Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 69 (2006) 2340–2345
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Letters

Phase space learning in an autonomous dynamical neural network

Hiroshi Inazawa[a,b,*], Garrison W. Cottrell[a,c]

[a]Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093, USA
[b]Center for Education in Information Systems, Kobe Shoin Women's University, 1-2-1 Shinohara Obanoyama-cho, Nada Kobe 657-0015, Japan
[c]Institute for Neural Computation and Computer Science and Engineering Department, University of California, San Diego, La Jolla, CA 92093, USA

Received 14 September 2005; received in revised form 26 April 2006; accepted 26 April 2006
Communicated by R.W. Newcomb
Available online 9 June 2006

Abstract

In this paper, we present an improved version of the online phase-space learning algorithm of Tsung and Cottrell (1995), called ARTISTE (Autonomous Real-TIme Selection of Training Examples). The new version's advantages derive from an online adaptive learning rate that depends on the error. We demonstrate the algorithm's efficacy on two problems: learning a pair of sine waves offset by 90° and the van der Pol oscillator. The online version of the algorithm allows the system to learn as it behaves. We show that the adaptive learning rate technique gives us excellent results in the learning of the above two tasks.
Published by Elsevier B.V.

Keywords: Neural network; Learning; Temporal data; Phase space; Adaptive learning rate

1. Introduction

In previous work, Tsung and Cottrell [4] showed how multiple periodic attractors could be embedded in a recurrent network by (1) first embedding the signal into a phase space using a delayed version of the signal (delay-space embedding), followed by (2) learning of the mapping from points in phase space to the next point in phase space ($x(t) \rightarrow x(t+\Delta t)$). With this approach, multiple attractors could be easily learned in the same network. However, the technique was inherently a batch technique, meaning that attractors had to be learned offline, and the network was trained with randomly chosen points in the phase space. This is unrealistic in a physical system, where learning from different points in the phase space would correspond to physically changing the current state of the system. Hence, an online version of the algorithm, where the system learns as it behaves, is desirable. Tsung and Cottrell [3] had such an algorithm, but its learning performance was poor. Here, we develop an improved version of the online algorithm, dubbed "ARTISTE," for Autonomous Real-TIme Selection of Training Examples. In this setting, the system moves under its own dynamics, but the teaching signals are generated online in order to alter the behavior of the system towards the desired trajectory. In this paper, we use feed-forward neural networks trained with back-propagation to learn the mapping [2].

In general, real-time recurrent learning (RTRL) [7] or back propagation through time (BPTT) [2,5] have played an important role in the field of temporal data learning. However, these models have complexity issues such as the large number of variables necessary to store derivatives: the number of variables is $O(n^3)$ for RTRL, $O(n^2T)$ for BPTT, and the computational complexity is $O(n^4)$ for RTRL, $O(n^2T)$ for BPTT, where n is the number of units in a network and T is the number of time steps [6]. At present, there exist some modifications of the models to improve these problems such as "truncated BPTT" [6]. However, BPTT remains a batch method, and RTRL has high computational complexity. The advantages of an algorithm that could learn using a simple feed-forward network in an online fashion, thus using no extra space, are clear. The only algorithm like this that we are aware of is Tsung and Cottrell's [3] ARTISTE algorithm, an online version of

*Corresponding author. Center for Education in Information Systems, Kobe Shoin Women's University, 1-2-1 Shinohara Obanoyama-cho, Nada-ku Kobe 657-0015, Japan. Tel.: +81 78 882 6122; fax: +81 78 882 5032.
E-mail address: genzoh@shoin.ac.jp (H. Inazawa).

