01 Jan 2005

Engine Data Classification with Simultaneous Recurrent Network using a Hybrid PSO-EA Algorithm

Xindi Cai

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

Engine Data Classification with Simultaneous Recurrent Network Using a Hybrid PSO-EA Algorithm

Xindi Cai and Donald C. Wunsch II

Applied Computational Intelligence Laboratory
Dept. of Electrical and Computer Engineering
University of Missouri - Rolla
Rolla, MO 65409, USA
E-mail: cai@umr.edu, dwunsch@ece.umr.edu

Abstract — We applied an architecture which automates the design of simultaneous recurrent network (SRN) using a new evolutionary learning algorithm. This new evolutionary learning algorithm is based on a hybrid of particle swarm optimization (PSO) and evolutionary algorithm (EA). By combining the searching abilities of these two global optimization methods, the evolution of individuals is no longer restricted to be in the same generation, and better performed individuals may produce offspring to replace those with poor performance. The novel algorithm is then applied to the simultaneous recurrent network for the engine data classification. The experimental results show that our approach gives solid performance in categorizing the non-linear car engine data.

I. INTRODUCTION

The traditional gradient-based training algorithms, such as BPTT [1] and EKF [2], have been known to suffer from local minima and have heavy computation load for obtaining the derivative information. Population-based algorithms provide alternative solutions to applications where conventional gradient counterparts failed [3]. Designing an efficient and powerful population-based training algorithm is thus necessary.

Evolutionary operators like selection and mutation have been integrated into the conventional particle swarm optimization (PSO) algorithm. By applying the selection operation in PSO, the particles with best performance are copied to next generation. Therefore, PSO can always keep the best performing particles [4]. The purpose of applying mutation operation to PSO is to increase the diversity of the population and thus overcome the local minima problem [5] in the PSO algorithm. Our approach employs the PSO to enhance the elites in the evolutionary algorithm (EA). The novel algorithm is then applied to train the simultaneous recurrent network on the car engine data classification problem.

The rest of the paper is organized as follows. Section II presents the architecture of the simultaneous recurrent network. Section III describes the PSO, EA and the hybrid PSO-EA learning algorithms. In Section IV, parameter selection is presented first, and then experimental results are provided. Finally, the conclusions are given in Section V.

II. SIMULTANEOUS RECURRENT NETWORK

Simultaneous recurrent network (SRN) is an architecture used for certain function approximation problems [15], [16]. The SRN uses recurrence to approximate a *static* deterministic mapping $X(t) \rightarrow Y(t)$ (Fig. 1). This mapping is computed by iterating the following equation:

$$Y^{(n+1)}(t) = f(Y^{(n)}(t), X(t), W) \qquad (1)$$

where f is any feed-forward network or system, and Y(t) is defined as:

$$Y(t) = \lim_{n \to \infty} Y^{(n)}(t) \qquad (2)$$

The recurrence in SRN is invoked at each time t, but it is not visible from the outside of the network. In applications, the number of iterations n can be limited to a reasonably large number.
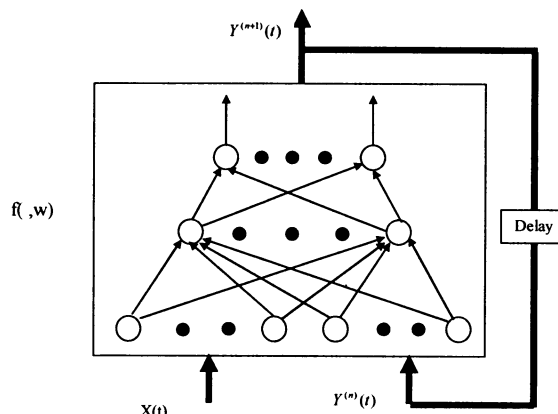

