This is a pre-publication draft. Please cite the published version:
Robins, A. Sequential learning in neural networks: A review and a discussion of pseudorehearsal based methods. Intelligent Data Analysis, 8(3), 301 – 322 (2004).

Sequential Learning in Neural Networks: A Review and a Discussion of Pseudorehearsal Based Methods

Anthony Robins
Computer Science
The University of Otago
Dunedin, New Zealand

Dr Anthony Robins
Computer Science
University of Otago
PO Box 56, Dunedin
New Zealand

Phone: +64 3 4798314
Fax: +64 3 4798529
Email: anthony@cs.otago.ac.nz

Short title: Sequential Learning in Neural Networks
Keywords: sequential learning, catastrophic forgetting, rehearsal, pseudorehearsal, consolidation

Abstract
In this review we explore the topic of sequential learning, where information to be learned and retained arrives in separate episodes over time, in the context of artificial neural networks. Most neural networks handle this kind of task very badly, as new learning completely disrupts information previously learned by the network. This problem, known as "catastrophic forgetting", has received a lot of attention in the literature. We illustrate the catastrophic forgetting effect, and summarise possible solutions. In particular, we review the literature relating to the pseudorehearsal mechanism, which is an effective solution to the catastrophic forgetting problem in back propagation type networks. We then review similar issues of capacity, forgetting, and the use of pseudorehearsal in Hopfield type networks. Finally, we briefly discuss these issues in the context of cognition, and summarise interesting topics for further research.

1.0 Introduction

The issue of learning in a dynamic environment, including such topics as conceptual drift (where the target concept changes over time) and sequential learning (where information arrives in separate episodes over time), represents a serious challenge for learning systems. Ideally the representations developed by learning should be stable enough to preserve important information during new learning, but plastic enough to incorporate new information when necessary. These requirements constitute the "stability / plasticity dilemma" [27] – both stability and plasticity are desirable, but their requirements are in direct conflict. In this paper we review and discuss the impact of these issues on artificial neural networks (ANNs). For a general introduction to the field (and a description of the back propagation and Hopfield type networks discussed in this paper) see introductory texts such as [28, 29].

As a style of computation ANNs have many useful features, such as content addressable memory, robustness to noise and damage, automatic generalisation, and so on. As a basis for modelling cognition they are in general a natural fit to the similar strengths and weaknesses of human information processing. With respect to sequential learning, however, most ANN learning algorithms perform spectacularly badly. This problem has been described in the literature using terms such as the "catastrophic forgetting" (CF), "catastrophic interference", or "serial learning" problem. The essence of CF is that *new learning will typically disrupt or even eliminate any information previously learned by the network.* This is the reason that typical ANN learning algorithms are based on *concurrent* learning, where 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 contrast, human learners are clearly capable of ongoing or *sequential* learning – integrating old and newly learned information as it is acquired. In short, CF is a property of typical ANNs that is uncharacteristically at odds with the capacities of the real neural networks in brains.

The cause of the CF problem is easy to identify. The use of variable connection weights as a medium for encoding information leads most ANNs to err in the direction of excessive plasticity. New learning changes the weights, thus causing previously learned inputs to generate the wrong outputs. This represents a major problem for ANN learning algorithms. CF limits the power of ANNs as general learning systems, and hugely constrains them as a framework for modelling cognition.

This topic is usually studied in the context of feed forward "multi layer perceptron" (MLP) type networks such as the popular back propagation family. In their early studies of CF [40, 48] the authors suggested that it might be a fundamental limitation of this general class of ANNs. Subsequently of course, CF and possible solutions to the problem, attracted a great deal of interest. Other important early studies include [17, 18, 30, 36, 37, 41, 45, 46]. For reviews see [63] or [21] for a recent and comprehensive overview. Although the two literatures have seldom been connected, related issues of capacity and catastrophic overload / forgetting also occur in the Hopfield / constraint satisfaction family of dynamical networks. Early studies include [5, 10, 33, 47].

In short, most ANN learning algorithms deal very poorly with ongoing / sequential learning. Excessive plasticity leads to the catastrophic forgetting of old information. The purpose of this paper is to review and discuss the CF problem and its solutions. In

particular we focus on a family of solutions based on pseudorehearsal [51]. Pseudorehearsal is one of the most widely studied solutions to CF, and the only one which has been applied to both the back propagation and Hopfield type families of networks. Since the method was proposed there have been a range of studies based on or related to it, including [1, 2, 3, 4, 16, 19, 20, 22, 23, 24, 38, 42, 52, 53, 54, 55, 56, 57, 58, 59, 65, 66]. In reviewing this literature we hope to summarise results to date, provide an overview of how the various themes relate to each other, and identify interesting topics for further research.

In the sections below we first summarise the CF problem and possible solutions to it (such as "reduced overlap" methods) in the back propagation family of networks (Section 2). We briefly describe the pseudorehearsal solution in general terms (Section 3). We explore various ways of implementing pseudorehearsal in ANNs, and summarise results relating to pseudorehearsal and related methods (Section 4). We then discuss (Section 5) CF and related issues in Hopfield type networks, and explore the relationship between pseudorehearsal and an alternative approach, the "unlearning" technique of Hopfield *et al*. [33] Crick & Mitchison [10] and others. In the discussion (Section 6) we briefly explore the issue of how these techniques might be related to learning and consolidation in real brains, focusing on the complementary learning systems proposed by McClelland *et al*. [39], and the possible role of sleep.

2.0 Catastrophic forgetting

In this and the following sections we summarise the CF problem and possible solutions to it in the back propagation (generalised delta rule) family of networks. These networks consist of three or more layers of units, with "feed forward" connections between layers. Inputs are presented at the first (input) layer, fed forward through any intermediate (hidden) layers, and create output patterns on the final (output) layer. Weights are adjusted so as to reduce error when outputs are compared to a "teaching" pattern which is simultaneously presented to the output layer. The input population is presented as often as required to reduce the output error to an acceptable criterion (each presentation is called an "epoch"). These networks can be used to learn mappings between input and output populations of patterns.

2.1 The problem

As CF has been well discussed in the literature we provide only a brief introduction here, see also [51, 63], or see [21] for a recent and comprehensive overview.

In typical illustrations of CF a back propagation type network is used to learn a "base" population of items (input / output pattern pairs). Subsequently a sequence of new items or groups of items is learned. The effect of this new learning on old information can be illustrated by re–testing the base population after learning each new item. The error of the old information (base population) typically increases catastrophically after any subsequent learning event.

In Figure 1 the "standard" condition illustrates this effect using a real world data set[1]. Initially ("0" on the *x* axis) the base population is learned to criterion. Subsequently, learning even a single new item causes a huge increase in the base population error. Although most of the damage is done by this first learning event, up to a point learning further new items causes even more error / forgetting. In short, new learning hugely disrupts the old information.

