NOTE Communicated by Fernando Pineda

Faster Learning for Dynamic Recurrent Backpropagation

Yan Fang
Terrence J. Sejnowski
The Salk Institute, Computational Neurobiology Laboratory,
10010 N. Torrey Pines Road, La Jolla, CA 92037 USA

The backpropagation learning algorithm for feedforward networks (Rumelhart et al. 1986) has recently been generalized to recurrent networks (Pineda 1989). The algorithm has been further generalized by Pearlmutter (1989) to recurrent networks that produce time-dependent trajectories. The latter method requires much more training time than the feedforward or static recurrent algorithms. Furthermore, the learning can be unstable and the asymptotic accuracy unacceptable for some problems. In this note, we report a modification of the delta weight update rule that significantly improves both the performance and the speed of the original Pearlmutter learning algorithm.

Our modified updating rule, a variation on that originally proposed by Jacobs (1988), allows adaptable independent learning rates for individual parameters in the algorithm. The update rule for the ith weight, ω_i, is given by the delta-bar–delta rule:

$$\Delta\omega_i(t) = \omega_i(t) - \omega_i(t-1) = -\varepsilon_i(t)\,\delta_i(t) \tag{1.1}$$

with the change in learning rate $\varepsilon_i(t)$ on each epoch given by

$$\Delta\varepsilon_i(t) = \begin{cases} \kappa_i & \text{if } \bar{\delta}_i(t-1)\delta_i(t) > 0 \\ -\phi_i\varepsilon_i(t) & \text{if } \bar{\delta}_i(t-1)\delta_i(t) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

where κ_i are parameters for an additive increase, and ϕ_i are parameters for a multiplicative decrease in the learning rates ε_i, and

$$\delta_i(t) = \frac{\partial E(t)}{\partial \omega_i(t)} \tag{1.3}$$

where $E(t)$ is the total error for epoch t, and

$$\bar{\delta}_i(t) = (1 - \vartheta_i)\delta_i(t) + \vartheta_i\bar{\delta}_i(t-1) \tag{1.4}$$

where ϑ_i are momentum parameters.

Unlike the traditional delta rule that performs steepest descent on the local error surface, the error gradient vector $\{\delta_i(t)\}$ and the weight update vector $\{\Delta\omega_i\}$ have different directions. This learning rule assures that the learning rate ε_i will be incremented by κ_i if the error derivatives of consecutive epochs have the same sign, which generally means a smooth local error surface. On the other hand, if the error derivatives keep on changing sign, the algorithm decreases the learning rates. This scheme achieves fast parameter estimation while avoiding most cases of catastrophic divergences. In addition to learning the weights, the time constants in dynamic algorithms can also be learned by applying the same procedure.

One problem with the above adaptational method is that the learning rate increments, κ_i, were too large during the late stages of learning when fine adjustments should be made. Scaling the increments to the squared error was found to give good performance:

$$\kappa_i(t) = \lambda E(t) \tag{1.5}$$

This introduces a global parameter, λ, but one that could be broadcast to all weights in a parallel implementation.

We simulated the figure "eight" presented in Pearlmutter (1989) using the modified delta-bar–delta updating rule, the result of which is shown in Figure 1a. This is a task for which hidden units are necessary because the trajectory crosses itself. According to the learning curve in Figure 1b, the error decreased rapidly and the trajectory converged within 2000 epochs to values that were better than that reported by Pearlmutter (1989) after 20,000 epochs.[1]

We also solved the same problem using a standard conjugate gradient algorithm to update the weights (Press et al. 1988). The conjugate gradient method converged very quickly, but always to local minima (Figure 1c). It has the additional disadvantage in a parallel implementation of requiring global information for the weight updates.

We have successfully applied the above adaptational algorithm to other problems for which the original method was unstable and did not produce acceptable solutions. In most of these cases both the speed of learning and the final convergence were significantly improved (Lockery et al. 1990a,b).

[1] We replicated this result, but the original algorithm was very sensitive to the choice of parameters and initial conditions.