Fig. 1. Structure of the simultaneous recurrent network

For the specific engine data classification problem, the architecture of SRN is shown in Fig. 2. The simultaneous recurrence occurs when the hidden layer is copied verbatim through weight connections, which are equal to 1, and stored in the context units. The iteration number n is set to 20. The input layer has 14 neurons (each engine data has 14 elements), the hidden layer has 10 neurons, the context layer has 10 neurons, and the output layer has 4 neurons, indicating 4 different classes. Neurons between adjacent layers are fully connected, as marked by bold arrows in Fig. 2. The transfer functions of the hidden layer and the output layer are *tansig*.
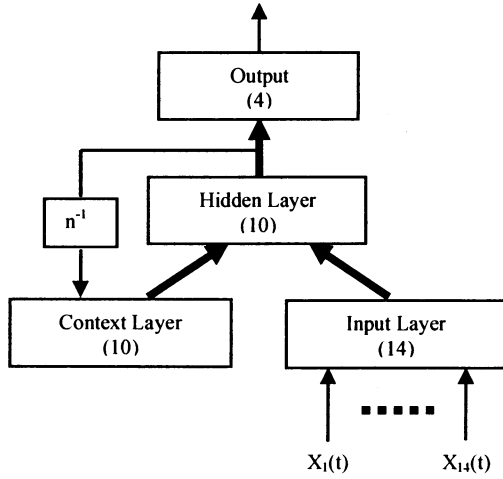

Fig. 2 Architecture of SRN, at time t, for engine data classification problem

III. HYBRID PSO-EA LEARNING ALGORITHM

A. Particle Swarm Optimization

Particle swarm optimization (PSO) is a form of evolutionary computation technique developed by Kennedy and Eberhart [6], [7]. Similar to Evolutionary Algorithms (EA), particle swarm optimization algorithm is a population based optimization tool, where the system is initialized with a population of random solutions and the algorithm searches for optima satisfying some performance index over generations. It is unlike an EA, however, in that each potential solution is also assigned a randomized velocity, and the potential solutions, called *particles*, are then "flown" through the problem space.

Each particle has a position represented by a position vector $\vec{x}_i$. A swarm of particles moves through the problem space, with the velocity of each particle represented by a vector $\vec{v}_i$. At each time step, a function f_i representing a quality measure is calculated by using $\vec{x}_i$ as input. Each particle keeps track of its own best position, which is

associated with the best fitness it has achieved so far in a vector $\vec{p}_i$. Furthermore, the best position among all the particles obtained so far in the population is kept track of as $\vec{p}_g$.

At each time step t, by using the individual best position, $\vec{p}_i(t)$, and global best position, $\vec{p}_g(t)$, a new velocity for particle i is updated by

$$\vec{v}_i(t+1) = w \times \vec{v}_i(t) + c_1\phi_1(\vec{p}_i(t) - \vec{x}_i(t)) + c_2\phi_2(\vec{p}_g(t) - \vec{x}_i(t)) \quad (1)$$

where c_1 and c_2 are positive constants, ϕ_1 and ϕ_2 are uniformly distributed random numbers in [0, 1] and w is the inertia weight. The term $\vec{v}_i$ is limited to the range $\pm \vec{v}_{max}$. If the velocity violates this limit, it is set at its proper limit. Changing velocity this way enables the particle i to search around its individual best position, $\vec{p}_i$, and global best position, $\vec{p}_g$. Based on the updated velocities, each particle changes its position according to the following:

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (2)$$

Based on (1) and (2), the population of particles tends to cluster together with each particle moving in a random direction. The computation of PSO is easy and adds only a slight computation load when it is incorporated into EA. Furthermore, the flexibility of PSO to control the balance between local and global exploration of the problem space helps to overcome premature convergence of elite strategy in EA, and also enhances searching ability.

The pseudo code for PSO is summarized as follows [8]:

(1) Initialize a population of particles with random positions and velocities of d dimensions in the problem space.

(2) For each particle, evaluate the fitness according to the given fitness function in d variables.

(3) Compare current particle's fitness with its previous fitness. If current value is better than the previous, then set $\vec{p}_i$ value equal to the current value, and the $\vec{p}_i$ location equal to the current location in d-dimensional space.

(4) Compare fitness evaluation with the population's overall previous best position. If the current value is better than $\vec{p}_g$, then reset $\vec{p}_g$ to the current particle's array index and value.

(5) Change the velocity and position of the particle according to equation (1) and (2), respectively.

(6) Repeat step (2) to (6) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations/epochs.