Various factors influence the extent to which forgetting occurs. In the special case that the new item to be learned is drawn from exactly the same distribution as the base population, CF may not occur at all [52]. In this case (due to generalisation) very few presentations are typically required to learn the new item. In general, however, the new items to be learned by a network can be drawn from significantly different distributions, and (as in Figure 1) CF is a serious practical problem. Ratcliff [48] notes that items learned in groups are significantly more resistant to forgetting than items learned singly. French [17] suggests that the extent to which CF occurs is largely a consequence of the overlap of distributed representations and can be reduced by reducing this overlap. Sharkey & Sharkey [61, 62] provide a useful overview of several practical issues, noting that catastrophic forgetting occurs most significantly in cases where training is sequential and without negative exemplars. Catastrophic forgetting will be worst when similar

[1] These results were originally presented in various simulations in [52]. They are based on a 4:3:4 autoassociative back propagation network with a learning constant of 0.05, momentum of 0.9, a sum of squares error measure, and an error criterion of 0.01. The network was trained to criterion on a base population of 30 examples of one species of iris from the well known Iris data set [44], then further trained on a sequence of 20 new irises drawn from a second species. All results were averaged over 50 replications of the simulation (using different populations for each replication). The same results are observed using either batch or online weight updates.

inputs, generating similar hidden unit patterns, require very different output patterns to be produced.

2.2 Scope of the problem

CF is a serious problem for most ANNs, including those based on the popular back propagation type algorithms. Related issues arise in the Hopfield type family of dynamical networks. This is the reason that most learning algorithms are based on training concurrently (with all training data available at the same time, and used in a number of training presentations / epochs until the desired performance is achieved). Training is then regarded as finished, and no further learning is attempted. Such restrictions limit the power of ANNs as general learning systems, and dramatically constrain them as a framework for modelling cognition.

By no means all ANNs exhibit CF however. The problem arises because learning (changing weights) for one item disrupts the behaviour of the network for other items. Obviously if items effectively share no weights then no forgetting will occur. This is the case for "localist" connectionist networks (where concepts are encoded using single units, see for example [14]), and ANNs using distributed representations that are orthogonal.

Fully localist and fully distributed representations are two ends of a continuum. Between these end points, CF is reduced to varying degrees in architectures with significant amounts of localisation and / or physical separation of the representations of information. One of the first examples was the original ART1 network [7, 8] which encodes new inputs (when sufficiently novel) using previously unused units (drawn from a fixed "pool" in the F2 layer). Although not usually noted in the CF literature, Carpenter

& Grossberg [8] explicitly describe the ability of ART1 to continuously learn new inputs as a significant advantage compared to other typical network architectures. CALM, an explicitly modular network architecture [45], also stores new inputs using previously uncommitted nodes. Similarly, CF may be reduced to varying extents for "constructive" learning algorithms that learn by adding new units and connections[2]. See [21] for further discussion of these and related methods.

A different approach to separating representations is illustrated by the ALCOVE network [35]. ALCOVE has a hidden layer of radial basis function units that partition the input space. In this case representations of new inputs that are not similar to previously learned items will necessarily activate different hidden units, and so reduce CF. By tuning the shape of the receptive fields a balance of "semi–distributed" representations can be achieved. Once again, see [21] for a discussion.

These kinds of network are generally less vulnerable to CF. Furthermore the principle of reducing the overlap of representations can be extended to standard ANNs which employ fixed architectures and fully distributed patterns of activation. This constitutes one of the two main families of methods which have been explored (in back propagation type networks) as possible solutions to the CF problem, as described in Section 2.3 below. In Section 3 we describe the second main family of possible solutions, the rehearsal based approach which forms the focus of this paper.

[2] More research is required to assess the extent to which constructive algorithms provide protection from catastrophic forgetting. Work currently in progress [12] shows, for example, that cascade correlation [13] is vulnerable to the problem.

2.3 Reducing overlap

Several authors have proposed ways of reducing the impact of CF in back propagation type networks which are based on the general principle of separating representations. This approach can be summarised by French's observation [17] that the extent to which CF occurs is largely a consequence of the overlap of distributed representations, and that its effects can be reduced by reducing the overlap.

In the networks described above the overlap between representations is reduced by physically segregating the network into modules, regions, or possibly single units using localist representations. In back propagation type networks reduced overlap can be achieved by modifying the learning algorithm (and / or training regime) so as to create sparser hidden unit representations (patterns of activation with fewer active units). Examples of this approach include the novelty rule [34], activation sharpening [17], context biasing [18] and techniques described in [41, 46]. Context biasing, for example, introduces an extra step to the learning process which adds weight changes that enhance the differences between the pattern of hidden unit activation generated by the current input and the pattern generated by the previous input.

Reduced overlap methods do not in general prevent CF when measured, as in Figure 1, by the ability of the network to correctly reproduce previously learned outputs (which French [21] calls an "exact recognition" measure). They do, however, mitigate its effects to the extent that after new learning, old information can be more quickly relearned by the network[3] (which French calls a "savings" measure). A further consideration when

[3] When compared to standard back propagation for the same learning regime.

using these methods is the complicating issue of the undesirable side effects of sparse hidden unit representations. French [18] identifies several problems that arise, arguing that they result in a reduced capacity to categorise and discriminate inputs, a reduced capacity to generalise, and even in some cases an *increased* vulnerability to CF. Sparse (more "localist") representations would also imply a decreased robustness in the face of noise and damage. Similarly, the use of distributed (non sparse) but separated (non overlapping) representations would reduce the abilities of networks to extract central / prototypical information and to generalise.

3.0 The pseudorehearsal solution

The main focus of this paper is on pseudorehearsal, which is an example of the second main family of possible solutions to CF, methods based on rehearsing / relearning old information. In Sections 3 and 4 we describe rehearsal based methods in general, and pseudorehearsal in particular, in the context of back propagation type networks. The use of these methods in Hopfield type networks is discussed in Section 5.

3.1 Rehearsal and pseudorehearsal

An obvious mechanism for protecting old information is to rehearse / relearn it along with the new information. Rehearsal was first explored in the context of catastrophic forgetting by Hetherington & Seidenberg [30] and Ratcliffe [48], and a range of specific methods have been explored [46, 51].

Following Ratcliffe [48], rehearsal can be thought of as introducing each new item not alone, but in a *rehearsal buffer* along with a number of old items. The population of items in the rehearsal buffer are then trained over a number of epochs in the usual way. If *all* old items were included, rehearsal would simply amount to retraining the entire base population as new items are introduced (as is the case in for example the particular implementation of "interleaved learning" used by McClelland *et al.* [39]). However, subsets of the base population or less rigorous training criteria can also be used effectively. The various possible ways of selecting and managing the old items in a rehearsal buffer define a family of possible *rehearsal regimes*. Robins [51] explored a range of regimes. By far the most successful was the "sweep" regime, where the

rehearsal buffer always contains the new item, and also contains a number of old items that are randomly selected *for each epoch* of training.

As shown in Figure 1, rehearsal condition, CF (in this real world data set) can be completely eliminated. Each new item is learned (to criterion) using a sweep rehearsal buffer (five old items were trained each epoch, replacing those used in the previous epoch so that the buffer remains of a fixed size). During this process the performance of the network on the old information (base population) remains excellent. Sweep rehearsal is very effective despite the fact that it does not use all old items every epoch, and does not explicitly retrain old items to criterion. This suggests that in general rehearsal should be "broad" but it does not need to be "deep".