0925-2312/$ - see front matter Published by Elsevier B.V.
doi:10.1016/j.neucom.2006.04.006

their phase-space learning technique. However, this algorithm did not converge well on simple problems such as the van der Pol oscillator. Here we show that using an adaptive learning rate, and a somewhat different specification of the teaching signal, we can achieve good convergence on such problems. In this article, we study the performance of our new version of ARTISTE for two cases: a pair of sine waves offset by 90° and van der Pol oscillator. In Section 2, we describe the specification of the model: the ARTISTE algorithm, the network architecture and introduce the adaptive learning rate. The conditions of the simulation and the results are shown in Section 3. Finally, we summarize our findings and describe future work in Section 4.

2. Specification of the model

The network learns temporal data as a trajectory in phase space—namely, autonomous temporal data are mapped from a state space to a phase space. In state space, one of the dimensions is t, the time. In phase space, time is represented as movement along the trajectory.[1] To embed a signal from state space into phase space, where no crossings are allowed, one standard approach is to use a *delay space embedding*. For example, a sine wave in one dimension, represented as $x(t)$, could be embedded into a two-dimensional phase space by mapping it into $(x(t), x(t+\Delta t))$ for an appropriate choice of Δt. For example, a Δt of 360° would clearly be inappropriate, as it would be impossible to recover the dynamics of the sine wave (which are inherently two dimensional), but a delay of 90° would work well, and results in a circular trajectory of the trajectory in phase space. A second feature of a well-defined phase space is that there should be no crossings in phase space. If the trajectory crosses itself, then it is impossible to know which direction to go next from the intersection. Hence, a well-defined phase space inherently induces a *mapping* from points on the trajectory to the next point on the trajectory, if an appropriate Δt is used (note that this Δt is different from the one defined from the one used for the delay-space embedding). It is this mapping that phase-space learning, and ARTISTE, take advantage of. We now describe ARTISTE in more detail.

ARTISTE is an algorithm in which the network learns a trajectory in real time as it is moving freely during learning. The ARTISTE procedure is illustrated in Fig. 1. The basic idea is that as the network progresses, a teaching signal is generated based on its current position in the phase space. In particular, the standard approach of *teacher forcing* is not used [7]. Let us summarize the algorithm:

(1) Start the network on any point near the trajectory: $\vec{x}(t)$.
(2) The network generates an output point: $\vec{z}(t)$.

[1]A more modern terminology also uses "state space" for what we call "phase space." We retain the older terminology to avoid confusion, and to retain the name "phase space learning."

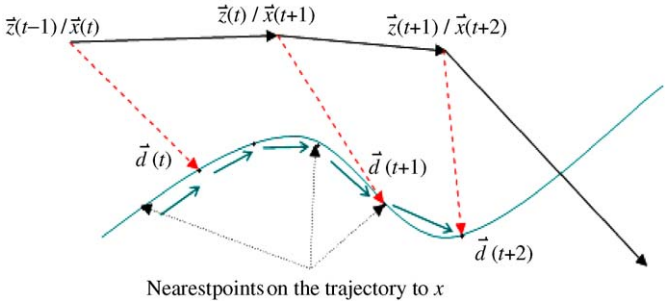

Fig. 1. Visual representation of the ARTISTE algorithm. The teaching signal is generated online. We use a discrete approximation to the trajectory. x, z, and d denote input, output, and teacher data, respectively and t denotes time, and here $\Delta t = 1$.

