[French, R. M. and Chater, N. (2002). Using Noise to Compute Error Surfaces in Connectionist Networks: A Novel Means of Reducing Catastrophic Forgetting. *Neural Computation, 14, 1-15.*]

Using Noise to Compute Error Surfaces in Connectionist Networks: A Novel Means of Reducing Catastrophic Forgetting

Robert M. French
Quantitative Psychology & Cognitive Science
Psychology Department
University of Liège,
4000 Liège, Belgium
rfrench@ulg.ac.be

Nick Chater
Institute for Applied Cognitive Science
Department of Psychology
University of Warwick
Coventry, CV4 7AL, UK
nick.chater@warwick.ac.uk

Abstract

In error-driven distributed feedforward networks new information typically interferes, sometimes severely, with previously learned information. We show how noise can be used to approximate the error surface of previously learned information. By combining this approximated error surface with the error surface associated with the new information to be learned, the network's retention of previously learned items can be improved and catastrophic interference significantly reduced. Further, we show that the noise-generated error surface is produced using only first-derivative information and without recourse to any explicit error information.

Introduction

Everyone forgets but, thankfully, it is typically a gradual process. Neural networks, on the other hand, and especially those that develop highly distributed representations over a single set of weights, can suffer from severe and sudden forgetting. Almost all of the early solutions to this problem, called the problem of "catastrophic forgetting," relied on learning algorithms that reduced the overlap of the network's internal representations (see French, 1999, for a review) by making these representations more sparse. This had the desired effect of reducing interference with the obvious trade-off being a decrease in the network's ability to generalize.

A significantly different approach, one which made use of noise, was developed by Robins (1995). The idea was as follows. When a network that had previously learned a set of patterns had to learn a new set of patterns, a series of random patterns (i.e., noise) was input into the network and the associated output was collected, producing a series of *pseudopatterns*. These pseudopatterns, which reflected the previously learned patterns, were then interleaved with the new patterns to be learned. This effectively decreased catastrophic forgetting of the originally learned patterns. The use of pseudopatterns will serve as the starting point for the algorithm developed in the present paper. Unlike Robins' algorithm, however, we will use pseudopatterns to directly approximate the error surface associated with the original patterns. This approximated error-surface will then be combined with the error surface associated with the new patterns and gradient descent will be performed on the combined error surface. This will be shown to significantly improve the network's performance on catastrophic forgetting.

Measures of forgetting

There are two standard measures of forgetting in connectionist models, both of which are related to standard psychological measures. The first is a simple error measurement. Suppose a first set of patterns $\{P_i : I_i \rightarrow O_i\}_{i=1}^{N}$ has been learned to criterion by a network. A new set $\{Q_i : I_i \rightarrow O_i\}_{i=1}^{M}$ is then learned to criterion. We measure the amount of network error produced by each of the patterns in the first set.

The second widely used measure of forgetting is an Ebbinghaus "savings" measure, first applied to neural networks by Hetherington & Seidenberg (1989): After learning $\{P_i\}_{i=1}^{N}$ and then $\{Q_i\}_{i=1}^{M}$, we measure the number of epochs required to retrain the network to criterion on the initial training set $\{P_i\}_{i=1}^{N}$. The faster the relearning, the less forgetting that is judged to have occurred. We will use both of these measures in the discussion that follows.

Overview of Hessian Pseudopattern Backpropagation (HPBP)

Any given set of patterns $\{P_i : I_i \rightarrow O_i\}_{i=1}^{N}$ has an associated error surface, $E(w)$, defined over the network's weights. This means that for each possible combination of values of the network's weights, there will be an overall error associated with the set of patterns (usually, the sum of the squared errors produced by each individual pattern, P_i). Learning the set of patterns $\{P_i\}_{i=1}^{N}$ is equivalent to the network's finding a minimum — call it w_0 — of this error surface.