Rehearsal is effective, and could be used as a practical method for building ANN applications. It may be, however, that the old items have been lost, or it is not practical for some reason to store them. In particular, rehearsal is unsatisfying as a framework for modelling cognitive processes. New learning can only take place if all old information learned by the organism is available from some other internal source for rehearsal. Not only is this impractical, but it seems to make the rehearsal based memory system itself redundant!

It is possible to achieve the benefits of rehearsal, however, even when there is no access to the previously learned information. In other words, we can do rehearsal even when we do not have the old items to rehearse. This "pseudorehearsal" mechanism [51] is based on the use of artificially constructed populations of "pseudoitems". A pseudoitem is constructed by generating a new input vector at random, and passing it forward through a network in the standard way. Whatever output vector this input

generates becomes the associated target output. For a given network (trained on some base population) a population of pseudoitems constructed in this way can be used instead of the actual base population items in any rehearsal regime. Learning proceeds exactly as before, except that instead of rehearsing items chosen from the old base population, they are chosen from the population of pseudoitems.

For example, using the task described above (see Figure 1), in the pseudorehearsal condition a population of pseudoitems is constructed before each new item is learned. The new item is then learned along with pseudoitems that are chosen at random for each epoch[4]. Training continues in this way until the new item is trained to criterion. Figure 1 shows the results for pseudorehearsal, which is highly effective at preserving the base population – despite the fact that no real base population items have been rehearsed. Although not evident from this graph the error for pseudorehearsal (which reaches 0.019 by the twentieth new item) is slightly higher than for real rehearsal (which remains at 0.009, just below the training criterion of 0.01).

In short, pseudorehearsal is an effective method for achieving the benefits of rehearsal in reducing catastrophic forgetting, without assuming access to old information. Rather than explicitly storing all learned items for later rehearsal, pseudorehearsal *approximates* this information whenever it is needed. Using back propagation type networks the method has now been applied to a number of data sets, including the Iris data used in [52] and summarised above; autoassociative and heteroassociative randomly constructed data sets [1, 51]; classification tasks using the Mushroom and Congressional Voting Records data sets (see [44]) and a Cat and Dog picture dataset [19, 20]; artificial and natural

categories [24]; an artificial 2D domain [66]; a coronary artery database [65] and a domain of arithmetic facts [2]. Using Elman type networks pseudorehearsal based methods have also been shown to be effective at preserving learned *sequences* [3, 4].

3.2 Functions and local learning

Why does pseudorehearsal work? Back propagation type networks can be interpreted as function approximators. The training population defines input / output pairs in some (usually multidimensional) space, and as a result of training the network fits a function to the training points. We can interpret both rehearsal and pseudorehearsal in this context. During further training a network using rehearsal retrains the originally learned points on the function (the base population). In the pseudorehearsal case the network retrains other randomly chosen points on the function (the randomly sampled input/output mappings that are pseudoitems).

We can illustrate this interpretation directly using (as proposed by Frean in [16]) a back propagation network with just a single input and a single output unit. This allows us to plot both the actual training data (input value / output value pairs) and, by sampling the input "space" (systematically inputting values in the range 0 to 1), the function learned by the net. Figure 2 illustrates learning in such a network. Part (a) shows the function fit to a base population of six training items / points (in each of five replications)[5]. As

[4] Each population of pseudoitems consisted of 128 items, of which five were used in each training buffer / epoch.

[5] The network consists 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. In the pseudorehearsal condition described in part (c) we use a population of 20 pseudoitems and a training buffer which contains 10 pseudoitems chosen at random each epoch.

expected, the function passes through the training points. From this starting state part (b) shows the effect of learning a single new item. The new data point is correctly learned, but the whole function has been moved to fit it. Base population inputs will now be mapped on to the wrong outputs – in other words CF has occurred. If however (from the original starting point shown in (a)) we learn the new item using pseudorehearsal, then we arrive at the function shown in part (c), which fits both the new data point and the base population points adequately. CF has been prevented not by rehearsing the base population items themselves, but by rehearsing pseudoitems (other randomly chosen points on the function).

As shown by this simulation, the essence of the pseudorehearsal solution is to restrict changes to the function learned by the network, so that new learning causes only *local* changes, leaving the rest of the function unaffected. Pseudoitems exert their 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 is stronger than the "inertia" generated by the pseudoitems.

Bottu & Vapnik [6] describe an algorithm as "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 [6] show that the k-nearest-neighbour (kNN) algorithm [11] is a particular case of the first approach, and that radial basis function (RBF) networks [43] are an example of the second. Clearly the pseudorehearsal mechanism illustrates a third way in which learning can be local, namely a manipulation of the training set that directly restricts changes to the output function. Robins & Frean [56] discussed the relationship between pseudorehearsal and other local learning methods such as RBF networks, activation sharpening [17] and kNN. Pseudorehearsal was shown to be significantly more effective than the other local methods tested on a serial learning task. Whereas most local algorithms are based on restricting the structure or representational power of the network, pseudorehearsal works on the output function directly, without restricting representations within the network. Pseudorehearsal is effective and flexible because the network is free (as is the case using standard back propagation) to adopt any representations allowed by the error function.

In short, pseudoitems "approximate" the old population and "map out" the function learned by the network. To implement pseudorehearsal effectively the network should be constructed so as to generalise well (fit the base population with a smooth function), so that pseudoitems are systematically related to the structure of the base population. Pseudorehearsal is most effective when the pseudoitems are generated from the same underlying distribution as the actual training data. Similarly, French *et al*. [22] argue that where the density of the data varies over different regions of the function then the density of the pseudoitems generated should match this distribution[6]. However, randomly

[6] One way of generating pseudoitems which were systematically similar to learned items would be to extract and retain just the principle components of the learned population, possibly using a separate small

generated / evenly distributed pseudoitems are very effective for most tasks, and for example have been shown to work well for a range of structured real world data sets (as described in Section 3.1). 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 samples the structure of the function in more detail, and a large training buffer fixes the function in more places. The "sweep" method described above (for 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.

3.3 Storage issues

Rehearsal requires the network to have available, in some other memory store, old learned items for relearning in the network. Such a store must be long term, stable, and very specific in its contents. A major advantage of pseudorehearsal is that it does away with this requirement. There is, however, apparently a need for a temporary store to hold the population of pseudoitems while it is used in the rehearsal process. As pseudorehearsal was initially described and implemented [51] this issue was not explored. A temporary store for pseudoitems, as shown schematically in Figure 3 (a), was simply assumed.

---

unsupervised network. This information could then be used in the generation of similarly structured pseudoitems. We are grateful to Jerome Friedman (personal communication) for this suggestion.

Ideally, and particularly for cognitive modelling, we would like to avoid the use of any explicit store and generate pseudoitems "on demand". In order to be useful pseudoitems must be created from the original function, and used to constrain the subsequent process of changing the function during further learning. Therefore, any mechanism for generating pseudoitems on demand will have to somehow preserve information about the original function, even as the function embodied by the weights of the network changes.

