Local learning algorithms for sequential learning tasks in neural networks

Anthony Robins[1] and Marcus Frean[2]

[1]Computer Science, The University of Otago, PO Box 56, Dunedin, New Zealand
coscavr@otago.ac.nz
[2]Electrical and Computer Engineering, University of Queensland, Brisbane 4072, Australia
marcusf@elec.uq.edu.au

Liaison person is Anthony Robins.

Abstract

In this paper we explore the concept of sequential learning and the efficacy of global and local neural network learning algorithms on a sequential learning task. Pseudorehearsal, a method developed by Robins (1995) to solve the catastrophic forgetting problem which arises from the excessive plasticity of neural networks, is significantly more effective than other local learning algorithms for the sequential task. We further consider the concept of local learning and suggest that pseudorehearsal is so effective because it works directly at the level of the learned function, and not indirectly on the representation of the function within the network. We also briefly explore the effect of local learning on generalisation within the task.

1. Introduction

Most artificial neural network (ANN) learning algorithms depend on "concurrent" learning, i.e. the whole population of training items is presented and trained as a single, complete entity. Training is then finished and no further information is learned by the network. In this paper we explore "sequential" learning, where training items can be learned at any time (in the limiting case items can be presented one at a time).

The task of sequential learning highlights the "stability / plasticity dilemma" (Grossberg, 1987). Ideally the representations developed by a learning system should be stable enough to preserve important information over time, but plastic enough to incorporate new information when necessary. The use of variable connection weights as a medium for encoding information leads most ANNs to err on the side of excessive plasticity – new learning changes the weights and thus disrupts any old information (items previously learned by the network). Grossberg suggests the analogy of a human trained to recognise the word "cat", and subsequently to recognise the word "table", being then unable to recognise "cat". This effect has been identified in many guises in the ANN literature under headings such as catastrophic forgetting, catastrophic interference, or the serial learning problem. It is this underlying problem that limits most ANN learning algorithms to concurrent learning, items cannot be learned in a sequence because later items will disrupt or eliminate earlier items.

To be restricted to concurrent learning is undesirable in practical terms, making it very difficult to deal with inherently sequential tasks. It is also modify or extend any given ANN application without completely retraining the network (compared with a traditional rule based system where information or rules can easily be added to or removed from the system). Concurrent learning is also a highly implausible constraint for ANN based cognitive models, where so much of human learning is clearly sequential or incremental in nature.

The pseudorehearsal method proposed by Robins (1995) provides a practical and interesting solution to the catastrophic forgetting problem, and therefore allows networks to learn sequentially. The method works by restricting changes in the function learned by the network to be *local* to the new item being learned. In this paper we illustrate this localisation of learning and the use of pseudorehearsal for sequential learning. We then explore the application of other local learning algorithms to the same sequential learning task, unsuccessfully in each case. We explore the concept of local learning in more detail, and note that only some kinds of localisation are effective for sequential learning. We then discuss the impact of local learning on generalisation.

2. Catastrophic forgetting and the pseudorehearsal solution

2.1. Catastrophic forgetting

A number of recent studies have used back-propagation type networks to highlight the problem of catastrophic forgetting and explore various issues (McCloskey & Cohen, 1989; Hetherington & Seidenberg, 1989; Ratcliff, 1990; Murre, 1992; French, 1992, 1994, 1997; Lewandowsky & Li, 1995; Sharkey & Sharkey, 1995; Robins, 1995, 1996; Frean & Robins, 1998). Typical illustrations of the problem use a back-propagation network to learn a population of items (input / output vector pairs) in the usual way (Rumelhart *et al.,* 1986). Subsequently a new item or population of items is learned. The effect of this new learning on the original population can be illustrated by re-testing the original population. The error of the old population of items typically increases "catastrophically" after learning even a single new item.

Such catastrophic increases in error are shown in the simulations in Section 3 below. In this paper (following Frean & Robins (1998)) we will initially present the catastrophic forgetting effect in a different way, focusing directly on the the function learned by the network. To do this we use a simple architecture consisting of a single input unit, 20 hidden units, and a single output unit (with a learning constant of 0.05, a momentum of 0.9, and an error criterion of 0.001). The initial population consists of 6 items (input value / output value pairs), which can be plotted as data points in two dimensions. After training we can plot the function learned by the network by systematically sampling the space of possible input values. As expected this function passes through the population data points, see Figure 1(a)[1]. To illustrate catastrophic forgetting we now train the network on a single new item. Figure 1(b) shows the resulting function, which correctly fits the new item data point. Notice however that this function no longer fits the initial data points at all (the original population inputs will not generate the correct outputs). After learning just a single new item catastrophic forgetting has occurred.