When a new pattern, P_{new}, is presented to the system, the original error surface $E(w)$ changes to $E^{P_{new}}(w)$. (For simplicity, we will discuss only the case where a single new pattern is presented to the network, but the argument is identical for any number of new patterns.) In general, w_0, a minimum of the original error surface, $E(w)$, will no longer be a

minimum of $E^{P_{new}}(w)$. In other words, the network "forgets" the original error surface $E(w)$. What we need is some way for the network to approximate $E(w)$ in the absence of the original patterns. We could then create a new overall error surface that would reflect both $E(w)$ and $E^{P_{new}}(w)$. We do this by taking a weighted sum of the approximation of $E(w)$, which we will call $\hat{E}(w)$, and $E^{P_{new}}(w)$. Our weight change algorithm will then be gradient descent on this combined error surface. In what follows we will develop the mathematics of HPBP and will demonstrate the algorithm by means of two simple simulations on empirical data, one in which we sequentially learn two sets of patterns to criterion, the other in which the network is presented with a series of patterns, each of which is learned to criterion before the presentation of the next pattern.

Noise and the calculation of an error surface

Assume, as above, that $E(w)$ is the unique error surface defined by a set of real input-output patterns $\{P_i : I_i \rightarrow O_i\}_{i=1}^{N}$ learned by the network. The network's having learned these patterns means it has discovered a local minimum w_0 in weight-space for which $E'(w_0) = 0$ where $E'(w)$ represents the first derivative of the error function.

If the function f underlying the original set of patterns is relatively "nice" (i.e., continuous, reasonably smooth, etc.), then a set of pseudopatterns $\{y_i : \hat{I}_i \rightarrow \hat{O}_i\}_{i=1}^{M}$ whose input values are drawn from a flat random distribution will produce a reasonable approximation of f. (See French, Ans, & Rousset, 2001, for a discussion of how this approximation could be improved by additionally making use of the values of the output associated with each random input, or Ans & Rousset, 2000, for a technique that produces an "attractor" input pattern from uniform random input. In the present case, however, we simply use flat random input to produce the pseudopatterns.) Just as the original set of patterns $\{P_i\}_{i=1}^{N}$ had a unique error surface associated with it, so does the set of pseudopatterns $\{y_i\}_{i=1}^{M}$. For this latter error surface, $\hat{E}(w)$, it follows from the definition of pseudopatterns that $\hat{E}(w_0) = 0$ and $\hat{E}'(w_0) = 0$. The question is how can we produce this approximation of the original error surface in the vicinity of w_0 (assuming that the original patterns $\{P_i\}_{i=1}^{N}$ are no longer available).

We know that for the original error surface $E(w)$, $E'(w_0) = 0$. While this tells us that w_0 is a local minimum of E, it does not provide any information about the about the shape (in particular, the steepness) of $E(w)$ around w_0, which is what we want. For this we need the higher derivatives of $E(w)$, which, unlike the first derivative, *do not disappear when evaluated at* w_0. Using this steepness and the local minimum information, we reconstruct the desired approximation of the original error surface by means of a truncated Taylor series. (For other techniques using the higher-order derivatives to improve backpropagation, see, for example, Bishop (1991), Becker & Le Cun (1988), etc.)

Somewhat counter-intuitively, approximating the original error surface using noise does not require any explicit error information; noise moving through the system is sufficient for the calculation. Thus, unlike other techniques that make use of pseudopatterns which require the system to learn a mixture of pseudopatterns and new patterns (Robins, 1995; French, 1997; Ans & Rousset, 1997, 2000), here noise is simply sent through the system, and this

alone allows us to approximate the error surface around w_0. This approximated error surface, combined with the error surface of the new patterns to be learned, produces an overall error surface on which gradient descent will be performed.

The details of this calculation are as follows.

Hessian pseudopattern backpropagation (HPBP)

Assume that the network has already stored a number of patterns and has found a point w_o in weight space for which all the previously learned patterns have been learned to criterion. Further assume that we are using the standard quadratic error function:

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{N_{outputs}}(y_i^p - t_i^p)^2 \qquad (1)$$

where

P is the number of patterns,

$N_{outputs}$ is the number of output units in the network.

y_i^p is the output of the ith output node for the p^{th} pattern

t_i^p is the teacher for the ith output node for the p^{th} pattern

E being a continuous, everywhere differentiable function, it has a Taylor series expansion about w_o which we can write as follows:

$$E(w) = E(w_0) + E'(w_0)(w - w_0) + \frac{1}{2!}(w - w_0)^T H\big|_{w_0}(w - w_0) + \ldots \qquad (2)$$

where:

w is a point in weight space

w_o is the point in weight space at which the network has arrived after learning the original patterns

$E'(w_0)$ is the gradient of E evaluated at w_o and

$H\big|_{w_0}$ is the Hessian matrix of second partial derivatives of E evaluated at w_o

For values of w sufficiently close to w_0 we will assume that we can truncate the Taylor series after the second term.

