentially. Although OS-ELM is proposed as useful online learning algorithm, the generalization performance may not be quite good results from the random generation of the input

```
Input: training set S, learner L, trial T, threshold $\lambda$
Steps:
(1) for $t = 1$ to $T${
    $S_T$ = bootstrap sample from $S$
    $N_T = L(S_T)$
}
(2) generate a population of weight vectors
(3) evolve the population by PSO, where the fitness of the weight vector $\omega$ is defined as
$f(\omega) = 1/\hat{E}^{V}_{\omega}$.
(4) $\omega^*$ = the evolved best weight vector
Output: ensemble $N^*$:
  $N^*(x) = \text{Ave} \sum_{\omega_i^* > \lambda} N_t(x)$ for regression
  $N^*(x) = \arg\max_{y \in Y} \sum_{\omega_i^* > \lambda, N_t(x) = y} 1$ for classification
```

Algorithm 2: PSOSEN.



Initialization
OS-ELM(1)
OS-ELM(2)
···
OS-ELM(N − 1)
OS-ELM(N)
$E = \sqrt{\frac{\sum_{i=1}^{N}(f - f_{i,L})^2}{N}}$
$E > \lambda$
No
Yes
M = PSOSEN of N
$M = N$
$Y_{\text{out}} = \frac{1}{M}\sum_{j=1}^{M} f_{j,L} = \frac{1}{M}\sum_{j=1}^{M} H_j \cdot \beta_{j,k}$
OS-ELM(1) update recursively
OS-ELM(2) update recursively
OS-ELM(N − 1) update recursively
OS-ELM(N) update recursively

Figure 2: Flowchart for the proposed framework.

parameters. Ensemble methods have been investigated in OS-ELM, that is, the EOS-ELM algorithm [16]. However, it is only very simple ensemble method, which just calculates the average of all the N individual OS-ELMs. In this section, selective ensemble, which is superior to simple ensemble, is applied to OS-ELM. The novel selective ensemble method proposed in Section 3 is adopted. Apparently, performing PSOSEN each step is a time consuming process. We design an adaptive framework to determine whether to perform PSOSEN or simple ensemble. Thus the accuracy and the complexity can be balanced well. The framework for the new algorithm can be explained as follows.

First, N individual OS-ELMs are initialized. The number of nodes is same for each OS-ELM, while the input weights and biases for each OS-ELM are randomly generated.

Second, the RMSE error is calculated:

$$E = \sqrt{\frac{\sum_{i=1}^{N}(f - f_{i,L})^2}{N}}, \tag{27}$$

where f is the expected output, while $f_{i,L}$ is the actual output of the ith individual OS-ELM.

The RMSE will be compared with a preset threshold λ. If E is bigger than λ, which means simple ensemble is not accurate, PSOSEN is performed and a selective ensemble M is obtained. And if E is smaller than λ, which indicates that simple ensemble is relatively accurate, the ensemble will not be selected.

Third, the output of the system is calculated as the average output of the individual in the ensemble set:

$$Y_{\text{out}} = \frac{1}{M}\sum_{j=1}^{M} f_{j,L} = \frac{1}{M}\sum_{j=1}^{M} H_j \cdot \beta_{j,k}, \tag{28}$$

where H_j is the output matrix of the jth OS-ELM, and $\beta_{j,k}$ is the output weight calculated by the jth OS-ELM at step k.

At last, each OS-ELM will update recursively according to the update equations presented in Section 2.

5. Performance Evaluation of PSOSEN Based OS-ELM

In this section, a series of experiments were conducted to evaluate the performance of the proposed algorithm. OS-ELM, EOS-ELM, and GASEN based OS-ELM are also compared with the new algorithm in this section. All the experiments were carried out in the MATLAB R2012b environment on a desktop of CPU 3.40 GHz and 8 GB RAM.

5.1. Model Selection. For OS-ELM, the number of hidden nodes is the only parameter that needs to be determined. Cross-validation method is usually used to choose this parameter. Fifty trials of simulations are performed, respectively, for regression and classification problems. The number of hidden nodes is then determined by the validation error.

Table 1: Network selection for New-thyroid dataset.

Number of networks	1	5	10	15	20	25	30
Testing accuracy	90.73	91.25	90.65	90.18	92.24	91.79	91.8
Testing dev.	0.0745	0.0254	0.0316	0.0276	0.0138	0.024	0.0156

Table 2: Specification of benchmark datasets.