– Figure 1 roughly here -

2.2. The pseudorehearsal solution: localisation of changes

One solution to the catastrophic forgetting problem is to rehearse (relearn) old items along with the new items during any new learning. At its simplest, the new items can just be added to the old item population and the whole network retrained on the combined

[1]This network architecture and training population have been chosen so as to reduce the learning task to its simplest possible conditions and allow us to visualise the function learned by the network as a two dimensional graph. We intend this as an approximation of typical ANN learning tasks, where in general a network with a large number of free parameters (weights) is used to learn a smaller number of training items that define a function over some (multi-dimensional) input space.

population (see for example McClelland, McNaughton & O'Rielly (1995)). A range of other rehearsal based solutions have been explored by Hetherington & Seidenberg (1989), Ratcliff (1990), Murre (1992), and Robins (1995). Note that by retaining and old items and combining them with new items to be learned, rehearsal effectively deals with a sequential presentation of inputs using a concurrent learning algorithm.

Rehearsal is an effective solution as long as the old items are actually available for relearning. Some tasks, however, are inherently sequential. It may be the case that the old items have been lost, or it is not practical for some reason to store them. Sharkey and Sharkey note, for example, that:

> "the interference [catastrophic forgetting] problem is [...] general and should be of concern to all those involved in developing applications in which the training data only become available piecemeal over an extended period of time. For example, in on-line learning of control processes, such as found in robotics or manufacturing, it may not be practical to maintain all of the training data in memory and retrain each time a novel aspect of the data is encountered." Sharkey & Sharkey (1995, p 302).

In any case, retaining old items for rehearsal in the network seems somewhat artificial, as it requires that they be available on demand from some other source, which would seem to make the network itself redundant!

These considerations lead to the development of pseudorehearsal (Robins, 1995) as a solution to catastrophic forgetting. This mechanism is similar to rehearsal, but it does not require the storage and access of old information. It is based on the use in the rehearsal process of artificially constructed populations of "pseudoitems" instead of the previously learned items themselves. A pseudoitem is constructed by generating a new input vector at random (setting input unit values randomly in the range 0 to 1), and passing it forward through the network in the standard way. Whatever output vector this input generates becomes the associated target output. For a given network we can construct a population of pseudoitems (input / output pairs) of any size in this way. Pseudorehearsal is the use in a rehearsal process of a population of pseudoitems instead of the actual previously learned items themselves.

We can illustrate the effects of pseudorehearsal in the task described in Figure 1. After training on the old population data points (i.e. starting from the network shown in Figure 1(a)) a population of 20 pseudoitems is constructed as described above. The new item is then learned not alone, but in a buffer which contains (in this case) ten pseudoitems.

Pseudoitems to be included in the buffer are selected from the population of pseudoitems at random for each epoch of training (replacing the ten pseudoitems used in the previous epoch)[2]. Training continues until the single new item, which is always in the buffer, has reached the error criterion. Figure 1(c) shows the kinds of function which result from pseudorehearsal in this task. These functions not only fit the new data point, but also reasonably fit the original data points. Pseudorehearsal has eliminated the catastrophic forgetting that occurs in normal learning (Figure 1(b)).

In other words, rehearsal relearns the old population data points to preserve the shape of the function during subsequent learning, while pseudorehearsal samples the function at random places to construct new data points which are then relearned in the same way to preserve the original function. The pseudoitems "approximate" the old population and "map out" the function learned by the network. Relearning the pseudoitems restricts any change in the function to be local to the area of the new item(s) being learned.

There are many ways to further improve the performance of pseudorehearsal. Pseudoitems could be trained to criterion like the new item. Up to a point the more pseudoitems in the population, and the more pseudoitems used in the training buffer, the better. A large number of pseudoitems both samples the structure of the function in more detail, and also leads to greater "inertia" in the sense that the function is unwilling to change anywhere except where it is being "pulled" by the error associated with the new item. The method used here of rotating a number of pseudoitems through the rehearsal buffer also has some useful properties. Without this rotation a buffer containing a fixed set of pseudoitems would use (for each new item) fewer pseudoitems more frequently, and would thus be prone to over-fitting these pseudoitems and to conflicts when pseudoitems fall very close to the new item.

The effectiveness of pseudorehearsal at reducing catastrophic forgetting has been proven using a range of populations including: randomly constructed autoassociative and hetroassociative data sets (Robins, 1995; Ans & Rousset, 1997); the Iris data set (Robins, 1996); a classification task using the Mushroom data set (French, 1997); a structured "task domain" (Silver & Mercer, 1998), and an alphanumeric character set using a Hopfield type network (Robins & McCallum, 1998).[3]

It is important to note that pseudorehearsal does *not* preserve the weights of a network, these can change dramatically during learning. The method works directly at the level of the function embodied by the network. The old population function is preserved, and changes necessary to accommodate the new item are restricted to being *local* to the region of the new item. This localisation of changes to the function is the essence of the

[2]This is the "sweep" pseudorehearsal regime described in Robins (1995, 1996).

[3]The Iris and Mushroom data sets are taken from Murphy & Aha (1994).

pseudorehearsal method. These and other issues are pursued in more detail in Robins (1995) and Frean & Robins (1997, 1998).

3 Local learning algorithms for sequential learning

In providing a solution to the catastrophic forgetting problem pseudorehearsal allows information (new items) to be sequentially added to the network (Robins, 1995; Robins, 1996; Ans & Rousset, 1997; Silver & Mercer, 1998). As the essence of the method is to localise changes to the function learned by the network, we decided to explore the efficacy of other local learning algorithms for sequential learning.