Since the network is at w_0 after having learned the original patterns, this implies that w_0 is a local minimum of the error surface and, consequently, $E'(w_0)$ is 0. We can therefore write the truncated Taylor series approximation of the error surface corresponding to the originally learned set of patterns as:

$$\hat{E}(w) \approx E(w_0) + \frac{1}{2!}(w - w_0)^T H\big|_{w_0}(w - w_0) \qquad (3)$$

Now assume that a new pattern, P, is presented to the network. This pattern induces an error surface, $E^P(w)$. (Note: as mentioned above, the argument is the same for a set of new patterns.)

Let $E(w) = \alpha\,\hat{E}(w) + E^P(w)$

where the constant, α, is a weighting factor.
The standard delta rule gives:

$$\Delta w = -h \frac{\partial E}{\partial w} \quad \text{where} \quad \frac{\partial E}{\partial w} = a \frac{\partial \hat{E}(w)}{\partial w} + \frac{\partial E^P}{\partial w}$$

But from (3) we have:

$$\frac{\partial \hat{E}(w)}{\partial w} = H\big|_{w_0} (w - w_0) \tag{4}$$

The weight change rule will therefore be

$$\Delta w = -h \left[a \, H\big|_{w_0} (w - w_0) + \frac{\partial E^P}{\partial w} \right] \tag{5}$$

where
h is the learning rate and
a is the weighting factor of the prior approximated error surface.

We will now show how noise allows us to calculate $H\big|_{w_0}$

Noise and the calculation of $H\big|_{w_0}$

For each pseudopattern the teacher and the output will, by definition, be the same. In other words,

$$\forall_{y \in \Psi} \forall_n (y_n^y - t_n^y) = 0 \tag{6}$$

where
Ψ is the set of all pseudopatterns
y_n is the output from the nth output node of the network
t_n is the teacher for the nth output node of the network.

The Hessian matrix evaluated at w_o is defined as follows:

$$H\big|_{w_0} = \left[\begin{array}{ccc} \frac{\partial^2 E}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_N \partial w_N} \end{array} \right]_{w_0}$$

where w_0 is a solution for the originally learned set of patterns.

Consider the <i,j>th term of H:

$$H_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j}$$

We begin with the error function for the pseudopatterns $y_1, y_1, y_1, \ldots, y_{N_\Psi}$ where N_Ψ is the number of pseudopatterns that will be used to calculate the error surface:

$$E = \frac{1}{2} \sum_{p=1}^{N_\Psi} \sum_{n=1}^{N_{outputs}} (y_n^p - t_n^p)^2$$

where

N_Ψ is the number of pseudopatterns,

$N_{outputs}$ is the number of output units of the network

y_n^p is the output of the n^{th} output unit for the p^{th} pseudopattern,

t_n^p is the teacher for the n^{th} output unit for the p^{th} pseudopattern.

The second partial derivatives of E are calculated as follows.

$$\frac{\partial^2 E}{\partial w_i \partial w_j} = \frac{\partial}{\partial w_i} \sum_{p=1}^{N_\Psi} \sum_{n=1}^{N_{outputs}} (y_n^p - t_n^p) \frac{\partial y_n^p}{\partial w_j}$$

$$= \sum_{p=1}^{N_\Psi} \sum_{n=1}^{N_{outputs}} \left(\frac{\partial}{\partial w_i} (y_n^p - t_n^p) \frac{\partial y_n^p}{\partial w_j} + (y_n^p - t_n^p) \frac{\partial^2 y_n^p}{\partial w_i \partial w_j} \right)$$

$$= \sum_{p=1}^{N_\Psi} \sum_{n=1}^{N_{outputs}} \left(\frac{\partial y_n^p}{\partial w_i} \frac{\partial y_n^p}{\partial w_j} + (y_n^p - t_n^p) \frac{\partial^2 y_n^p}{\partial w_i \partial w_j} \right)$$

But we know from (6) that for pseudopatterns:

$$\forall_p \forall_n (y_n^p - t_n^p) = 0$$

and, therefore, the second term above is zero, giving:

$$\frac{\partial^2 E}{\partial w_i \partial w_j} = \sum_{p=1}^{N_\Psi} \sum_{n=1}^{N_{outputs}} \frac{\partial y_n^p}{\partial w_i} \frac{\partial y_n^p}{\partial w_j} \tag{7}$$

(The precise terms of the pseudopattern-induced Hessian matrix are given in Appendix 1.)