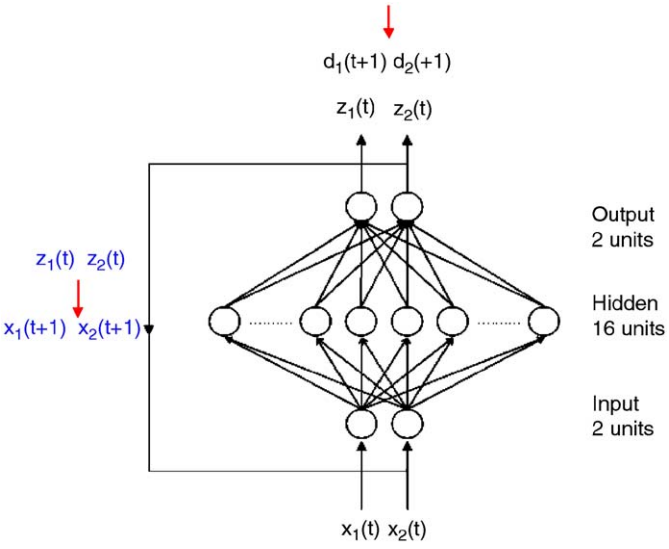

Fig. 2. Feed forward neural network with decay. $x_1(t)$ and $x_2(t)$ are an input and $z_1(t)$ and $z_2(t)$ are the outputs. $d_1(t+1)$ and $d_2(t+1)$ are the teacher signals, where $\Delta t = 1$. The input units are fully connected to the hidden units, and also the hidden units are fully connected to the output units.

(3) Calculate the nearest point on the trajectory to $\vec{x}(t)$, and the corresponding next point on the trajectory $\vec{d}(t)$.
(4) Provide the teaching signal $\vec{d}(t)$ to the network, and train using online backpropagation.
(5) Feed back $\vec{z}(t)$ as a next input point $\vec{x}(t+\Delta t)$ to input units,

where $\vec{x}$ is an input vector and $\vec{z}$ is an output vector. Note that we use a discrete approximation to the actual trajectory.

We use a three-layer feed-forward network with delays, shown in Fig. 2. The input layer and the output layer have two units because the data treated here are two dimensional. We use a 16-unit hidden layer, as this number worked well in our simulations. This parameter could be automatically chosen using cross-validation (Bishop, 1995). We use the following activation function for every

unit in the network [1]:

$$f(u) = 1.7159 \tanh\left(\frac{2}{3}u\right), \quad (1)$$

where u is the weighted sum of the inputs. The error function is defined as follows:

$$E(t) = \frac{1}{2}(\vec{d}(t) - \vec{z}(t))^2, \quad (2)$$

where $\vec{d}$ is a teacher vector. The correction of weights with momentum is defined as follows:

$$\Delta w_{ij}^L(t+1) = -\eta(E(t))\frac{\partial E(t)}{\partial w_{ij}^L(t)} + \mu \Delta w_{ij}^L(t), \quad (3)$$

where w_{ij}^L indicates the weights of the units in the output layer and the hidden layer for $L = OUT$, $HIDDEN$, respectively, $\eta(E(t))$ denotes the learning rate and μ is the coefficient of the momentum. $\eta(E(t))$ explicitly shows that the learning rate depends on the error function of (2), and we call this the "*adaptive learning rate*". It is defined by

$$\eta(E(t)) = c_1 + c_2 \ln(E(t) + \varepsilon),$$
$$\text{if } \eta(E(t)) \leqslant 0 \text{ then } \eta(E(t)) = 0, \quad (4)$$

where c_1, c_2 and μ are arbitrary positive constants that must be determined empirically. ε is a small positive constant to avoid taking the log of 0; we use an $\varepsilon = O(10^{-20})$. $\eta(E(t))$ is a monotonically increasing function of $E(t)$. Note that the value becomes negative when $E(t)$ is less than $e^{-c_2/c_1} - \varepsilon$. $\eta(E(t)) < 0$ means anti-learning. In order to avoid anti-learning, we have an additional condition in (4). We use the same network architecture for the tasks of learning a pair of sine waves offset by 90° and the van der Pol oscillator.

3. Simulation

3.1. The learning of a pair of sine waves

The first task is to learn a pair of sine waves offset by 90° (which describes the dynamics of a single sine wave delay space embedded at 90°). This describes a limit cycle attractor of a circle centered at (0, 0) with radius 1.0 in the phase space. The circle consists of 60 points because of using a discrete approximation.