In the following section we explore various proposals for solving this problem, and implementing pseudorehearsal in a realistic ANN without using a separate store of pseudoitems.

4.0 Pseudorehearsal based ANNs

In Sections 4.1 to 4.3 we describe and discuss various pseudorehearsal based ANNs that do not require an external pseudoitem store. In Section 4.4 we note further results arising from explorations of pseudorehearsal and of these networks.

4.1 Dual networks

The most thoroughly explored approach to implementing a pseudorehearsal based ANN is the dual network method independently proposed by [1] and [19]. The insight underlying this approach is that a learning system can consist of two networks, one with the main function of learning items as usual, and one with the main function of representing previously learned information and using it to generate pseudoitems.

In [1] the authors proposed two separate networks, as illustrated in Figure 3 (b), and two distinct phases of training. We can summarise the procedure as follows. Assume that both networks (in the simplest case these are typical feed forward networks) start in the same state, encoding some previously learned knowledge / function. In the first phase of learning a new item is to be learned by one of the networks, designated Net1. The item is learned by Net1 along with pseudoitems generated on demand from Net2. To generate pseudoitems the same random input is presented to Nets 1 and 2, and the output from Net2 is used as a teaching signal for Net1 (and compared to the Net1 output in the usual way). As Net1 learns Net2 does not change its weights at all, they remain an accurate representation of the original function. In short, Net2 is serving exactly the same role as an external pseudoitem store. Otherwise the learning process is the same,

the new item is learned along with pseudoitems that represent and therefore preserve the original function.

In the second phase of learning the roles are reversed, and given the same pseudoitem inputs to both nets Net1 is used to generate pseudoitem outputs that are used to teach Net2 (no external / new items are learned by either network in this phase). The purpose of this phase is to "update" Net2 so that it represents the current state of Net1 (which currently combines old information and the newly learned information)[7]. The system is now ready for further learning – as further new items are presented the first phase of learning begins again.

French [19] also proposed a dual network strategy, but his "pseudorecurrent" architecture has shared input units at the first layer, as illustrated in Figure 3 (c). This simplifies the presentation of pseudoinputs (which must be the same for both networks). French [19] demonstrated a two phase learning process like [1], but most of his simulations were conducted using "weight copying", where after the first phase of learning the second phase is assumed and the weights of Net1 are simply copied to become the new weights for Net2. (In this case Net1 and Net2 must have the same internal architecture, which as noted above is not necessarily the case when pseudoitems are used to transfer information between networks). Weight copying is obviously not biologically plausible, it is just a practical simplification of the full two phase learning

[7] As the dual network approaches illustrate, pseudoitems can be used not only to consolidate information within a network (preserve the original learned function), but also to "transfer" information between networks (communicate a learned function from one network to another). As shown by [54] this transfer is effective even between networks of different internal architectures (number and size of hidden unit layers), though the size of the input and output layers must be the same so that pseudoitem inputs and outputs can be used by both networks.

process. Some interesting results produced with the pseudorecurrent architecture are noted in Section 4.4 below.

The behaviour of a simple dual network architecture using weight copying will be exactly the same as for a single network using an external pseudoitem store – the methods are formally equivalent (if the store is managed in such a way that each pseudoitem is used without repetition). The practical difference lies in the way in which pseudoitems are managed: in one case they are generated by the original network in advance and stored, and in the other they are generated by an exact copy of the original network as needed. Using full two phase learning instead of weight copying will result in very similar (though not identical) behaviour.

The pseudoitem based transfer of information (where one network "teaches" another) seems, in the context of the brain, like a useful and plausible mechanism by which different assemblies or structures could communicate information to one another. In fact if we are to assume that memory related processes (and cognition in general) involve the transfer of information between neural structures at all, it must be done using either exact learned items / patterns, or pseudoitem–like approximations.

4.2 Dual weights

The underlying requirement in implementing pseudorehearsal is that pseudoitems need to be generated from the original learned function (in order to preserve that function while integrating new information). The dual network approach is one way of achieving this. Robins [53] independently proposed a related method for implementing pseudorehearsal within a single network.

Robins proposed a network where each connection is assumed to be able to represent two weights, as pictured in Figure 3 (d). These dual weights can be used to implement pseudorehearsal as follows. One set of weights, e.g. w_{ij} in the figure, operates as usual for a back propagation type network. Assuming that the network has done some learning (learned the base population) the other set of "old" weights o_{ij} start with the same initial values. As new items are learned the weights w_{ij} are adjusted as usual, but the network also learns internally generated pseudoitems. To generate pseudoitems random inputs are presented and fed forward using each set of weights in turn. The "output" generated by the weights w_{ij} is compared with the "teacher" generated by the weights o_{ij} and the weights w_{ij} are adjusted in the usual way. As the weights o_{ij} are not adjusted by learning they remain an accurate representation of the original learned function / base population and can be used to generate pseudoitems as required. When the new items have been learned the weights w_{ij} will encode both old and new items. On each connection the weight o_{ij} updates itself to be a copy of the weight w_{ij} , and the network is ready for further learning.

This interpretation of a dual weight network is obviously similar to the dual network systems described above, but it is as if the networks are "superimposed" on each other and share the same units at all layers. Other variations within a dual weight approach are possible. As originally described and implemented by [53] one weight was used as a fixed base weight to generate pseudoitems, and the other as an offset which was added to the base weight and adjusted during the course of learning new items.

While the dual network approach is the more fully explored, the dual weight approach seems to be simpler in some respects. Most obviously, a dual network system requires a

more complex architecture, using roughly twice as many units, and requiring accurate unit to unit connections (in both directions) between the networks, for example between the output layers of the two networks to convey teaching patterns. Dual networks also require that the "contents of memory" (the function encoding the previously learned items) are continuously being moved: from Net2 to Net1 in the first phase of learning, and back from Net1 to Net2 in the second. This involves a cost in time (two learning processes) and risk from damage or other kinds of breakdown. In contrast the dual weight approach does not require extra units or connectivity. All learning and consolidation occurs in the same network, the contents of memory are not "moved". Only a single learning phase is required (and then a step in which one weight on each connection is updated using the other).

This simplicity comes as the cost of assuming two weights per physical connection, but we suggest that this is not implausible. In terms of the neural realisation of ANN constructs there seems to be no reason to insist that the "units" of the nervous system must have "connections" that can carry only one "weight". In general, central nervous system function appears to involve diverse mechanisms of processing and learning. The pattern of firing (in a given neuron) is mediated by complex membrane potential effects which are based on the structure of the cell, and the location, intensity, and temporal properties of the input stimulus. Different types of neuron also have different activation properties. A given cell can make more than one kind of synapse with subsequent cells, and a given synapse can release different neurotransmitters under different conditions. Information can also be encoded in a pattern of firing in many different ways (such as frequency coding, burst duration coding, phase coding, and so on). Several mechanisms

of synaptic modification have also been identified, including different kinds of long term potentiation effects. In short, we suggest that the use of different kinds of "weights" with different functions in learning is biologically plausible. In the ANN literature the use of two distinct weights for each connection has also been proposed (for other purposes) by Hinton & Plaut [31].