Interestingly, *only first derivative information* is required in this pseudopattern-induced Hessian, which means that the complexity of this calculation is $O(N^2)$ where N is the number of weights in the network.

In short, from (3) and (7) we conclude that noise passing through the network is sufficient to approximate the error surface for the original patterns close to w_0.

The pseudocode for the HPBP algorithm is shown below. We assume that the network has already learned a set of patterns, $P = \{P_i\}_{i=1}^N$ and is at a point local minimum w_0 in weight space. The network must then learn a new data set, $Q = \{Q_i\}_{i=1}^M$. To create the Hessian, we use R pseudopatterns.

Initialize the Hessian to 0.
Set network activation values to 0.
Hessian Loop:
    Put random input vector through the network to produce a pseudopattern;
    Use these activation levels and network weight values to create a matrix of second-derivative values to be added to the Hessian $H|_{w_0}$;
    Exit Hessian Loop after R pseudopatterns;
Training Loop: For each pattern in Q, do:
    Feedforward pass;
    Error backpropagation, changing the weights according to (5), including momentum;
    When all patterns in Q are learned to criterion, exit Training Loop;
Test errors for all patterns in P.

Simulations

In order to show that the HPBP algorithm works, we performed two simulations, the first involving catastrophic forgetting and the second involving sequential learning.

Simulation 1: Catastrophic forgetting

We created two sets of four patterns, P and Q. The two sets were intentionally designed to maximally interfere with one another (even though a network would have been able to learn all patterns in the combined set $P \cup Q$). The network was trained first on P and then on Q. Once it had learned Q to criterion, we tested it to see how well it had remembered P. An 8-32-1 network was used for both BP and HPBP networks with learning rate 0.01, momentum 0.9, Fahlman offset 0.1, and with a maximum weight-change step size of 0.075. For the HPBP network we used 100 pseudopatterns and, because we wanted to give more weight to approximated error surface associated with past learning, we set its weighting factor to 8. All results were averaged over hundred runs of the program.

Results

After learning the first set of patterns P, then Q, the standard backpropagation network produced an average error over all items in P of 0.80. (Thus, as intended, interference of the items in P by the items in Q was extremely severe.) By contrast, the HPBP network produced an average error for these previously learned items of only 038. Further, the HPBP network correctly generalized on 67.5% of the previously learned items, whereas the backpropagation network was able to generalize correctly on only 10.25% of the items in P. (See Figures 1a and 1b.) In addition, we measured the number of epochs required for both networks to relearn P. Not surprisingly, HPBP also relearned P to criterion in 42% fewer epochs than the BP network.

Although much work clearly remains to be done on this type of algorithm, we believe that these early results demonstrate that the HPBP algorithm can be very effective in reducing catastrophic interference.

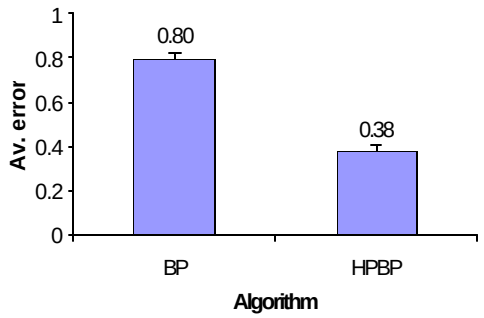

Figure 1a. Errors on the originally learned patterns in set *P* for BP and HPBP after learning set *Q*.

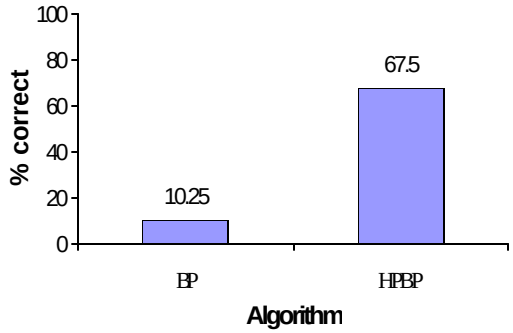

Figure 1b. Correct generalization for the originally learned items for BP and HPBP after learning *Q*.