In the simulation, the network starts its learning from an position $(0, 1) \pm \alpha (\alpha \leqslant 0.05)$ after initializing various network variables and learns the 60 points on the attractor, where we used 0.05 as c_1, 0.005 as c_2 and 0.05 as μ. Fig. 3 shows a successful result after 42K time steps of learning. Fig. 3a shows the result in phase space and Fig. 3b shows the state of the network over time. In the recall process, we gave 1st point on the attractor to the network as an initial point: (0, 1). The results in Fig. 3 clearly show that learning is successful.

In order to objectively measure learning performance, we use the average Euclidean distance between the network and the attractor over several cycles:

$$Gap = ||network(t) - attractor(t)||/N_t, \quad (5)$$

where $network(t)$ means the outputs of the network and $attractor(t)$ means the desired trajectory and N_t is the number of evaluation steps. If Gap has a small value, we can judge that the learning process worked well and we had good learning performance. Based on our own observations, we decided to define learning as being successful when $Gap \leqslant 0.3$. For instance, the value of Gap in Fig. 3 is 0.191, where N_t is 300, which mean 5 laps of the circle. We use this criterion for the sine wave in what follows.

Let us show the learning performance in more detail for the sine wave task. We use the following method to test the performance: we execute the learning procedure 42K times with several fixed learning rates and calculate Gap over 300 steps. We term this one trial. Note that when calculating Gap no learning occurs, and we start the network on the trajectory. The fixed learning rates we used are 0.1, 0.05, 0.01, 0.005, and 0.001. Fig. 4 shows the results of the learning performance with $Gap \leqslant 0.3$, where $\langle$Percent Success$\rangle$ indicates the percentage of the 100 trials in which the learning was successful. As shown in Fig. 4, we have a remarkable improvement of the learning

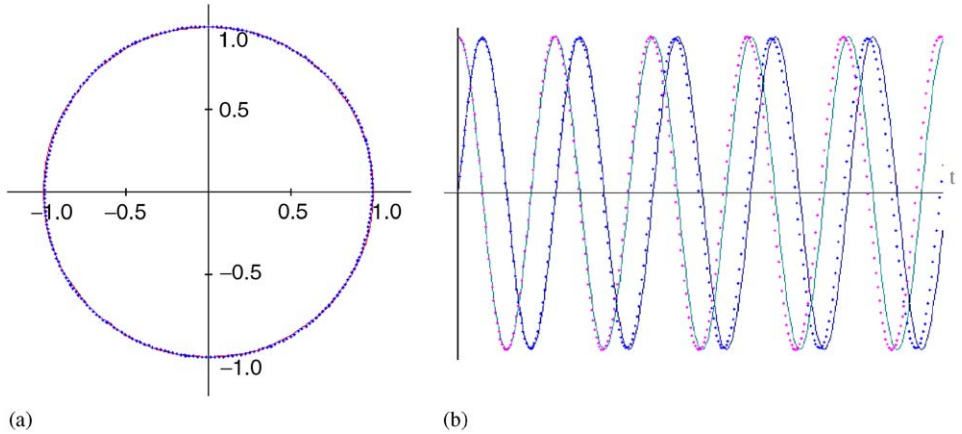

Fig. 3. A free-running a pair of sine waves offset by 90°. *Gap* (see text) is 0.191. (a) Shows the result in phase space. (b) Shows the state over time. The dots indicate the outputs of the network and the solid lines indicate the trajectory of the actual sine waves.

performance using the adaptive learning rate. Fig. 5 shows the relation between the adaptive learning rate and the learning time ($\times 60$) on a successful trial, where the adaptive learning rate is averaged by the number of points on the trajectory. We can see how the learning is working from this figure. Since the adaptive learning rate shown here is a logarithmic function of $E(t)$, we can also see how the error is changing over time from this graph. $E(t)$, varied over this time from $O(1)$ to $O(10^{-8})$. In the first stage, the learning is rather rapid, followed by an extended period from about 50 trials to about 100 trials where a relatively spiky plateau is reached, until finally there is a sudden reduction around 130 trials.