Dataset	Classes	Training data	Testing data	Attributes
Regression problems				
Abalone	—	3000	1177	8
California housing	—	8000	12640	8
Mackey-Glass	—	4000	500	4
Classification problems				
New-thyroid	3	140	75	5
Image segmentation	7	1500	810	19
Satellite image	6	4435	2000	36

For EOS-ELM, SEOS-ELM (GASEN), and SEOS-ELM (PSOSEN), there is another parameter that needs to be determined, that is, the number of networks in the ensemble. The parameter is set from 5 to 30 with the interval 5. Finally, the optimal parameter is selected according to the RMSE for regression, testing accuracy for classification, and standard deviation value. Under the same problem, the number of OS-ELMs is selected based on the lowest standard deviation and the comparable RMSE or accuracy compared with OS-ELM. Table 1 is an example of selecting the optimal number of networks for SEOS-ELM (PSOSEN) with RBF hidden nodes on New-thyroid dataset. As illustrated by Table 1, the lowest standard deviation occurs when the number of OS-ELMs is 20. Meanwhile, the prediction accuracy of SEOS-ELM is better than OS-ELM. Hence we set the number of networks to be 20 for the New-thyroid dataset. The numbers of OS-ELMs for other datasets are determined in the same way.

Both the Gaussian radial basis function (RBF) $G(a, b, x) = \exp(-\|x-a\|^2/b)$ and the sigmoid additive $G(a, b, x) = 1/(1 + \exp(-(a \cdot x + b)))$ are adopted as activation function in OS-ELM, EOS-ELM, SEOS-ELM (GASEN), and SEOS-ELM (PSOSEN).

In the experiments, OS-ELM, EOS-ELM, and SEOS-ELM (GASEN) were compared with SEOS-ELM (PSOSEN). Some general information of the benchmark datasets used in our evaluations is listed in Table 2. Both regression and classification problems are included.

For OS-ELM, the input weights and biases with additive activation function or the centers with RBF activation function were all generated from the range $[-1, 1]$. For regression problems, all the inputs and outputs were normalized into the range $[0, 1]$, while the inputs and outputs were normalized into the range $[-1, 1]$ for classification problems.

The benchmark datasets studied in the experiments are from UCI Machine Learning Repository except California Housing dataset from the StatLib Repository. Besides, a time-series problem, Mackey-Glass, from UCI was also adopted to test our algorithms.

5.2. Algorithm Evaluation. To verify the superiority of the proposed algorithm, RMSE for regression problems and testing accuracy for classification problems are, respectively, computed. The initial size of the dataset is very small, which equals to the number of the hidden nodes to guarantee the model to work. All the data then is sent to the model in a one-by-one learning mode. The evaluation results are presented in Tables 3, 4, 5, and 6, which are, respectively, corresponding to the models with sigmoid hidden nodes and RBF hidden nodes for both regression and classification problems. Each result is an average of 50 trials. And in every trial of one problem, the training and testing samples were randomly adopted from the dataset that was addressed currently.

From the comparison results of four tables, we can easily find that EOS-ELM, SEOS-ELM (GASEN), and SEOS-ELM (PSOSEN) are more time consuming than OS-ELM, but they still keep relatively fast speed at most of the time. It should be noted that the complexity of SEOS-ELM is adjustable, which depends on the threshold λ.

What is important, EOS-ELM, SEOS-ELM (GASEN), and SEOS-ELM (PSOSEN) all attain lower testing deviation and more accurate regression or classification results than OS-ELM, which shows the advantage of ensemble learning. In addition, both SEOS-ELM (GASEN) and SEOS-ELM (PSOSEN) are more accurate than EOS-ELM. This verifies that selective ensemble is better than simple ensemble method.

In terms of the comparison between SEOS-ELM (GASEN) and SEOS-ELM (PSOSEN), it can be observed that both of the two selective ensemble algorithms achieve comparable accuracy. However, the advantage of the new algorithm is that it is more computational efficient. This verifies that PSOSEN is a fast and accurate selective ensemble algorithm.