Simulation 2: Sequential Learning

In order to further test the HPBP algorithm on a sequential learning task drawn from a real-world database, we selected the 1984 Congressional Voting Records database from the UCI repository (Murphy & Aha, 1992). Twenty members of Congress (10 Republicans, 10 Democrats, each "pattern" being defined a yes-no voting record on 16 issues associated with a party affiliation) were chosen randomly from the database and were learned sequentially by the network (i.e., each pattern was learned to criterion before the next pattern was presented to it). The network was then tested with both standard measures of forgetting on each of the twenty patterns.

Both BP and HPBP algorithms used a 16-3-1 feedforward backpropagation network was used, with a learning rate of 0.01, momentum of 0.9, a Fahlman offset of 0.1, with a maximum weight step of 0.09. For the HPBP network, 25 pseudopatterns were generated each time a new pattern was to be learned. The weighting parameter associated with the approximation of the original error surface was set to 3.

For each new pattern that was sequentially learned, 25 pseudopatterns were generated to calculate the Hessian and, thereby, to produce the approximation of the prior error surface. Specifically, the network learned the first pattern, P_1, until the difference between target and output for the pattern was below 0.2. Then 25 pseudopatterns were generated and the associated error surface was produced. The second pattern, P_2, was then presented to the network. The new error surface induced by P_2 was then combined with the previously approximated error surface and gradient descent was performed on this combined surface until the network had learned P_2. Then 25 new pseudopatterns were generated to produce an approximation of this error surface. P_3 was then presented to the network, etc.

Results

First, we considered the extent to which the addition of the approximated error surface made the initial learning more difficult. Second, once the network had sequentially learned all twenty patterns, we measured the error for each of the previously learned patterns (i.e., the error measure of forgetting). Finally, examined how difficult it was to relearn the original patterns (i.e., the savings measure of forgetting described above).

All of the data reported was averaged over 100 runs of each algorithm. The order of presentation of the patterns and the initial weights of the networks were randomized at the beginning of each run.

Original learning

Figure 2 shows that, on average, it is more difficult for HPBP to learn the first few patterns. Presumably, this is because, before any learning of the patterns has occurred, $\widehat{E}(w)$ defines an error surface that is quite unlike the error surface associated with that of any of the twenty patterns to be learned (because the network weights are initialized randomly). However, the average number of epochs required for learning the items with HPBP soon converges to that of BP.

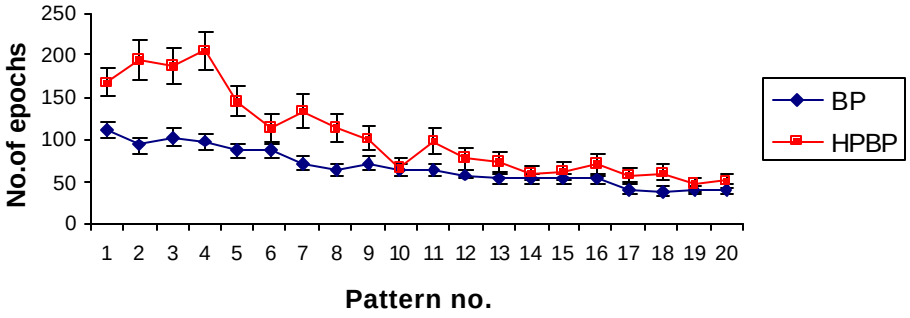

Figure 2. Original learning of the 20 items. It is somewhat harder for the Hessian pseudopattern network to learn the first few patterns, given the "inertial" effect of $\widehat{E}(w)$. Standard error bars show the evolution of the variance for both algorithms.

Error measure of forgetting

Finally, we computed average and median error scores for both HPBP and BP for each item after each network had sequentially learned all 20 items. The influence of the noise-computed error-surface in reducing these errors can be seen in Figures 3 and 4. All results were averaged over 100 runs. Standard error bars are shown for the values produced by each algorithm.