4.3 Other architectures

The dual weight approach shows how learning could be consolidated within a single network. The exchange of pseudoitems between dual networks is a different approach, and a useful mechanism by which different assemblies or structures could communicate information to one another. It is certainly possible that other methods of implementing pseudorehearsal could be found. We suggest, however, that the methods outlined above are the most direct and plausible, and that analogous mechanisms could actually be occurring in the brain (see Section 6).

Moving beyond implementations in basic feed forward networks offers some interesting possibilities. French *et al*. [22] argue that, particularly for learned functions that are very regular over certain regions, randomly generated pseudoitems may not be as effective as pseudoitems generated so as to particularly sample the most relevant regions of the learned function. Demonstrating at least one way in which this might be accomplished, Ans & Rousset [1, 2], as well as the simple feed forward version of their system, also explore the use of recurrent connections in their networks. Recurrent connections are used between the input and hidden layers to implement a process of "reverberation" (feedback) between the layers. Random inputs are reverberated between

layers for several iterations before being fed forward as the pseudoitem input in the usual way. These reverberated pseudoitems are more effective than the usual randomly generated pseudoitems for the tasks explored, and the authors suggest that they function like network attractors and better capture the underlying structure of the learned populations. Similarly, Friedman (personal communication) has suggested that a simple unsupervised net which extracts the principle components of learned populations could also be used to generate pseudoitems which are highly representative of the original training data. Robins & McCallum [58, 59] have, as discussed below, explored pseudorehearsal in the context of Hopfield–type networks, where network attractors are the most effective pseudoitems.

4.4 Further results

The above review has covered most of the basic results and topics of interest. In this section we briefly note some remaining results which have arisen from an exploration of pseudorehearsal and related topics.

One phenomenon links both pseudorehearsal and the reduced overlap family of solutions to CF (Section 2.3). As shown in [19, 55] the pseudorehearsal process naturally results in a moderate sharpening of hidden unit representations. Unlike explicit methods for reducing overlap, which exaggerate / sharpen existing representations, the sharpening which arises from pseudorehearsal can also involve a re–ordering of the most active units [55]. This suggests that considerable flexibility is required to solve the learning task, and emphasises the fact that pseudorehearsal is focused not on maintaining weights, but on maintaining the behaviour of the network.

French and colleagues [19, 20, 24] have explored various topics relating to modeling cognitive processes with pseudorehearsal based learning. In particular French [19] showed that his "pseudorecurrent" architecture exhibited plausible list–length and list–strength effects, both topics which have been well explored in the psychological literature (see for example [49, 50, 64]).

Ans *et al*. [3, 4] show that pseudorehearsal based methods are effective at preserving learned *sequences* in Elman type networks. Frean & Robins [16] present, in the context of single layer linear networks, a formal analysis of pseudorehearsal, and show that in some cases (for this kind of network) there is an optimal version of pseudorehearsal which is equivalent to a simple modification of the delta rule. In a detailed formal analysis French & Chater [23] generalise the concept pseudoitem to the point that explicit error information is no longer required, and error surfaces can be approximated (using first–derivative information) purely from noise.

5.0 Hopfield type networks

Although the two literatures are almost completely distinct, very similar issues relating to ongoing learning and capacity arise in the context of Hopfield type networks. These networks (see for example [28, 29]) consist of a single layer of units, usually fully connected with symmetrical weights. They can be used to learn (autoassociate) patterns, and retrieve these patterns from partial input cues (thus functioning as a "content addressable memory"). From a given input state, activation iterates through the network until a stable state (or "attractor") is reached. The stable states of a network (defined in terms of an energy function) may correspond to learned patterns, or to other "spurious" states created by interactions between learned patterns.

In this section we explore the use of rehearsal and pseudorehearsal based mechanisms in this family of ANNs.

5.1 Capacity and forgetting

As originally proposed [32] and usually implemented, this family of networks is trained using variants of Hebbian learning. Each pattern to be learned is presented, and the weights between each pair of units is adjusted in proportion to the product of the activations of the units (so that strong weights develop between coactive units). Only a single epoch (presentation of the population) is required. (A second epoch would simply double the magnitudes of the weights, without changing their distribution in any way). The properties of this approach to learning have been extensively explored. In particular, the capacity of networks trained in this way (the number of arbitrary patterns which can

be stored) is roughly $0.14N$, where N is the number of units in the network (see discussion in [29]).

With Hebbian learning the input population is never treated as a "whole" (in the way it is in error correcting algorithms such as back propagation, where the whole input population must be concurrently available to be re–presented over a number of epochs). Instead, each input item / pattern is presented just once. Learning is inherently serial, it can be thought of as a sequence of discrete events, and these events could be occurring at any time.

In this case any division of items into a "base population" and "new items" would be arbitrary. The question of how new learning effects old becomes mostly a matter of how many of a sequence of trained items the network can successfully store. This is exactly the question of capacity, which as noted above is known to be approximately $0.14N$.

Table 1 illustrates the performance of a typical Hopfield network as this limit is approached and exceeded[8]. The network is trained on populations of various sizes, as shown in the first column. The next group of three columns shows the performance of the standard Hebbian training procedure. "Stable" indicates how many of the trained patterns are stable when presented as inputs to the network. "Found" indicates how many of these stable states are actually found when sampling the network using 1000 randomly constructed inputs (each of which is iterated until it converges on a stable state).

[8] The network consists of 100 units (with activations –1 / 1 and a threshold of 0) symmetrically connected. Each trial it is trained on a population of randomly constructed patterns (excluding duplicates) where elements are set to –1 or 1 with equal probability. Patterns to be learned are presented once in a single epoch of training. Weights are adjusted using basic Hebbian learning: $\Delta w_{ij} = \eta\, a_i\, a_j$ where a_i and a_j are the activations of units i and j. The learning constant η is set to 1.0. As usual for –1 / 1 activations, the exact inverse of a pattern is counted as an instance of the pattern.

"Spurious" shows how many distinct spurious stable states are found by the 1000 random inputs. The values shown are averages for 20 replications of each simulation.

As the network consists of 100 units, we expect performance to degrade for populations with more than (roughly) 14 training patterns. This can be observed in the data shown. For populations of size 14, on average about 12 patterns are stable after training, and most of these are found when sampling the network. For larger populations the proportion of patterns that are stable decreases rapidly. For populations of size 30, less than 2 trained patterns are stable, and these are hardly ever found by sampling. In short, this kind of network does not perform well with respect to ongoing learning. As more and more items are learned, the memory system completely collapses, and almost all learned items are lost.

This problem has been addressed in the literature by a method known as "unlearning", first proposed in [10, 33], and extended in other studies such as [9, 67, 68]. Unlearning rests on the assumption that the collapse of memory, as described above, is caused by a build-up of unwanted attractors that "obscure" or "overwrite" the learned attractor states. The intention is to find and remove these unwanted attractors. The method by which this is accomplished is to randomly sample the network as described above, but then to *unlearn* (learn with a small negative learning constant) the stable states found.