By sequential learning we mean that items to be learned at a given time are to be learned in isolation, i.e. with no external access to other items in the population (either previously learned or subsequently to be learned) – we call this "strong" sequential learning. In comparison by "weak" sequential learning we mean that items to be learned at a given time are not learned in isolation but concurrently with other items in the population (as is the case using the rehearsal methods described above). In this paper we are interested in strong sequential learning, and we will assume the limiting case of presenting items to be learned individually, one at a time. (Note that sequential learning should not be confused with the learning of sequences, a task where the desired behaviour of the network for a given input is to output a sequence of states).

3.1. A sequential learning task

The task which forms the basis of all simulations reported below uses the Iris data set (Murphy & Aha, 1994) consisting of 150 items divided into three classes (distinct species of iris) of 50 items each. Each item consists of four real valued measurements of an iris (such as petal length). Unless otherwise noted we use a 4:3:4 autoassociative back-propagation network with a learning constant of 0.05 and a momentum constant of 0.9, and an error criterion of 0.01. All results reported were averaged over 20 individual replications of the simulation (using different populations for each replication).

The learning task is strongly sequential, presenting items to be learned individually, one at a time. We use 30 individual items from the Iris population which are to be learned sequentially by the network. The first 15 items (Population A) are drawn from one species, and the second 15 items (Population B) are drawn from a second species. As the items within each population are drawn from the same underlying distribution we do not expect catastrophic interference to occur between them (Robins, 1996). As the two populations are drawn from different underlying distributions, however, we expect interference to occur as we move from learning Population A to learning Population B (the

transition from the 15th to the 16th item).

3.2. Global vs. local learning

Most neural network learning algorithms, such as the standard back-propagation algorithm, are "global". The input space is not partitioned in any way. Any changes as a result of learning affect the whole network and consequently the whole output function. This can be seen in Figure 1 (a) and (b) where learning a single new item has changed the output of the network over the whole input space.

As described by Bottu & Vapnik (1992) a learning algorithm is "local" if it can adjust its capacity (ability to learn items) independently in different regions of the input pattern space. They outline two ways in which such localisation can be achieved:

- by manipulating the training set so that for a given testing pattern the network is trained on inputs in a small neighbourhood around the testing pattern, and
- by setting the structure of the network so that each parameter affects the capacity of the system only in a small neighbourhood.

Bottu & Vapnik show that the *k*-nearest-neighbour (kNN) algorithm (Duda & Hart, 1973) is a particular case of the first approach, and that radial basis function (RBF) networks (Moody & Darken, 1989) are an example of the second. They go on to describe their own local algorithm based on kNN which achieves very good error and rejection rates on a digit recognition task.

3.3. Global learning for sequential tasks

Global algorithms are clearly inappropriate for sequential learning tasks. In the sequential task described above a standard back-propagation network can be used to learn the 30 items individually one after another – this is the "BP (seq)" condition shown in Figure 2. As expected the network is able to learn the first 15 items (drawn from Population A) without interference, but when the 16th item (drawn from Population B) is learned the average error of the items learned so far shoots up. The latest (Population B) item is learned correctly but as a consequence of this learning the previously learned fifteen items (from Population A) have a high error and are effectively eliminated – the catastrophic forgetting effect. As further Population B items are learned the average error of the previously learned items declines (Population B items have low error but the error of the 15 Population A items remains high). This simulation confirms that the global changes to the network made by standard back-propagation make it impossible to retain old items during sequential learning in a population of non uniform structure.

- Figure 2 roughly here -

Standard global back-propagation is easily capable of learning all 30 items from both populations if we abandon the strong sequential learning requirement. The "BP (conc)" condition shows the network learning each new item concurrently with all previously learned items (the simplest form of rehearsal, as discussed above). Items are added to the network one at a time, but old items must be available as the new item is added so that all items can be learned concurrently. Although this is only weakly sequential learning we include this condition to show that the network has the capacity to learn the required items, and to establish a baseline of "perfect" performance to compare the strongly sequential conditions with.

3.4. Local learning for sequential tasks

Having established that global learning algorithms are inappropriate for strongly sequential learning in a population of non uniform structure, we now consider a range of local learning algorithms applied to the same sequential task.

3.4.1. Pseudorehearsal

As noted above pseudorehearsal restricts changes to the output function of the network to be local to the region of the new item being learned.

In this application of pseudorehearsal to the sequential learning task we generate 128 pseudoitems each time a new item as to be learned. The new item is then learned in a buffer with 4 different pseudoitems randomly selected every epoch. The pseudoitems "map out" the function embodied by the network and make it resistant to change, except in the region where it is being pulled by the consistent error contributed by the new item.

The results are shown in the "PR" condition of Figure 2. Pseudorehearsal maintains excellent performance on the whole population as each new item is learned individually, even as we move from Population A to Population B items. In other words, pseudorehearsal provides an effective tool for sequential learning in this non uniformly structured population.