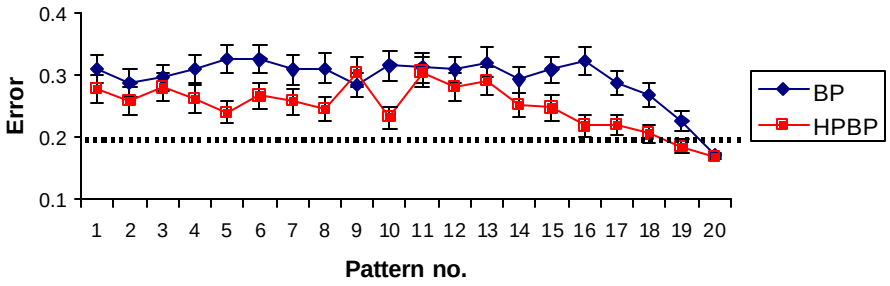

Figure 3. Average error measures for the 20 sequentially learned patterns. The average overall error for HPBP is 0.05 better than for standard BP. The learning criterion is 0.2.

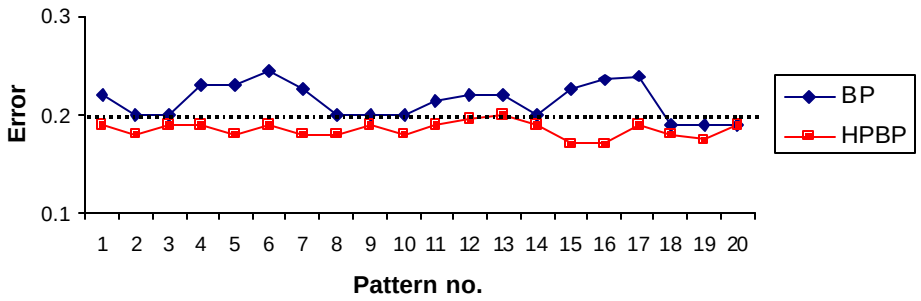

Figure 4. Median errors for BP and HPBP for all items after sequential learning of the original patterns. For HPBP all 20 patterns remain at or below the 0.2 learning criterion; only 9 of the 20 items are at or below criterion for BP.

Savings measure of forgetting

The most striking improvement with respect to standard BP can be seen in the average re-learning time for each of the individual items (Figures 5 and 6). First, all twenty patterns are learned to criterion one after the other by the network. After the 20th pattern has been learned, the network's weights, w_{final}, are saved. Then each of the first 19 items is tested to see how many epochs the network requires to relearn it. Specifically, the first pattern in the list is given to the network and it relearns that pattern to criterion, the number of epochs required for re-learning being recorded. The network weights are then reset to w_{final}. The second pattern in the list is then re-learned to criterion by the network and the number of epochs required to do so is recorde d, and so on.

Overall, relearning is about 45% faster for the patterns learned by the HPBP network compared to BP (30.8 epochs for BP vs. 16.9 epochs for HPBP). In short, with HPBP, there is still forgetting, but it is "shallower" forgetting, (i.e., re-learning of the previously learned patterns is significantly easier for HPBP than BP).

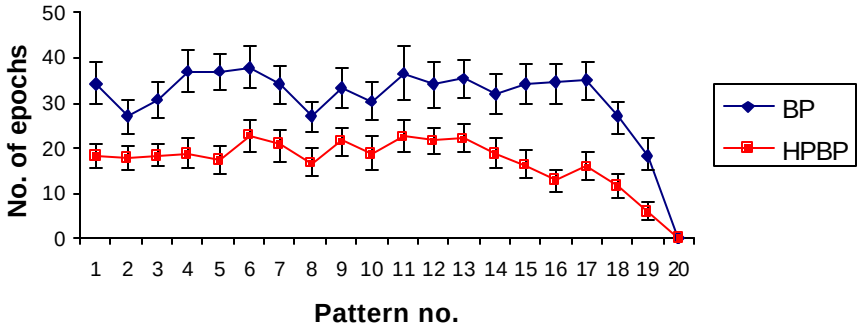

Figure 5. Average relearning times for all 20 patterns after the network has learned them sequentially. Standard error bars are shown for the means for each pattern in the sequence.

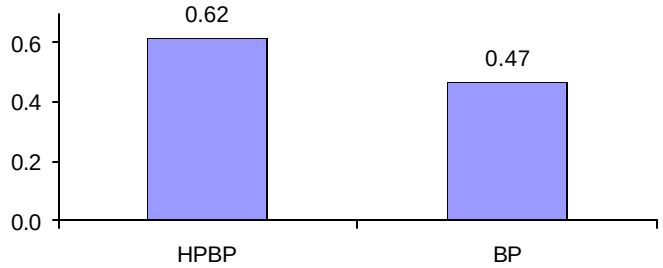

Figure 6. The proportion of patterns requiring no relearning is significantly higher for HPBP than for BP.