Unlearning can be implemented in various ways. To generate the results shown in Table 1 the training procedure described above was extended as follows. While learning a given population, after each item has been learned (1 presentation of Hebbian learning, with a learning constant of 1.0 as usual) 50 unlearning episodes are applied. Each episode consists of iterating a random input until a stable state is reached, and then

applying Hebbian learning with a learning constant of –0.01. In other words, random attractor states are found and (slightly) unlearned.

As shown in the Unlearning columns in the figure, this modification to learning has a significantly beneficial effect[9]. As the population size grows larger a more or less constant number of trained patterns remain stable[10]. There is a strong tendency for the more recently learned items to be the ones which are retained (perhaps because the older items have had more time to be unlearned), with the most recent item being found most often. In short, when extended with unlearning, this kind of Hopfield network deals with ongoing learning by retaining the most recently learned items.

We have not explored the use of rehearsal / pseudorehearsal based methods in this kind of network. As noted above, a second presentation of all previously learned items (full rehearsal) simply doubles the magnitude of the weights, without changing the behaviour of the system. Thus rehearsing old items would very quickly cause those items to dominate the weights, and lock out any further learning.

5.2 Forgetting with error correcting learning

The standard Hopfield network described above has a low capacity. A higher capacity can be achieved by varying the architecture and the learning algorithm employed. If we allow asymmetric connections, the theoretical maximum capacity of such a network is 2*N*

[9] One of the problems with Hebbian learning is that weights can grow too large and therefore dominate. Unlearning acts to reduce the size of weights. In fact, the results achieved for unlearning are very similar to the results achieved for simple weight decay [9, 59].

[10] Note that, for larger populations, unlearning does not appear (when compared with standard learning) to reduce the overall number of spurious states in the network (some may be removed, others may be created). But it does significantly reduce the "damage" that the resulting spurious states do to the learned states.

(arbitrary) patterns [25][11]. The most effective learning algorithm in this case is the delta rule [26][12]. In [58] we explored the efficacy of rehearsal and pseudorehearsal using a Hopfield network of this type.

The delta rule, like other error correcting learning algorithms, trains using the input population concurrently over a number of epochs. We can therefore (as for the back propagation simulations) use the network to learn a base population followed by a sequence of new items. Figure 4 summarises results from simulations in [58][13]. In this case the error measure is the number of base population items which are unstable when presented as inputs (i.e. have been "forgotten"). After the base population is learned, the standard condition shows what happens as a sequence of new items are presented. After learning a single new item (on average) 13 base population items have become unstable, and by the time all 20 new items are learned (on average) 39 of the 44 base population items are unstable. In short, the network experiences typical CF.

As is the case in back propagation type networks, rehearsal protects the base population from forgetting. Rehearsing the whole population with each new item (rehearse 100% condition in Figure 4) shows that perfect performance is possible. Rehearsing half the population (rehearse 50%) is only moderately effective.

---

[11] In [29] the authors briefly review the impact of asymmetric connections, noting that asymmetry effectively introduces some "noise", and may slow down the approach to an attractor. If the asymmetry is "systematic or very strong" then it can produce limit cycles or chaotic behaviour.

[12] The delta rule is an error correcting algorithm which adjusts weights on the basis of the difference between actual and desired / teacher outputs. It underlies the back propagation / generalised delta rule which is used in multi layer nets.

[13] The Hopfield type network consists of 64 units with activations of 0 / 1 and a threshold of 0. Input patterns consist of an alphanumeric character set, and are learned (with noise) using the thermal perceptron

Pseudorehearsal must be implemented somewhat differently in this kind of dynamical network. Instead of creating pseudoitems by feeding a random input forward and recording the associated output (which fits with the function approximation interpretation of back propagation type networks), we iterate a random input to a stable state, and use that stable state as a pseudoitem (which fits the attractor state space interpretation of Hopfield type networks). Relearning a population (of 256) pseudoitems with each new item, as shown in the pseudorehearse condition, provides good protection for the base population[14]. Creating pseudoitems in this way means that the pseudoitem population (randomly chosen attractor states) will naturally include actual learned base population items (which are significant attractors in the network). Pseudorehearsal continues to work, however, even when (although there is no reason to do this) actual base population items are deliberately excluded. In other words, even relearning attractors which are usually regarded as spurious (or "crosstalk") can protect actual learned attractors from the damage caused by new learning [58]. Note however, that Meeter [42] shows that in some circumstances the rehearsal process can become dominated by "runaway consolidation", where a small number of attractors come to dominate the rehearsal process. Meeter also describes and evaluates possible solutions to this problem.

---

delta rule variant [15]. The base population consists of 44 items, and a further 20 new items are learned in sequence. Results are averages over 50 replications.

[14] Note that in this simulation we do not generate new pseudoitems after learning each new item, so previously learned new items suffer some forgetting. See [58] for details.

5.3 Discussion

It is interesting to note that, albeit in different network variants and in slightly different ways, both the unlearning and the relearning of randomly selected attractors can have beneficial effects on memory. It is, however, difficult to directly compare the efficacy of the unlearning and the rehearsal / pseudorehearsal based methods. Unlearning is used in networks with symmetric connections and Hebbian learning, where rehearsal is not effective. Rehearsal / pseudorehearsal has been explored in much higher capacity networks with asymmetric connections and delta rule learning where, similarly, we have not found unlearning to be effective. The methods can be compared, however, in terms of their plausibility for cognitive modelling. In [59] we argue that the pseudorehearsal based account is significantly more plausible in terms of its cognitive predictions, neurological support, and evidence from an evolutionary perspective relating to the stabilisation of memory circuits.

Note that it seems that both approaches could be improved by being able to distinguish between learned and spurious attractors. In the case of the unlearning approach, unlearning could be restricted to spurious attractors, and in the case of pseudorehearsal, relearning could be restricted to real / learned attractors. Robins & McCallum [60] describe such a method, which reliably distinguishes between learned and spurious attractors for a wide range of network types. We are currently exploring the use of this information in unlearning and pseudorehearsal.

6.0 Summary and discussion

The ongoing integration of newly learned information into old is a serious problem for most ANN learning algorithms. Because information is encoded using shared weights, and new learning changes the weights, previously learned information tends to be catastrophically forgotten. In short, with respect to the stability vs. plasticity dilemma, typical ANNs suffer from too little of the former, and too much of the latter.

Not all ANNs are effected. Those that store information using separate structures (localist units, or newly added units) are less vulnerable. One approach to reducing CF in standard fully distributed nets, in particular in back propagation types, is based on adapting this principle of separation. These methods attempt to decrease the overlap between the hidden unit representations of inputs, so as to reduce the interference between them. This approach provides some protection from forgetting (often as measured by the ability to relearn performance on the base population more quickly), but can also result in some undesirable consequences (such as reduced generalisation).

An obvious method of protecting old information is to rehearse / relearn it as new information is learned. Rehearsal regimes have been explored by various authors [46, 48 51]. Robins [51] found the "sweep" regime to be very effective, suggesting that rehearsal needs to be "broad" (involve many old items), but it does not need to be "deep" (rehearse each old item extensively).