3.4.2 Radial basis function networks

Radial basis function (RBF) networks structure the "receptive neighbourhoods" of their hidden units so that these units divide the input space into regions. The output of a hidden unit represents a measure of the distance from the center of the unit's neighbourhood to the current input (Moody & Darken (1989)). The neighbourhood of the hidden units can be established by assigning the centers uniformly across the input space, by assigning

them to random input patterns, or by using some clustering algorithm. Hidden to output unit connections are then trained using the standard delta rule.

In this application of an RBF network to the sequential learning task we train the 3 hidden units of the network using an adaptive *k*-means algorithm (see for example Darken & Moody (1990)) which is essentially competitive Kohonen type learning. We use a width of 2 for the transfer function.

The results are shown in the "RBF" condition of Figure 2. The RBF network fares no better than sequential back-propagation, learning the consistent items from Population A adequately but suffering catastrophic interference when starting to learn Population B items.

Closer examination showed that this RBF network suffered from a fundamental limitation in this sequential learning situation. Learning the Population A items involved and modified all hidden units in the network. The first Population B item, however, was "captured" by the closest hidden unit, and then all subsequent Population B items involved and modified just that one hidden unit. Although RBF networks localise changes in the input space the clustering algorithm depends on concurrent learning to assign neighbourhood centers appropriately and does not extend robustly to sequential learning.

3.4.3. Hidden unit sharpening

Hidden unit activation sharpening was proposed by French (1992) as a way of reducing some of the effects of catastrophic forgetting. French noted that interference was caused by overlapping hidden unit representations and suggested that reducing this overlap should alleviate interference. Hidden unit sharpening involves adding to standard back-propagation an extra weight update phase for the input to hidden weights in the network. This update causes the *k* (one or more) most active hidden units become slightly more active and others to become slightly less active, thus "sharpening" the representation of the input and reducing the overlap of the hidden unit representations. This helps localise the representation of the input space by increasing the separation of the representations of dissimilar inputs.

In this application of hidden unit sharpening to the sequential learning task we set *k* to one (sharpening the most active hidden unit for each input) with a sharpening constant of 0.2.

The results are shown in the "Sharpened" condition of Figure 2. The sharpened network fares no better than sequential back-propagation, learning the consistent items from Population A adequately but suffering catastrophic interference when starting to learn Population B items.

As was the case with RBF networks this sharpening algorithm suffered from a

fundamental limitation in the sequential learning situation. Learning the first individual item to criterion (instead of learning many items concurrently) allowed the weights of the winning hidden unit for that input to become so dominant that the same unit remained the winner not only for every subsequent Population A item, but for all Population B items as well. This effectively increased the overlap of the hidden unit representations instead of decreasing it! Once again this concurrent algorithm does not extend robustly to sequential learning.

3.4.4. Nearest Neighbours

The *k*-nearest-neighbours (kNN) algorithm is often described as a local algorithm for classification tasks. In the first of the senses noted by Bottu & Vapnik (1992) (see Section 3.2. above) kNN based networks localise changes in the input space by manipulating the training set so that for a given item to be tested the network is trained on known items in a small neighbourhood around that item.

This general idea can be extended to our autoassociative learning network by simply using standard back-propagation on a training set consisting of the *k*-nearest-neighbours of the item to be tested. This approach is inherently concurrent however, as the entire population must be available so that the nearest neighbours can be selected and trained. There is no sensible way to apply a kNN based approach to a strongly sequential task as the algorithm assumes concurrent access to the entire training population.

3.5. Local learning revisited

Clearly local learning does not in and of itself provide an effective solution to sequential learning tasks. Of the local methods explored above only pseudorehearsal is capable of strongly sequential learning with a population of non uniform structure. RBF networks and hidden unit sharpening do not in their current forms generalise well to sequential learning. It may well be possible, however, to overcome the problems noted here and develop versions of these algorithms suited to sequential tasks, in which case it would be interesting to explore the consequences of their approaches to localisation in more detail.

It is clear that the concept of "local" learning is a complex one. To the methods of localisation identified by Bottu & Vapnik (see Section 3.2.) pseudorehearsal adds directly restricting changes to the output function of the network. Most local algorithms are based on restricting on the structure / representational power of the network. Pseudorehearsal restricts changes to the output function by using, like kNN methods, a manipulation of the training set (the inclusion of pseudoitems). Unlike kNN however the training set is not restricted in the input space, but exerts its influence directly on the output function via the errors back-propagated from the output units. The output function is "fixed" by the errors

associated with the pseudoitems, and can change only in the region where the constant influence of the error associated with the new item forces it to change. In short, pseudorehearsal works on the output function directly, rather than indirectly via restricted representations within the network.

Most local algorithms restrict the structure / representational power of the network. Although this may result in useful properties such as the advantages of RBF networks, we suggest that the costs of such restrictions may be particularly relevant in the case of sequential learning. Pseudorehearsal places no restrictions on representation, the network is free (as is the case using standard back-propagation) to adopt any representations allowed by the error function. This freedom may well be necessary to cope with arbitrary sequential learning tasks, and lack of it may hamper other local algorithms. For example, in an earlier study comparing pseudorehearsal and hidden unit sharpening (Frean & Robins, 1997) it was found that although pseudorehearsal naturally sharpened hidden unit representations, it also *reordered* them, whereas standard hidden unit sharpening (which was much less effective) entrenches existing orderings of activations by further enhancing the existing differences between units.