Discussion

It is clear that HPBP shows improved forgetting performance compared to standard backpropagation. However, there are a number of issues concerning the generality and computational complexity of this algorithm that must be addressed. We will briefly discuss the quality of the approximation of $E(w)$ by $\hat{E}(w)$ and whether or not HPBP would scale to larger networks.

The quality of the approximation of $E(w)$ by $\hat{E}(w)$ is largely dependent on three factors: i) the form of the original error surface, ii) the choice of pseudopatterns, and iii) the divergence from w_0. If we assume that the error surface close to w_0 is approximately quadratic, then in this neighborhood we would need only as many points as there are degrees of freedom in the weights to determine the shape of the bowl. Since we assume a quadratic approximation, as the network grows in size, the number of pseudopatterns required to determine the approximation should scale with the number of weights. Finally, the further we move from w_0, the less accurate the Taylor series expansion of E will be.

HPBP, as we have said, requires only first-order derivative information and thus is has a complexity of $O(n^2)$ where n is the number of weights. Further, there exists an $O(n)$ approximation of the Hessian (Le Cun, 1987; Becker & Le Cun, 1988) that could also be produced by pseudopatterns. This would ensure that scaling would be linear in the number of weights. Further, this approximation of the Hessian has only diagonal elements and, as a result, weight changes in the HPBP algorithm would require only local information. Explorations of the scaling performance of the network with the diagonal approximation to the Hessian are needed.

Conclusions

We have presented a connectionist learning algorithm that significantly improves network forgetting performance by turning noise to its advantage. A number of authors have shown that a certain amount of noise can enhance the performance of various systems in a wide range of contexts. For example, Linsker (1988) has shown elementary perceptual detectors can emerge from noise. Numerous authors (e.g., Collins, Chow & Imhoff, 1995; Grossberg & Grunewald, 1997; Sougné, 1998; etc.) have shown how the addition of noise to neural networks can enhance weak signal detection. In a neurobiological setting, Douglass, Wilkens, Pantazelou & Moss (1993), Bezrukov & Vodyanoy (1995) and others have shown that optimal noise intensity in biological neurons can enhance signal detection.

In this paper, we have shown how noise can be harnessed to improve memory performance in feedforward backpropagation networks. We believe that this work, and the

work by others on related problems, represent the tip of the iceberg in the exploration of how noise can be turned from a problem into a performance-enhancing advantage.

Acknowledgments

This work has been supported in part by a grant from the European Commission HPRN-CT-1999-00065. The authors would like to thank Anthony Robins and Dave Noelle for their discussions of the ideas of this work. Particular thanks to Gary Cottrell and an anonymous reviewer whose insightful comments contributed significantly to the quality of this paper.

References

Ans, B. & Rousset, S. (1997). Avoiding catastrophic forgetting by coupling two reverberating neural networks. *Academie des Sciences, Sciences de la vie*, 320, 989 - 997

Ans, B. & Rousset, S. (2000). Neural Networks with a Self-Refreshing Memory: Knowledge Transfer in Sequential Learning Tasks without Catastrophic Forgetting. *Connection Sciences, 12*, 1, *1-19*

Becker, S. & Le Cun, Y. (1988) Improving the convergence of back-propagation learning with second order methods. In D. S. Touretzky, G. E. Hinton, and T. J. Sejnowski, (eds.), *Proceedings of the 1988 Connectionist Models Summer School*, San Mateo, CA, 1988. Morgan Kauffman. pp. 29–37.

Bezrukov, S. & Vodyanoy, I. (1995). Noise induced enhancement of signal transduction across voltage-dependent ion channels. *Nature, 378*, 362-364.

Bishop, C. (1991). A fast procedure for retraining the multilayer perceptron. *International Journal of Neural Systems, 2*(3), 229-236.

Collins, J., Chow, C. & Imhoff, T. (1995). Stochastic resonance without tuning. *Nature, 376*, 236-238.

Douglass, J., Wilkens, L., Pantazelou, E., & Moss, F. (1993). Noise enhancement of information transfer in crayfish mechanoreceptors by stochastic resonance. *Nature, 365*, 337-340.

French, R. M. (1997) Pseudo-recurrent connectionist networks: An approach to the "sensitivity–stability" dilemma. *Connection Science, 9*, 353-379.