While rehearsal is an effective working solution, it does require having old items stored available to be rehearsed, which in some cases may not be practical, and is in any case a completely implausible framework for cognitive modelling. Pseudorehearsal is a

method which allows us to achieve most of the benefits of rehearsal, even when we do not have the actual old items available. It works by randomly sampling the trained network to generate pseudoitems, an "approximation" of the actual old items (or a "map" of the function learned by the network), and using the pseudoitems in place of the real old items in a rehearsal process. This has the effect of "fixing" the learned function in place everywhere except where it is being "pulled" by the constant error associated with the new item. In other words, the essence of pseudorehearsal is to restrict changes to be *local* to the new item. Comparing this approach to other local learning schemes, we see that, by using training error to make changes that are local in the output function (rather than using mechanisms that involve localisation in the training set or architecture) we are able to employ the full flexibility and power of the learning algorithm.

Very similar issues arise in the context of Hopfield type networks, which also suffer from various forgetting effects. In standard networks based on Hebbian learning, ongoing learning quickly leads to overloading and CF. If learning is mixed with an unlearning mechanism, then a number of recently learned items are able to be retained. In Hopfield type networks based on delta rule learning, a version of pseudorehearsal is able to maintain good performance on old information as new information is added. The principle is the same as in the back propagation network case, relearning random approximations of old information during subsequent learning is effective at protecting the real old information itself.

Considering the disadvantages of pseudorehearsal, in back propagation type networks these methods are much less effective in networks which generalise poorly, i.e. networks which for some reason (such as using too many hidden units) fit a "noisy" function to the

learned items. A problem with our current implementation of pseudorehearsal in Hopfield type networks is that new items which are learned early in the sequence are not well retained. In both kinds of network using pseudorehearsal involves some kind of extra overhead (such as for example the use of dual weights as described in Section 4.2), and slower learning (as pseudoitems are learned along with each new item).

Do pseudorehearsal type processes have any relevance to modeling cognitive processes or the brain? CF is a very general consequence of a plastic representing medium such as the weights of a neural network. The synapses of a brain are also plastic. McClelland *et al*. [39] and Robins [51, 52] have both suggested that during its evolution the mammalian brain may well have encountered the CF problem. In their influential paper McClelland *et al*. proposed that the solution which developed is based on complementary learning systems, a fast hippocampal system for learning new information, and a slower cortical system that acts as a long term memory store. Newly learned information is transferred from the hippocampus to the cortex, in their simulation by using a full rehearsal of all old information. In a different approach Robins [52] explored the links between CF and the popular theory that newly learned information is consolidated in the brain during sleep. Note that sleep involves both a random stimulation of the cortex, and a "dialog" between cortex and hippocampus. Robins suggested that sleep based consolidation in the brain, and pseudorehearsal in ANNs, are both solutions to the same underlying CF problem, and are functionally equivalent mechanisms. These accounts [39, 52] can clearly be combined, with a sleep / pseudorehearsal type process occurring in the context of hippocampal and cortical

learning systems. Evidence relating to consolidation during sleep, and supporting this combined model, is reviewed in [52, 59].

In future research we hope to explore a number of open questions. Using back propagation type networks, are there more effective methods for generating pseudoitems, and for utilising them during pseudorehearsal? What are the predictions made by both the dual network and the dual weight implementations of pseudorehearsal, and can these be tested? Much more work needs to be done in the context of Hopfield type networks. Can the performance of pseudorehearsal be improved? What is the nature of the attractor spaces that are created during learning? Given a method for distinguishing between learned and spurious attractors [60], can we improve the performance of both the unlearning and the pseudorehearsal methods? With respect to links with the psychological literature, does the developing experimental evidence, and a further examination of the evolution of learning mechanisms (see discussion in [59]), continue to support a pseudorehearsal account? All of these are small aspects of the interesting underlying question, namely, what is the nature of the relationship between learning in ANNs and learning in the brain?

Acknowledgments

Parts of this paper were written while the author was on study leave. Many thanks to Bob French at the University of Liège (Belgium) for hosting my visit, and for many fascinating conversations on a wide range of topics! Thanks also to my colleagues Simon McCallum and Marcus Frean, who have collaborated with me on much of the research summarised in this review.

References

[1] B. Ans & S. Rousset, Avoiding catastrophic forgetting by coupling two reverberating neural networks, *C. R. Acad. Sci. Paris, Life Sciences, 320* (1997), 989 - 997.

[2] B. Ans & S. Rousset, Neural Networks With a Self-refreshing Memory: Knowledge transfer in Sequential Learning Tasks without Catastrophic Forgetting, *Connection Science, 12* (2000), 1 - 19.

[3] B. Ans, S. Rousset, R. M. French & S. Musca, Preventing catastrophic interference in multiple-sequence learning using coupled reverberating Elman networks, *Proceedings of the 24th Annual Conference of the Cognitive Science Society*, Earlbaum, NJ, 2002, pp. 71 - 76.

[4] B. Ans, S. Rousset, S. Musca, & R. M. French, Self-refreshing memory in artificial neural networks: Learning temporal sequences without catastrophic forgetting, Submitted to *Cognitive Science* (2003).

[5] N. Burgess, J. L. Shapiro, & M. A. Moore, Neural network models of list learning, *Network, 2* (1991), 399 - 422.

[6] L. Bottu & V. Vapnik, Local Learning Algorithms, *Neural Computation, 4* (1992), 888 – 900.

[7] G. Carpenter & S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics, and Image Processing, 37* (1987), 54 - 115.

[8] G. Carpenter & S. Grossberg, The ART of adaptive pattern recognition by a self-organizing neural network, *Computer, 21* (1988), 77 - 88.

[9] G. A. Christos, Investigation of the Crick-Mitchison reverse-learning dream sleep hypothesis in a dynamical setting, *Neural Networks, 9* (1996), 427 - 434.

[10] F. Crick, & G. Mitchison, The function of dream sleep, *Nature, 304* (1983), 111 - 114.

[11] R. Duda & P. Hart, *Pattern Classification and Scene Analysis,* Wiley, New York, 1973.

[12] T. Eastman, Catastrophic Interference in Constructive Neural Networks, MSc Thesis (in progress), Otago University, New Zealand, 2003.

[13] S. Fahlman & C. Lebiere, The cascade-correlation learning architecture, in: D. Touretzky, ed., *Advances in Neural Information Processing Systems II*, Morgan Kauffman, San Mateo, 1990, pp. 524 - 532.

[14] J. A. Feldman & D. H. Ballard, Connectionist models and their properties, *Cognitive Science, 6* (1982), 205 - 254.

[15] M. Frean, A "thermal" perceptron learning rule, *Neural Computation, 4* (1992), 946 - 957.

[16] M. Frean & A. Robins, Catastrophic forgetting in simple networks: An analysis of the pseudorehearsal solution, *Network: Computation in Neural Systems, 10* (1999), 227 – 236.

[17] R. M. French, Semi-distributed representations and catastrophic forgetting in connectionist networks, *Connection Science, 4 (3&4)* (1992), 365 - 377.