B. Evolutionary Algorithm

To begin the evolutionary algorithm, a population of n neural networks, K_i, i=1,.., n, defined with weights and bias for each network, was created at random. Weights and biases were generated by sampling from a uniform random distribution over [-1, 1]. Each neural network had an associated self-adaptive parameter vector σ_i, i=1,..., n, where each component corresponded to a weight or bias and served to control the step size of the search for new mutated parameters of the neural network. To be consistent with the range of initialization, the self-adaptive parameters for weights and biases were set initially to a constant in [-1, 1].

Each parent generated an offsprings strategy by varying all of the associated weights and biases. Specifically, for each parent K_i, i=1,..., n, an offspring K_i', i=1,..., n, was created by

$$\sigma_i'(j) = \sigma_i(j)\exp(\tau N_j(0,1)), \qquad j=1,...,N_w \quad (3)$$

$$w_i'(j) = w_i(j)+\sigma_i' N_j(0,1), \qquad j=1,...,N_w \quad (4)$$

where N_w is the number of weights and biases in the recurrent neural network, $\tau = 1/\sqrt{2\sqrt{N_w}}$, and $N_j(0,1)$ is a standard Gaussian random variable resampled for every j [9].

For the car engine data classification problem, a population of 40 individuals is competing for the best prediction. Each individual represents a SRN described in Section II. The number of weights and biases, i.e., N_w, in such a SRN (*size* – *14-10R-5*) is 315, and hence τ is 0.1678. Half of population with best fitness is used as parents to create offspring for next generation.

C. Hybrid of PSO and EA

The social and cognitive adaptation makes PSO focus more on the co-operation among the particles. With memory, each particle tracks the best performance in its own history, and its neighborhood throughout the entire evolution when sharing the memory. However, particles of PSO will never be removed even if they are impossible to be the best solution, which may waste the limited computational

resources. On the other hand, individuals in EA compete for survival, but the winning survivors discard the valuable searching information of the previous generation when they almost randomly pick up the search direction of their offspring for the next generation. Clearly, the advantage of one algorithm can be the remedy for the other's shortcoming. It is natural and wise for us to develop a hybrid algorithm.

Based on the complementary properties of PSO and EA, we design a novel hybrid algorithm that combines the co-operative and competitive characteristics of both PSO and EA. In another word, we apply the PSO to improve the survival individuals, and maintain the properties of competition and diversity in EA. In each generation, the hybrid algorithm selects half of the population as the winners according the fitness, and discards the rest half as losers. These elites are enhanced, sharing the information in the community and benefiting from their learning history, by standard PSO procedure. The enhanced elites then serve as parents for an EA mutation procedure to produce same amount of offspring to fill up the vacuum that the discarded individuals left. The offspring also inherit the social and cognitive information from the corresponding parents in case that they become winners in the next generation. Fig. 3 illustrates this hybrid PSO-EA method.
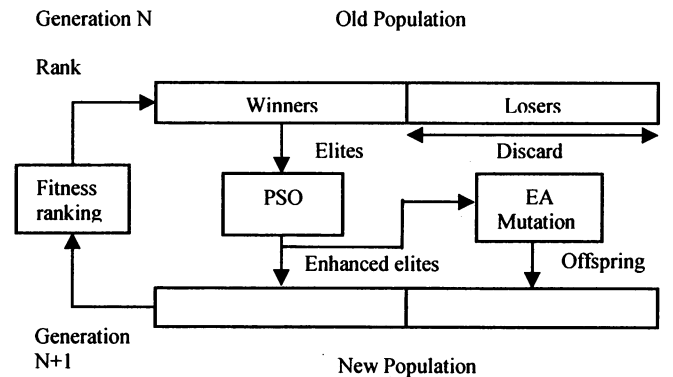

Fig. 3. Flow of hybrid PSO-EA method. Winners are enhanced by PSO and kept in the population for the new population. Offspring created by the enhanced elites of PSO as EA parents replaced the discarded losers in the old population generation to generation.

IV. RESULTS

A. PSO Parameters

The velocity change of a PSO, i.e. equation (1), consists of three parts. The first part is the momentum part, which prevents velocity to be changed abruptly. The second part is the "cognitive" part which represents private thinking of itself – learning from its own flying experience. The third part is the "social" part which represents the collaboration

among particles – learning from group best's flying experience. The balance among these three parts determines the balance of the global and local search ability, therefore the performance of a PSO.