3.2. Learning the van der Pol oscillator

Here we test our algorithm on a more complicated task, learning the van der Pol oscillator. The equation of the van der Pol oscillator is described as follows:

$$\frac{dx}{dt} = y$$
$$\frac{dy}{dt} = -a(x^2 - b)y - \omega^2 x, \quad (6)$$

where we numerically solve Eq. (6) with $a = 1.0$, $b = 1.0$, $\omega = 1.0$, and $x(t = 0) = 3$, $y(t = 0) = -3$, and the time step is 0.15. We have a global periodic attractor that is shown as the solid line in Fig. 6a, where the size is scaled by 30%. In the simulation, the network starts its learning from an position $(1, -1) \pm \alpha(\alpha \leqslant 0.05)$ after initializing various network variables and learns 60 points on the attractor, where 60 points locate a closed loop without the first transient part. Fig. 6 shows a result with $Gap = 0.264$ after the learning, where the dots denote the output of the network. N_t is 300 which mean 5 laps of the closed loop. In the recall process, we gave 16th point on the attractor to the network as an initial point, where 16th point is obtained by the van der Pol equation. Fig. 6a shows the result in the state space of R^2 phase space and Fig. 6b shows the result of the state versus time t. We used the adaptive learning rate with $c_1 = 0.09$, $c_2 = 0.01$, and the learning times are 42K. According to Fig. 6, we can say that the learning is going well. However, we can see a little difference between the trajectory and the simulation points from Fig. 6a. In the state-space plot in Fig. 6b we can see that the frequency of the learned oscillation is slightly slower than the target signal, a situation that would become more pronounced over time. In addition, note that this result depends on the initial values of (6). For example, for the case of the initial values of $x(t = 0) = 3$, $y(t = 0) = 2$, there is no limit cycle, and the learning does not work well. However, for trajectories similar to Fig. 6, our results are quite similar. Fig. 7 shows the results of the learning performance, where we used 0.2, 0.1, 0.05, 0.01, 0.005, and 0.001 as the fixed learning rates. The calculation is 100 trials and the condition of the simulation is the same of those of Fig. 6.

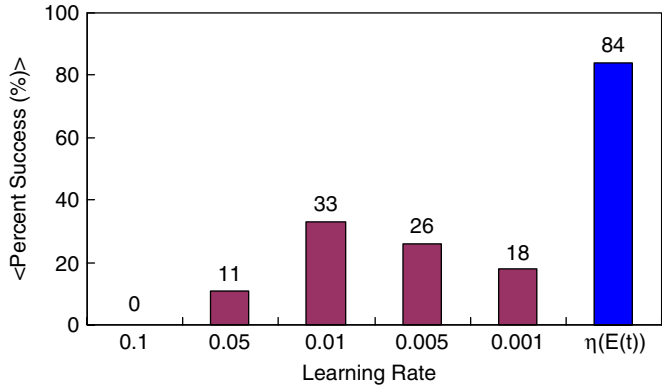

Fig. 4. The learning performance of learning a pair of sine waves offset by 90° with $Gap \leqslant 0.3$. The vertical axis denotes the averaged values, where we tried to execute 100 trials for each. The first five bars on the left-hand side denote the results for the fixed learning rates and the rightmost bar shows the result of the adaptive learning rate.