4. Local learning and generalisation

Generalisation is based on smooth interpolations (and, less reliably, extrapolations) of and between training data points in the function learned by the network. We might expect, therefore, that restricting changes to the function to areas which are local to each new item learned will have a detrimental effect on the ability of the network to generalise. This topic was explored in Robins & Frean (1997) where we concluded that local learning using pseudorehearsal appeared to significantly affect generalisation only in very small populations. In Figures 3 and 4 we further explore this topic by looking at generalisation in the sequential task described above.

- Figures 3 & 4 roughly here -

Figure 3 shows the performance of the network – as the training items are learned – on a test set (of size 20) of items drawn from the same distribution / species of iris as Population A. As expected, all learning methods generalise well to this test set while learning the first 15 items from Population A. However, only those methods resistant to catastrophic forgetting (standard concurrent back-propagation, and pseudorehearsal in the sequential case) continue to generalise well as the network switches to learning Population B.

Figure 4 shows the performance of the network – as the training items are learned – on a test set (of size 20) of items drawn from the same distribution / species of iris as Population B. In general, all learning methods generalise poorly to this test set while learning the first 15 items from Population A, but all methods generalise well as the network switches to learning Population B. The interesting exception to this general rule is standard concurrent back-propagation, which significantly improves its performance on this test set while learning Population A items. Whereas in every other case pseudorehearsal has closely matched the ideal performance of concurrent back-propagation, in this case there is a significant difference. This suggests that global learning (concurrent back-propagation) is able to exploit some similarities in the structure of Population A and Population B which local learning (pseudorehearsal) is not. In other words there may indeed be some significant costs to generalisation as a consequence of local learning. We hope to explore this issue in more detail in future work.

5. Summary

Catastrophic forgetting is a very basic consequence of a plastic representing medium, and most ANNs are vulnerable to this problem. Consequently most ANN learning algorithms involve learning all information of interest concurrently. Pseudorehearsal provides a useful solution to the catastrophic forgetting problem which allows information to be learned sequentially. The essence of this method is that it constrains changes to the function learned by the network to be local to the new item(s).

In this paper we have explored the application of pseudorehearsal and other local learning algorithms to a strongly sequential learning task, where items from a non uniform population are learned in isolation one at a time. Clearly local learning does not in and of itself provide an effective solution to such tasks. Of the local methods explored only pseudorehearsal was successful. RBF networks and hidden unit sharpening do not in their current forms generalise well to the sequential learning paradigm, although it may be possible to extend them.

The concept of "localisation" was found to be complicated, however. Different algorithms achieve localisation of learning in different ways, via restrictions on the training set, via internal constraints, and via restrictions on changes to the output function. Pseudorehearsal manipulates the training set by including pseudoitems which "map out' the structure of the output function and prevent it from changing except in the neighbourhood of the new item(s). In exploring the generalisation performance of the local algorithms we found evidence of a possible cost of localisation compared to global learning in some circumstances. We hope to explore this issue further in future work.

Extending the capabilities of current neural network learning algorithms to allow

sequential learning is an important goal. This should enable a wider range practical applications of networks to be developed, and topics such the consolidation of newly learned information, ongoing / lifelong learning, and developmental effects to be more easily modelled and explored within the neural network framework.

Acknowledgements

Thanks to Michael Webber and Janet Wilson who assisted in this study, and Simon McCallum for interesting discussions on pseudorehearsal.

References

B. Ans & S. Rousset, "Avoiding Catastrophic Forgetting by Coupling Two Reverberating Neural Networks", *C. R. Acad. Sci. Paris, Life Sciences, 320,* 989 - 997, (1997).

L. Bottu & V. Vapnik, "Local Learning Algorithms", *Neural Computation, 4,* 888 - 900, (1992).

C. Darken & J. Moody, "Fast Adaptive k-Means Clustering: Some Empirical Results", *Proceedings of the IJCNN, Volume II,* IEEE Neural Networks Council, 233 - 238, (1990).

R. Duda & P. Hart, *Pattern Classification and Scene Analysis",* New York: Wiley, (1973).

M. Frean & A. Robins, "Catastrophic Forgetting in Neural Networks: Further Exploration of The Pseudorehearsal Solution and Implications for Consolidation and Transfer", *Tech. Rep. AIM-36-97-2,* Otago, New Zealand: Otago University, Department of Computer Science, (1997).

M. Frean &A. Robins, "Catastrophic Forgetting and 'Pseudorehearsal' in Linear Networks", *Proceedings of the Australian Conference on Neural Networks,* in press, (1998).

R. French, "Semi-distributed Representations and Catastrophic Forgetting in Connectionist Networks", *Connection Science, 4,* 365 - 377, (1992).