As an online learning algorithm, the online learning ability is another important evaluation criterion. To illustrate the online learning ability of the proposed algorithm, a simulated regression dataset is adopted. The dataset was generated from the function $y = x^2 + 3x + 2$, comprising 4500 training data and 1000 testing data. Noting that this function is chosen arbitrarily just to simulate the regression problem, Figures 3 and 4 explicitly depict the variability of training accuracy of SEOS-ELM (PSOSEN), EOS-ELM, and OS-ELM with respect to the number of training data in the process of learning. It can be observed that with the increasing number of training samples, RMSE values of the three methods significantly decline. As the online learning progressed, the training models are continuously updated and corrected. We can then conclude that the more training data the system learns, the more precise the model is. Whether sigmoid or RBF the hidden nodes is, SEOS-ELM always obtains smaller RMSE than EOS-ELM and OS-ELM, which indicates that the performance of SEOS-ELM is considerably accurate compared with the other methods. Moreover, the smaller testing deviation of SEOS-ELM in

Table 3: Comparison of algorithms for regression problems with sigmoid hidden nodes.

Datasets	Algorithm	Number of nodes	Number of networks	Training time (s)	RMSE or Accuracy		Testing dev.
					Training RMSE	Testing RMSE	
Abalone	OS-ELM	25		0.1191	0.0758	0.0782	0.0049
	EOS-ELM	25	5	0.5942	0.0754	0.0775	0.0023
	SEOS-ELM (GASEN)	25	5	7.3864	0.0744	0.0760	0.0016
	SEOS-ELM (PSOSEN)	25	5	4.1528	0.0742	0.0758	0.0015
Mackey-Glass	OS-ELM	120		0.9827	0.0177	0.0185	0.0018
	EOS-ELM	120	5	4.8062	0.0176	0.0183	0.0007
	SEOS-ELM (GASEN)	120	5	37.4371	0.0172	0.0180	0.0005
	SEOS-ELM (PSOSEN)	120	5	25.1608	0.0173	0.0179	0.0006
California Housing	OS-ELM	50		0.6871	0.1276	0.1335	0.0035
	EOS-ELM	50	5	3.2356	0.1280	0.1337	0.0019
	SEOS-ELM (GASEN)	50	5	20.7635	0.1242	0.1321	0.0014
	SEOS-ELM (PSOSEN)	50	5	15.6326	0.1238	0.1323	0.0014

Table 4: Comparison of algorithms for classification problems with sigmoid hidden nodes.

Datasets	Algorithm	Number of nodes	Number of networks	Training time (s)	RMSE or Accuracy		Testing dev.
New-thyroid	OS-ELM	20		0.0043	93.18%	89.66%	0.1138
	EOS-ELM	20	15	0.0627	94.32%	90.92%	0.0276
	SEOS-ELM (GASEN)	20	15	1.2476	95.14%	91.58%	0.0201
	SEOS-ELM (PSOSEN)	20	15	0.5012	95.23%	91.78%	0.0198
Image segmentation	OS-ELM	180		1.8432	97.07%	94.83%	0.0078
	EOS-ELM	180	20	36.2458	97.08%	94.79%	0.0055
	SEOS-ELM (GASEN)	180	20	432.1987	97.60%	95.12%	0.0045
	SEOS-ELM (PSOSEN)	180	20	254.0721	97.56%	95.21%	0.0043
Satellite image	OS-ELM	400		42.2503	92.82%	88.92%	0.0058
	EOS-ELM	400	20	853.2675	92.80%	89.05%	0.0026
	SEOS-ELM (GASEN)	400	20	8241.4093	93.54%	89.92%	0.0017
	SEOS-ELM (PSOSEN)	400	20	6928.0968	93.96%	90.16%	0.0018

Table 5: Comparison of algorithms for regression problems with RBF hidden nodes.

Datasets	Algorithm	Number of nodes	Number of networks	Training time (s)	RMSE or Accuracy		Testing dev.
					Training RMSE	Testing RMSE	
Abalone	OS-ELM	25		0.3445	0.0753	0.0775	0.0027
	EOS-ELM	25	25	8.5762	0.0752	0.0773	0.0023
	SEOS-ELM (GASEN)	25	25	54.2453	0.0742	0.0760	0.0016
	SEOS-ELM (PSOSEN)	25	25	49.3562	0.0741	0.0761	0.0017
Mackey-Glass	OS-ELM	120		1.6854	0.0181	0.0185	0.0092
	EOS-ELM	120	5	8.4304	0.0171	0.0171	0.0028
	SEOS-ELM (GASEN)	120	5	79.3216	0.0155	0.0158	0.0021
	SEOS-ELM (PSOSEN)	120	5	55.1469	0.0159	0.0156	0.0016
California Housing	OS-ELM	50		1.8329	0.1298	0.1317	0.0017
	EOS-ELM	50	5	9.0726	0.1296	0.1316	0.0011
	SEOS-ELM (GASEN)	50	5	69.8636	0.1216	0.1262	0.0008
	SEOS-ELM (PSOSEN)	50	5	64.9625	0.1202	0.1243	0.0009

Table 6: Comparison of algorithms for classification problems with RBF hidden nodes.