The inertia weight w controls the balance of global and local search ability. A large w facilitates the global search while a small one enhances local search. The introduction of an inertial weight also frees the selection of maximum velocity $\vec{v}_{max}$. The $\vec{v}_{max}$ can be reduced to $\vec{x}_{max}$, the dynamic range of each variable which is easier to learn; and the PSO performance is as good or better [10], [11].

Since the search process of a PSO is nonlinear and complicated, static parameters set, if well selected, can do a good job, but much better performance can be achieved if a dynamically changing scheme for the parameters is well designed, either a linearly decreasing inertia weight [10], a nonlinearly fuzzy changing [12], or involving a random component rather than time-decreasing [13]. All intuitively assume that the PSO should favor global search ability at the beginning and local search at the end.

Based on previous work [8], the authors have chosen the following parameters for the engine data classification:

Maximum velocity, V_{max}	*2*
Inertia weight, W	*0.8*
Acceleration constants, c_1, c_2	*2, 2*
Size of swarm	*40*

B. Engine Data Classification

For the car engine data classification problem, there are four classes in both training and testing data sets, with different numbers of samples in each class. Each sample consists of 14 elements obtained from a car engine (factors in an engine diagnostic experiment). Each element represents a certain diagnostic parameter from a test run on a partially assembled engine. Various unknown combinations of parameter values indicate normal engines, or they may be indicative of different defects. It is known that the classification problem (whether the engine is normal or defective, and if so what kind of defect it is likely to have) is not linearly separable. A 14-10R-4 network with bipolar sigmoid transfer functions in the hidden and output layers is used. Neurons between different layers are fully connected.

We use batch training method, and the weights are updated based on a cumulative error function. The process can be repeated over a number of epochs.

For the convenience, the original data are normalized on [-1, 1] before training and testing.

C. Results

There are 41, 14, 13 and 10 sample vectors for each class in training set (78 in total), respectively, and 4, 3, 2 and 2 in the testing set (11 in total).

A population of 40 particles, each representing a 14-10R-4 SRN, is evolved for 3000 generations. The EA, PSO and hybrid PSO-EA algorithm are employed for the evolution. The cumulative RMSE for the best individual using the hybrid algorithm drops to 0.032 after 3000 generations, compared to those used EA and PSO down to 0.3. Fig. 4 shows the training error of the best individuals for the whole process by the EA, PSO and hybrid PSO+EA. After 3000 generations, the best SRN networks trained by EA, PSO and hybrid achieve 86%, 83% and 100% in the training set respectively. In the testing set, the SRNs trained by the above three algorithms identify 4 classes with 82%, 78% and 100% accuracy, respectively (see Table I). The predictions can be further improved by optimizing the PSO parameters as explained in [14].
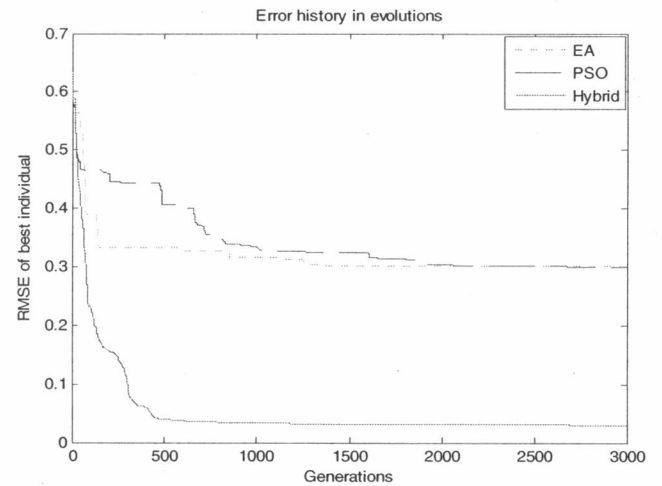

Fig. 4 Training error for the EA, PSO and hybrid PSO-EA. The errors reflect the performance of the best particle, i.e. the $\vec{p}_g$, at each generation for training data set.