R. French, "Dynamically Constraining Connectionist Networks to Produce Distributed, Orthogonal Representations to Reduce Catastrophic Interference", *Proceedings of the 16th Annual Cognitive Science Society Conference,* Hillsdale, NJ: Earlbaum, 335 - 340, (1994).

R. French, "Interactive Connectionist Networks: An Approach to the 'Sensitivity-Stability' Dilemma", *Connection Science, 9,* 353 - 380, (1997).

S. Grossberg, "Competitive Learning: From Interactive Activation to Adaptive Resonance", *Cognitive Science, 11,* 23 - 63, (1987).

P. Hetherington & M. Seidenberg, "Is There 'Catastrophic Interference' in Connectionist Networks?", *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* Hillsdale NJ: Lawrence Earlbaum, 26 - 33, (1989).

S. Lewandowsky & S. Li, "Catastrophic Interference in Neural Networks: Causes, Solutions, and Data", In F. Dempster & C. Brainerd (Eds.), *Interference and Inhibition in Cognition,* San Diego: Academic Press, 329 - 361, (1995).

J. McClelland, B. McNaughton. & R. O'Rielly. (1995) "Why There Are Complementary Learning Systems in the Hippocampus and Neocortex: Insights From the Successes and Failures of Connectionist Models of Learning and Memory", *Psychological*

Review, 102 (3), 419 - 457, (1995).

M. McCloskey & N. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem", In G. Bower (Ed.), *The Psychology of Learning and Motivation: Volume 23,* New York: Academic Press, 109 - 164, (1989).

J. Moody & C. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units", *Neural Computation, 1,* 281- 294, (1989).

P. Murphy & D. Aha, *UCI Repository of Machine Learning Databases* (http://www.ics.uci.edu/~mlearn/MLRepository.html). Irvine, CA: University of California, Department of Information and Computer Science, (1994).

J. Murre, *Learning and Categorization in Modular Neural Networks,* Hillsdale, NJ: Earlbaum, (1992).

R. Ratcliff, "Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions", *Psychological Review, 97,* 285-308, (1990).

A. Robins, "Catastrophic Forgetting, Rehearsal, and Pseudorehearsal" *Connection Science, 7,* 123 - 146, (1995).

A. Robins, "Consolidation in Neural Networks and in the Sleeping Brain", *Connection Science, 8,* 259 - 275, (1996).

A. Robins & S. McCallum, "Pseudorehearsal in Hopfield Type Networks", Connection Science, in press, (1998).

D. Rumelhart, G. Hinton & R. Williams, "Learning Internal Representations by Error Propagation", In D. Rumelhart, J. McClelland and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations,* Cambridge MA: MIT Press, 318 - 362, (1986).

N. Sharkey & A. Sharkey, "An Analysis of Catastrophic Interference", *Connection Science, 7,* 301 - 329, (1995).

D. Silver & R. Mercer, "The Task Rehearsal Method of Sequential Learning", Technical Report, Ontario: University of Western Ontario, Department of Computer Science, (1998).

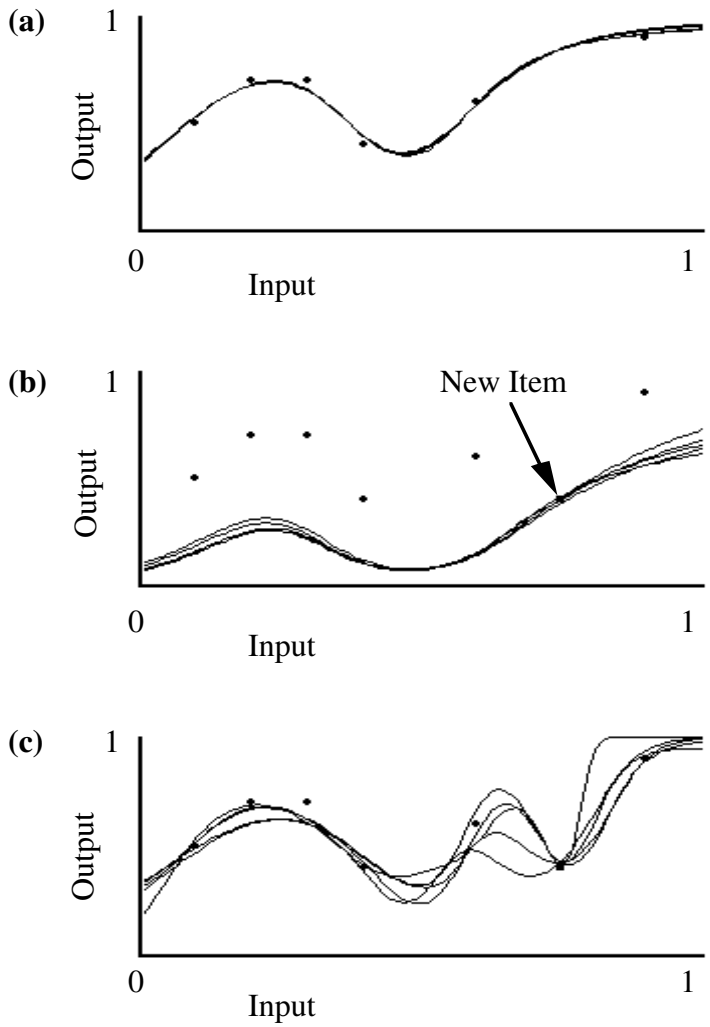

Figure 1. Functions learned by a network (five replications). Part (a) shows a network trained on six data points. Parts (b) and (c) show the subsequent learning of the same single new data point, using standard back-propagation learning and using pseudorehearsal respectively. (Adapted from Robins (1997)).