Datasets	Algorithm	Number of nodes	Number of networks	Training time (s)	RMSE or Accuracy		Testing dev.
New-thyroid	OS-ELM	20		0.0118	93.45%	89.92%	0.0702
	EOS-ELM	20	15	0.1682	93.87%	89.86%	0.0428
	SEOS-ELM (GASEN)	20	15	1.9745	94.53%	91.05%	0.0332
	SEOS-ELM (PSOSEN)	20	15	1.2315	94.68%	91.32%	0.0315
Image segmentation	OS-ELM	180		2.6702	94.98%	91.92%	0.0324
	EOS-ELM	180	5	13.2174	94.39%	91.35%	0.0148
	SEOS-ELM (GASEN)	180	5	128.3215	95.62%	95.06%	0.0085
	SEOS-ELM (PSOSEN)	180	5	90.2856	96.02%	95.24%	0.0079
Satellite image	OS-ELM	400		45.2702	93.62%	89.54%	0.0056
	EOS-ELM	400	10	448.1347	93.86%	89.37%	0.0034
	SEOS-ELM (GASEN)	400	10	4263.1406	94.61%	90.38%	0.0022
	SEOS-ELM (PSOSEN)	400	10	3145.8528	94.85%	90.57%	0.0019

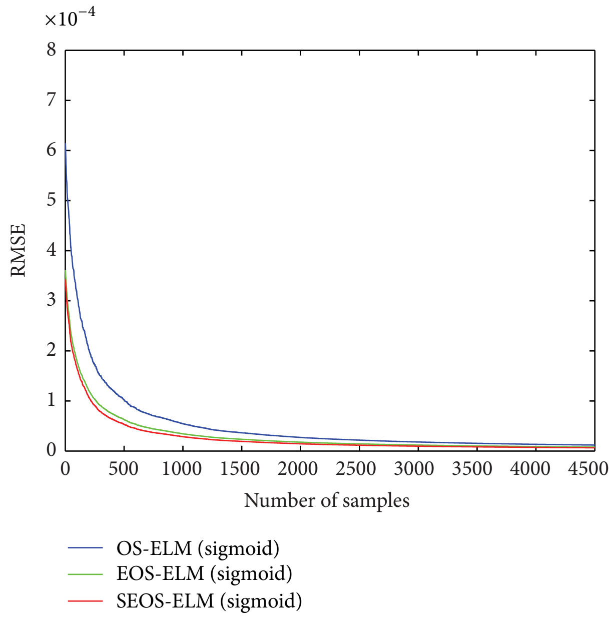


Figure 3: RMSE with respect to the number of training samples for sigmoid hidden nodes.

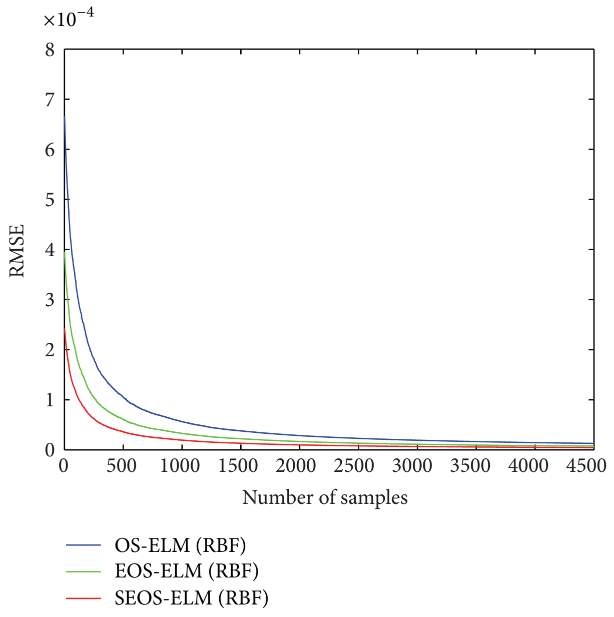


Figure 4: RMSE with respect to the number of training samples for RBF hidden nodes.