TABLE I SRN ENGINE DATA CLASSIFICATION ON THE TRAINING AND TESTING SETS

	TRAINING SET				
	I	II	III	IV	ACCURACY
EA	40/41	14/14	13/13	0/10	87%
PSO	41/41	2/14	12/13	10/10	83%
HYBRID	41/41	14/14	13/13	10/10	100%
	TESTING SET				
	I	II	III	IV	ACCURACY
EA	4/4	3/3	2/2	0/2	82%
PSO	4/4	0/3	2/2	2/2	72%
HYBRID	4/4	3/3	2/2	2/2	100%

V. CONCLUSIONS

We have presented and discussed a novel algorithm, hybrid PSO-EA. We explored how it combines the search capabilities of these two global optimization methods. The purpose of applying mutation in EA to PSO is to increase the diversity of the population and the ability to have the PSO to escape the local minima.

The hybrid PSO-EA learning algorithm proved to be successful in training SRN for the car engine data classification problem. The hybrid procedure takes advantage of the complementary properties of PSO and EA, which makes it more powerful than each of the individual algorithms in enhancing the survivors of a population and generating offspring.

The results show that our approach gives a perfect classification for the linearly inseparable car engine data.

ACKNOWLEDGEMENT

REFERENCES

[1] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of IEEE*, vol. 78, no. 10, pp. 1550-1560, Oct. 1990.

[2] G. V. Puskorius, and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks," *IEEE Trans. Neural Networks*, Vol. 5 No. 2, pp. 279 -297, March 1994.

[3] X. Cai, N. Zhang, G. K. Venayagamoorthy and D. C. Wunsch II, "Time series prediction with recurrent neural networks using hybrid PSO-EA algorithm," *Proc. of INNS-IEEE International Joint Conference on Neural Networks (IJCNN)*, Vol. 2, pp 1647-1652, Budapest, Hungary, July 25-28, 2004.

[4] P. J. Angeline, "Using selection to improve particle swarm optimization," *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 84-89, May 4-9, 1998.

[5] T. Blackwell and P. Bentley, "Don't push me! Collision-avoiding swarms," *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1691-1696, May 12-17, 2002.

[6] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *IEEE International Conference on Neural Networks*, Perth, Australia, vol. 4, pp. 1942 -1948, Nov. 27-Dec. 1, 1995.

[7] J. Kennedy, R. Eberhart and Y. Shi, *Swarm Intelligence*. San Meteo, CA: Morgan Kaufmann, 2001.

[8] V. G. Gudise and G. K. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks", *Proceedings of the IEEE Swarm Intelligence Symposium*, April 24- 26, 2003, Indianapolis, Indiana, USA, pp. 110 - 117.

[9] Kumar Chellapilla and David B. Fogel, "Evolution, neural networks, games, and intelligence," *Proceedings of the IEEE, Special Issue on Computational Intelligence*, vol. 87 (9), pp. 1471-1496.

[10] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," *Proceedings of the 1998 Annual Conference on Evolutionary Computation*, March 1998.

[11] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 69-73, May 1998.

[12] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," *Proceedings of the 2001 Congress on Evolutionary Computation*, Seoul, Korea, 2001.

[13] Y. Shi and R. C. Eberhart, "Particle swarm optimization with fuzzy adaptive inertia weight," *Proceedings of the Workshop on Particle Swarm Optimization*, Indianapolis, IN, 2001.

[14] S Doctor, G K Venayagamoorthy, V G Gudise, "Optimal PSO for Collective Robotic Search Applications", *IEEE Congress on Evolutionary Computation*, June 19 – 23, 2004, Portland, OR, USA.

[15] X. Z. Pang and P. J. Werbos, "Neural Network Design for J Function Approximation in Dynamic Programming", *Math. Modelling and Scientific Computing (a Principia Scientia journal)*, Vol. 5, No. 2/3, 1996.

[16] D. C. Wunsch II, "The cellular simultaneous recurrent network adaptive critic design for the generalized maze problem has a simple closed-form solution," *Proc. Int. Joint Conf. Neural Networks*, vol. 3 Como Italy July 2000, pp. 79-82.