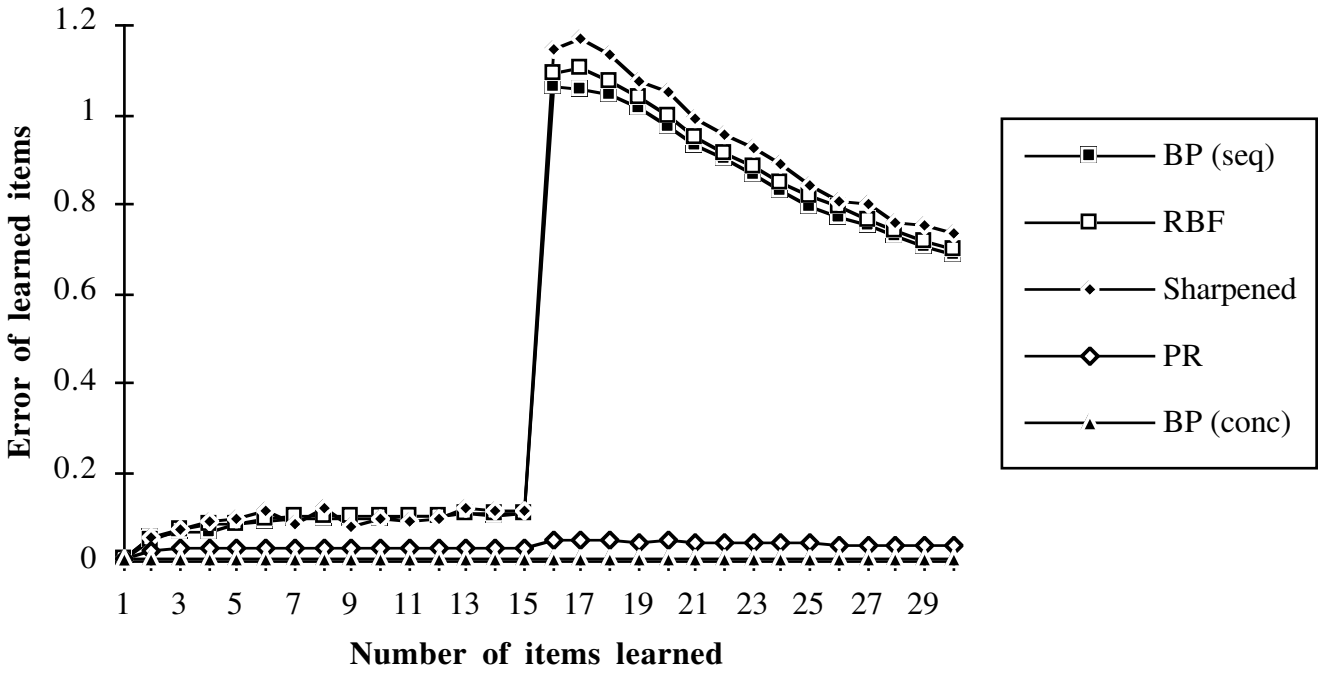

Figure 2. The average error of all previously learned items as the 30 items in the training set are learned in sequence. For example, after learning the eleventh item individually this figure shows the average error of the eleven patterns learned so far (for each of the noted learning algorithms).