French, R. M. (1999). Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Sciences, 3*(4), 128-135.

French, R. M., Ans, B., & Rousset, S. (2001). Pseudopatterns and dual-network memory models: Advantages and shortcomings. In *Connectionist Models of Learning, Development and Evolution*. R. French, J. Sougné (eds.). London: Springer-Verlag.

Grossberg, S. & Grunewald, A. (1997). Cortical synchronization and perceptual framing. *Journal of Cognitive Neuroscience, 9*, 117-132.

Hetherington, P. & Seidenberg, M., (1989), Is there "catastrophic interference" in connectionist networks?, In *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, 26-33, Hillsdale, NJ: LEA

Le Cun, Y. (1987). *Modèles connexionnistes de l'apprentissage*. Unpublished Ph.D thesis. Université Pierre et Marie Curie, Paris, France.

Linsker, R. (1988). Self-organization in a perceptual network. *Computer*. March, 1988. 105-117.

Murphy, P. & Aha, D. (1992). UCI repository of machine learning databases. Maintained at the Dept. of Information and Computer Science, U. C. Irvine, CA

Robins, A. (1995). Catastrophic forgetting, rehearsal, and pseudorehearsal. *Connection Science, 7*, 123 - 146.

Sougné, J. (1998). Period doubling as a means of representing multiply-instantiated entities. *Proceedings of the 20th Annual Conference of the Cognitive Science Society*. NJ:LEA. 1007-1012.

Appendix 1

In order to approximate the error surface associated with the originally learned patterns by means of pseudopatterns, it is sufficient to calculate the terms of the Hessian matrix. The Hessian matrix evaluated at w_o is defined as follows:

$$H\big|_{w_0} = \left[\begin{array}{ccc} \frac{\partial^2 E}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_N \partial w_N} \end{array} \right]_{w_0}$$

where $\vec{w}_0$ is the vector of weights $(w_1^0, w_2^0, w_3^0, \cdots, w_N^0)$ which was a solution for the originally learned set of patterns.

We have shown in (7) that for any two weights, w_i and w_j in a network with $N_{outputs}$ output nodes and for $N_?$ pseudopatterns, the <i,j>th term of the Hessian matrix is:

$$\frac{\partial^2 E}{\partial w_i \partial w_j} = \sum_{p=1}^{N_\Psi} \sum_{n=1}^{N_{outputs}} \frac{\partial y_n^p}{\partial w_i} \frac{\partial y_n^p}{\partial w_j}$$

For each pseudopattern and each output node (with output y) and for all pairs of weights w_i and w_j, we calculate: $\frac{\partial y}{\partial w_i}\frac{\partial y}{\partial w_j}$. Each term of the Hessian matrix will be the sum over all output nodes and over all pseudopatterns. The notation conventions are as follows:

y_a: the output from node a

y_a': the first derivative of the squashing function, evaluated at y_a. For the standard squashing function: $y = \frac{1}{1+e^{-x}}$ we have: $y_a' = y_a(1-y_a)$

N_{OUTPUT}: number of output nodes

w_{ab}, w_{cd} : the weight from node b to node a, and from node d to node c

There are three cases of pairs of weights to consider — namely:

Case I: when $w_{ab}, w_{cd} \in$ Hidden-Output layer

$$\frac{\partial^2 E}{\partial w_{ab} \partial w_{cd}} = \begin{cases} (y_a' y_b)(y_c' y_d) = (y_a')^2 y_b y_d & \text{if } a = c \\ 0 & \text{if } a \neq c \end{cases}$$

Case II: when $w_{ab} \in$ Input-Hidden layer and $w_{cd} \in$ Hidden-Output layer

$$\frac{\partial^2 E}{\partial w_{ab} \partial w_{cd}} = (y_a' y_b)(y_c' y_d)(y_a' w_{ac}) = (y_a')^2 y_c' y_b y_d w_{ac}$$

Case III: when $w_{ab}, w_{cd} \in$ Input-Hidden layer

$$\frac{\partial^2 E}{\partial w_{ab} \partial w_{cd}} = (y_a' y_b)(y_c' y_d) \sum_{i=1}^{N_{OUTPUT}} (y_i' w_{ia})(y_i' w_{ic}) = y_a' y_c' y_b y_d \sum_{i=1}^{N_{OUTPUT}} (y_i')^2 w_{ia} w_{ic}$$