Table 3 to Table 6 also confirms the stability performance of SEOS-ELM.

6. Discussion

In the experiments, PSOSEN showed its higher accuracy than the original OS-ELM and simple ensemble of OS-ELM, which verified the feasibility of the selective ensemble method. In addition, compared with GASEN, PSOSEN showed comparable accuracy while much faster learning speed. Taking the complexity and accuracy into consideration, PSOSEN is a good choice for selective ensemble. Experiments on online version ELM have demonstrated the advantages. However, it should be noted that, as a general selective ensemble method, PSOSEN is applicable to any learning algorithms, both batch learning and online learning. So, applying PSOSEN to other learning algorithms are of interest in the future.

The experiments also showed that although ensemble learning, both simple ensemble and selective ensemble, attains higher accuracy, it is more time consuming than the original learning algorithm. In addition, selective ensemble is slower than simple ensemble. As a selective ensemble method, PSOSEN is also slower than the original learning algorithm and the simple ensemble. So, selective ensemble is a trade-off between complexity and accuracy. In the future, new selective ensemble method should be designed to further improve the speed of the algorithm.

7. Conclusion

In this paper, PSOSEN is proposed as a novel selective ensemble algorithm. Benefiting from the fast speed of PSO, PSOSEN proves to be faster than other selective ensemble algorithms. It is a general selective ensemble algorithm, which is applicable to any learning algorithms. To improve the generalization performance of the online learning algorithm, we then apply PSOSEN to OS-ELM. And in purpose of balancing the complexity and accuracy, an adaptive selective ensemble framework for OS-ELM is designed. Experiments were carried out on UCI data set. The results convincingly show that the new algorithm improves the generalization performance of OS-ELM and also keeps balance on complexity.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is partially supported by Natural Science Foundation of China (41176076, 31202036, 51379198, and 51075377).

References

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Neural Networks, 2nd edition, 2004.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[3] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[4] R. Zhang, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Multicategory classification using an extreme learning machine for microarray gene expression cancer diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 3, pp. 485–495, 2007.

[5] Y. Xu, Z. Y. Dong, J. H. Zhao, P. Zhang, and K. P. Wong, "A reliable intelligent system for real-time dynamic security assessment of power systems," *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1253–1263, 2012.

[6] K. Choi, K. A. Toh, and H. Byun, "Incremental face recognition for large-scale social network services," *Pattern Recognition*, vol. 45, no. 8, pp. 2868–2883, 2012.

[7] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[8] J. C. Yin, Z. J. Zou, F. Xu, and N. N. Wang, "Online ship roll motion prediction based on grey sequential extreme learning machine," *Neurocomputing*, vol. 129, pp. 168–174, 2014.

[9] X. Wang and M. Han, "Online sequential extreme learning machine with kernels for nonstationary time series prediction," *Neurocomputing*, vol. 145, pp. 90–97, 2014.

[10] J. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, Hangzhou, China, June 2014.

[11] N. Liu and H. Wang, "Ensemble based extreme learning machine," *IEEE Signal Processing Letters*, vol. 17, no. 8, pp. 754–757, 2010.

[12] X. Xue, M. Yao, Z. Wu, and J. Yang, "Genetic ensemble of extreme learning machine," *Neurocomputing*, vol. 129, pp. 175–184, 2014.

[13] H. J. Rong, Y. S. Ong, A. H. Tan, and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing*, vol. 72, no. 1–3, pp. 359–366, 2008.

[14] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[15] P. S. A. Krogh, "Learning with ensembles: how over-fitting can be useful," in *Proceedings of the 1995 Neural Information Processing Systems Conference*, vol. 8, p. 190, 1996.

[16] Y. Lan, Y. C. Soh, and G. B. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, no. 13–15, pp. 3391–3395, 2009.

[17] Z. H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[18] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, New York, NY, USA, 2010.

[19] C. Blake and C. J. Merz, "UCI repository of machine learning databases," Tech. Rep., University of California, 1998.

[20] J. Kennedy and W. M. Spears, "Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 78–83, Anchorage, Alaska, USA, May 1998.