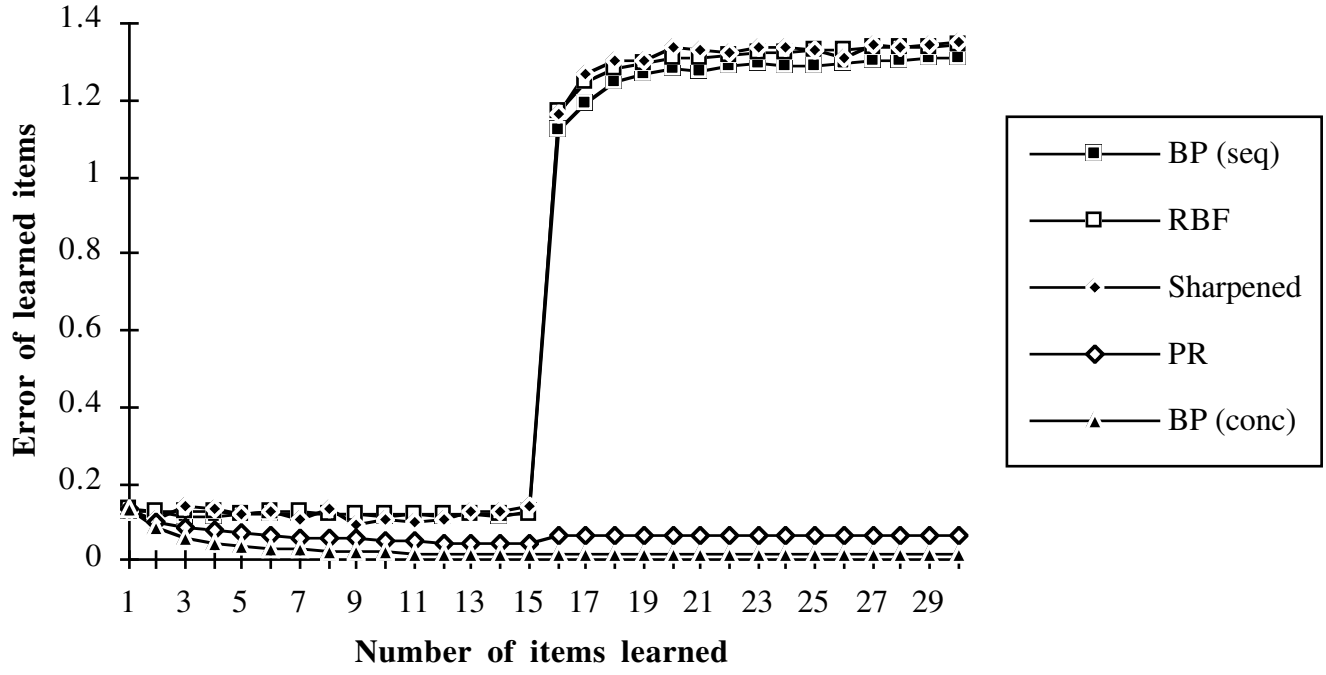

Figure 3. The average error of the 20 items in the test set drawn from the same distribution as Population A as the 30 items in the training set are learned in sequence.

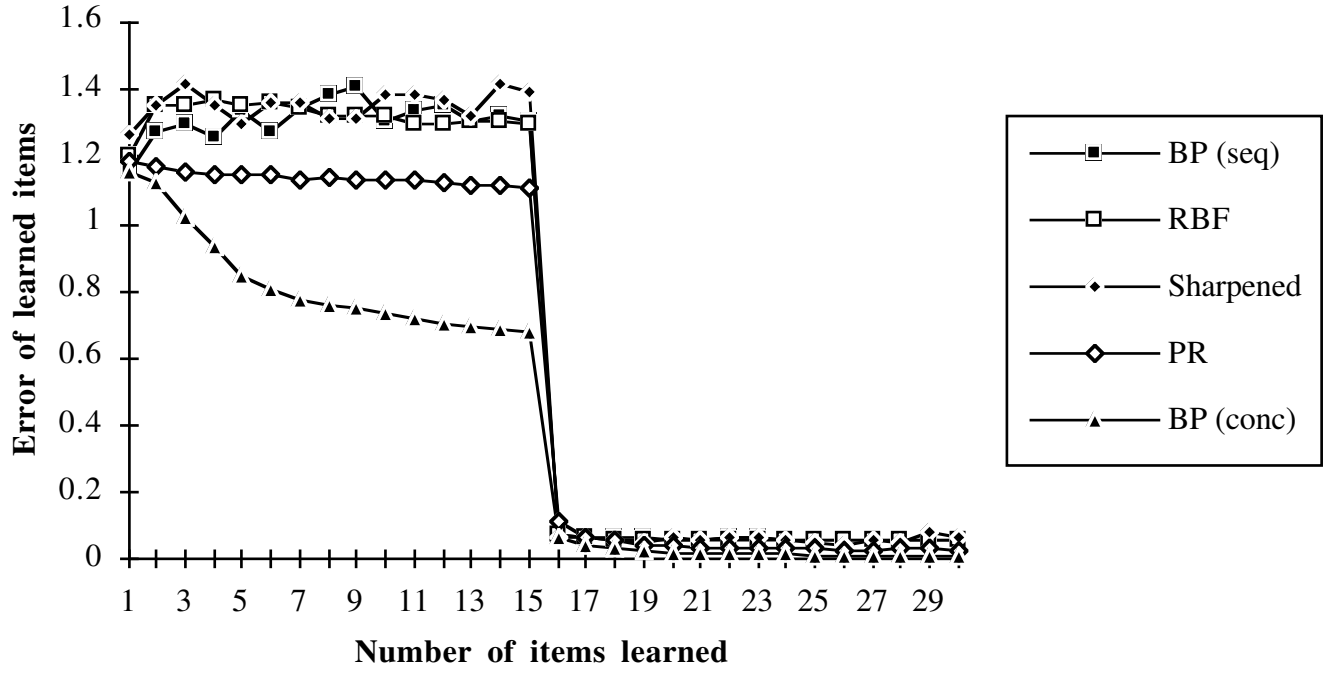

Figure 4. The average error of the 20 items in the test set drawn from the same distribution as Population B as the 30 items in the training set are learned in sequence.