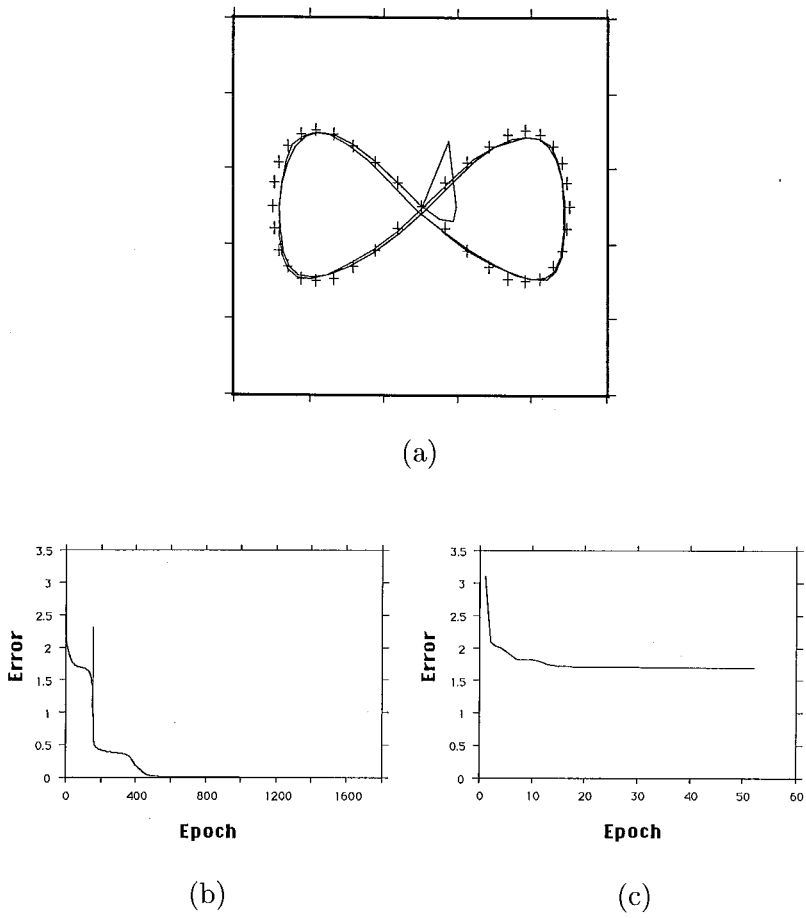

Figure 1: (a) Output from a trained network (solid) plotted against the desired figure (markers) after 1672 learning epochs. Initial weights were randomly sampled from -1.0 to 1.0 and initial time constants from 1.0 to 3.0. An upper limit of 10 and a lower limit of 0.01 were put on the range of the time constants to reduce instabilities. About 75% of the simulation runs produced stable solutions and this example had better than average performance. (b,c) Learning curve of the same situation as in (a). Parameters used: $\phi = 0.5$, $\vartheta = 0.1$, $\lambda = 0.01$, time step size $\Delta t = 0.25$. Final error $E = 0.005$. Average CPU time per epoch (on a MIPS M/120) was 0.07 sec. Notice the dramatic spiking after the first plateau. (c) Learning curve using a conjugate gradient method started with the same initial weights and time constants. Final error $E = 1.7$. Average CPU time per epoch was 2 sec.

References

Jacobs, R. A. 1988. Increased rates of convergence through learning rate adaptation. *Neural Networks* **1**(4), 295–307.

Lockery, S. R., Fang, Y., and Sejnowski, T. J. 1990a. Neural network analysis of distributed representations of sensorimotor transformations in the leech. In *Neural Information Processing Systems 1989*, D. Touretzky, ed. Morgan-Kaufmann, Los Altos.

Lockery, S. R., Fang, Y., and Sejnowski, T. J. 1990b. A dynamic neural network model of sensorimotor transformations in the leech. *Neural Comp.* **2**, 274–282.

Pearlmutter, B. A. 1989. Learning state space trajectories in recurrent neural networks. *Neural Comp.* **1**(2), 263–269.

Pineda, F. J. 1989. Generalization of back-propagation to recurrent neural networks. *Phys. Rev. Lett.* **19**(59), 2229–2232.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. 1988. *Numerical Recipes in C*, Chapter 10. Cambridge University Press, Cambridge.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986. Learning representations by backpropagating errors. *Nature* (London) **323**, 533–536.

Received 8 January 90; accepted 23 May 90.