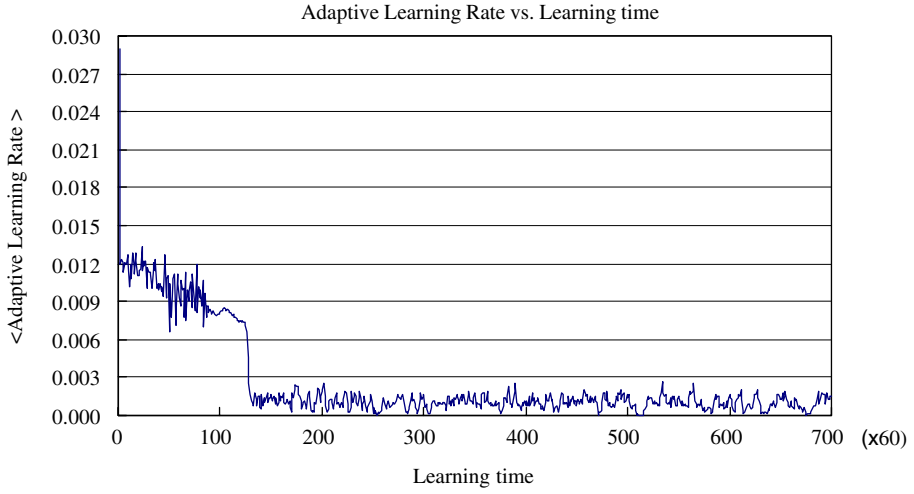

Fig. 5. An example of the relation between the adaptive learning rate and learning time. The values of the adaptive learning rate are averaged over one cycle. The adaptive learning rate decreases gradually and has a sudden reduction at about 130($\times 60$).

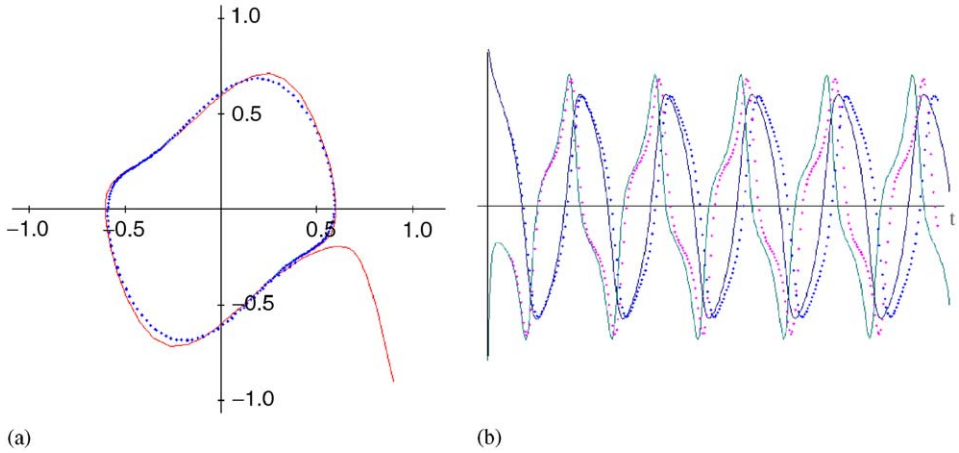

Fig. 6. A free running network that has learned the van der Pol oscillator with a *Gap* of 0.264. The dotted trajectories are the network and the solid lines are the correct trajectory. (a) Shows the result in phase space and (b) Shows the state over time.

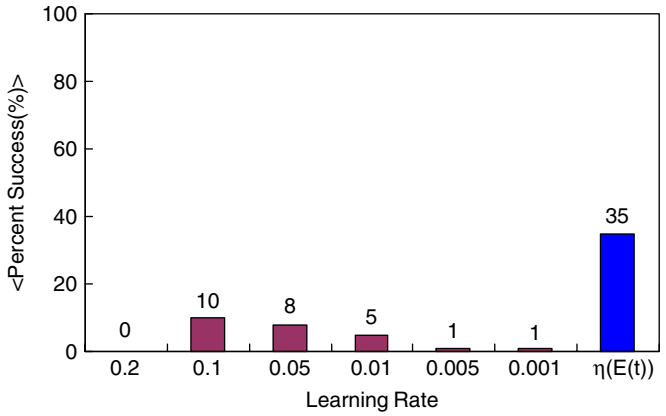

Fig. 7. The learning performance on the van der Pol oscillator. with $Gap \leqslant 0.3$. The vertical values denote the average values over 100 trials. The first six bars are the results for fixed learning rates and the rightmost bar denotes the result of the adaptive learning rate.

As we can see from Fig. 7, our adaptive learning rate procedure has improved convergence three-fold over the best fixed learning rate we tried. We can conclude that the adaptive learning rate is very effective for the task. However, the value "35" in Fig. 7 suggests that there is room to improve the learning method.

4. Conclusions and future works

We confirmed that the phase-space learning with the adaptive learning rate is very effective for learning autonomous temporal data. The feed-forward network with ordinary backpropagation and ARTISTE with the adaptive learning rate gave us remarkable improvement in the learning performance over using a fixed learning rate. We saw that the values of the weights are changed largely at the first stage of the learning process, followed by fine tuning, because the adaptive learning rate depends on the current value of the error function. We should note that one of the failure models of the learning algorithm was that occasionally the network dynamics quickly converged to the origin.

As our future tasks, it is necessary to study how to decide the values of the free parameters, such as c_1, c_2. In addition, the form of the adaptive learning rate used here was not obtained through any principled derivation. Although the form we used worked very well in the simulations, better forms of the adaptive learning rate may exist. We leave this issue for future work. Moreover, there is clearly a room for improvement in the method, as evidenced by the result on learning the van der Pol oscillator. One possibility is to allow the learning rate to actually become negative, instead of avoiding this situation, as we did in the version reported in this article. Such "anti-learning" may be useful, if carefully controlled, for avoiding local minima. In addition, we wish to try to learn other more complicated attractors by the present method, including embedding multiple attractors in the same network.

References

[1] Y. LeCun, L. Bottou, G.B. Orr, K. Müller, Efficient BackProp, in: L. Orr, K. Muller (Eds.), Neural Networks: Tricks of the Trade, Springer, Berlin, 1998.
[2] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by back-propagating errors, Nature 323 (1986) 533–536.
[3] F.-S. Tsung, G.W. Cottrell, Phase space learning for recurrent networks, Technical Report CS93-285, Department of Computer Science and Engineering, University of California, San Diego, February 1993.
[4] F.-S. Tsung, G.W. Cottrell, Phase space learning, in: D.S. Tesauro, Touretzky, T.K. Leen (Eds.), Advances in Neural Information Processing Systems, vol. 7, MIT Press, Cambridge, MA, 1994.
[5] P.J. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (10) (1990) 1550–1560.
[6] R.J. Williams, J. Peng, An efficient gradient-based algorithm for on-line training of recurrent network trajectories, Neural Comput. 2 (4) (1990) 491–501.

[7] R.J. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Comput. 1 (2) (1989) 270–280.

Hiroshi Inazawa is a Professor of Center for Education in Information Systems at Kobe Shoin Women's University. Professor Inazawa's main interest is Evolutionary Computation, in particular, learning models of Neural Network and Genetic Algorithm. In recent years, he has been studying learning of recurrent models in a field of Neural Network and function optimization in a field of Genetic Algorithm. He received his Ph.D. in 1984 from Kobe University under Professor Keizo Kobayakawa. He joined Kobe Shoin Women's University in 1988.

Garrison W. Cottrell is a Professor of Computer Science and Engineering at UC San Diego. Professor Cottrell's main interest is Cognitive Science, in particular, building working models of cognitive processes and using them to explain psychological or neurological processes. In recent years, he has focused upon face processing, including face recognition, face identification, and facial expression recognition. He has also worked in the areas of modeling psycholinguistic processes, such as language acquisition, reading, and word sense disambiguation. He received his Ph.D. in 1985 from the University of Rochester under James F. Allen (thesis title: A connectionist approach to word sense disambiguation). He then did a postdoc with David E. Rumelhart at the Institute of Cognitive Science, UCSD, until 1987, when he joined the CSE Department.