[18] R. M. French, Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference,

Proceedings of the 16th Annual Cognitive Science Society Conference, Earlbaum, Hillsdale NJ, 1994, pp. 335 - 340.

[19] R. M. French, Interactive connectionist networks: An approach to the 'sensitivity-stability' dilemma, *Connection Science, 9* (1997), 353 - 380.

[20] R.M. French, Selective memory loss in aphasics: An insight from pseudo-recurrent connectionist networks, in: J. Bullanaria, G. Houghton, D. Glasspool, eds., *Connectionist representations: Proceedings of the fourth annual computation and psychology workshop*. Springer Verlag, Berlin, 1997, pp. 183 - 195.

[21] R.M. French, Catastrophic forgetting in connectionist networks: causes, consequences and solutions, *Trends in Cognitive Science, 3(4)* (1999), 128 - 135.

[22] R.M. French, B. Ans & S. Rousset, Pseudopatterns and dual-network memory models: advantages and shortcomings, in: French, R. M. & Sougné , J. P. (Eds) *Connectionist Models of Learning, Development and Evolution*, Springer, Berlin, 2001, pp. 13 – 22.

[23] R. M. French & N. Chater, Using noise to compute error surfaces in connectionist networks: a novel means of reducing catastrophic forgetting, *Neural Computation, 14* (2002), 1755 - 1769.

[24] R. M. French, & D. Mareschal, Could category-specific semantic deficits reflect differences in the distribution of features within a unified semantic memory? *Proceedings of the 20th Annual Conference of the Cognitive Science Society*, Lawrence Earlbaum, New Jersey, (1998), pp. 374 – 379.

[25] E. Gardner, Maximum storage capacity in neural networks. *Europhysics Letters, 4* (1987), 257– 270.

[26] E. Gardner, N. Stroud & D. Wallace, Training with noise and the storage of correlated patterns in a neural network model, *Journal of Physics A: Mathematics and General, 22* (1989), 2019 - 2030.

[27] S. Grossberg, Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science, 11* (1987), 23 - 63.

[28] S. Haykin, *Neural Networks A Comprehensive Foundation (Second edition)*, Prentice–Hall, New Jersey, 1999.

[29] J. Hertz, A. Krough & R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City CA, 1991.

[30] P. A. Hetherington & M. S. Seidenberg, Is there "catastrophic interference" in connectionist networks? in: *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Lawrence Earlbaum, Hillsdale NJ, 1989, pp. 26 - 33.

[31] G.E. Hinton & D.C. Plaut, Using Fast Weights to Deblur Old Memories, *Proceedings of the 9th Annual Conference of the Cognitive Science Society,* Earlbaum, Hillsdale NJ, 1987, pp. 177 - 186.

[32] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences, USA 79* (1982), 2554 - 2558.

[33] J. J. Hopfield, D. I. Feinstein & R.G. Palmer, Unlearning has a stabilising effect in collective memories, *Nature, 304* (1983), 158 - 159.

[34] C.A. Kortge, Episodic memory in connectionist networks, *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, Lawrence Earlbaum, Hillsdale NJ, 1990, pp. 764 – 771.

[35] J. K. Kruschke, ALCOVE: An Exemplar-based Connectionist Model of Category Learning, *Psychological Review, 99* (1992), 22 - 44.

[36] S. Lewandowsky, Gradual unlearning and catastrophic interference: A comparison of distributed architectures, in: W. E. Hockley & S. Lewandowsky, eds., *Relating Theory and Data: Essays on Human Memory in Honour of Bennet B. Murdok*, Lawrence Earlbaum, Hillsdale NJ, 1991, pp. 445 - 476.

[37] S. Lewandowsky & S. Li, Catastrophic interference in neural networks: causes, solutions, and data, in: F. N. Dempster & C. Brainerd, eds., *New Perspectives on Interference and Inhibition in Cognition*, Academic Press, San Diego, 1995, pp. 329 - 361.

[38] S. McCallum & A. Robins Mechanisms for memory consolidation, *New Zealand Journal of Computing, 7* (1999), 13 - 20.

[39] J.L. McClelland, B.L. McNaughton & R.C. O'Reilly, 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, Vol 102 (3)* (1995), 419 - 457.

[40] M. McCloskey, & N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: G. H. Bower, ed., *The Psychology of Learning and Motivation: Volume 23*, Academic Press, New York, 1989, pp. 109 - 164.

[41] K. McRae, & P. A. Hetherington, Catastrophic interference is eliminated in pretrained networks. *Proceedings of the Fifteenth Annual Meeting of the Cognitive Science Society*, Lawrence Earlbaum, Hillsdale NJ, 1993, pp. 723 - 728.

[42] M. Meeter, Control of consolidation in neural networks: Avoiding runaway effects, Submitted to *Connection Science,* (2003).

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

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

[45] J. Murre, *Learning and Categorization in Modular Neural Networks,* Earlbaum, Hillsdale NJ, 1992.

[46] J. Murre, The effects of pattern presentation on interference in backpropagation networks, in: *Proceedings of the 14th Annual Conference of the Cognitive Science Society*, Earlbaum, Hillsdale NJ, 1992, pp. 54 - 59.

[47] J. P. Nadal, G. Toulouse, J. P. Changeux & S. Dehaene, Networks of formal neurons and memory palimpsets, *Europhysics Letters, 1* (1986), 535 - 542.

[48] R. Ratcliff, Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions, *Psychological Review,* 97 (1990), 285 - 308.

[49] R. Ratcliff, S. Clark & R. Shiffrin, The list–strength effect: Data and discussion, *Journal of Experimental Psychology: Learning, Memory and Cognition, 16* (1990), 163 - 178.

[50] R. Ratcliff & B. Murdock, Retrieval processes in recognition memory, *Psychological Review, 83* (1976), 190 - 214.

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

[52] A. Robins, Consolidation in neural networks and in the sleeping brain, *Connection Science, 8* (1996), 259 - 275.

[53] A. Robins, Maintaining stability during new learning in neural networks, in: *Proceedings of the IEEE International Conference on Systems, Man, & Cybernetics (SMC97)*, IEEE Computer Society Press, Los Alamos, 1997, pp. 3013 - 3018.

[54] A. Robins, Towards a framework for consolidation and transfer effects in neural network models, *Dynamical cognitive science: Proceedings of the Fourth Australasian Cognitive Science Conference*, CD-ROM, University of Newcastle, Australia, 1997.

[55] A. Robins, Solutions to the catastrophic forgetting problem, *Proceedings of The 20th Annual Conference of the Cognitive Science Society,* Lawrence Earlbaum, New Jersey, 1998, pp. 899 - 904

[56] A. Robins & M. Frean, Local learning algorithms for sequential tasks in neural networks, *Advanced Computational Intelligence, 2* (1998), 221 - 227.

[57] A. Robins & M. Frean, Learning and generalisation in a stable network, *Progress in Connectionist-based Information Systems: Proceedings of the 1997 Conference on Neural Information Processing and Intelligent Information Systems*, Springer-Verlag, Singapore, 1998, pp. 314 - 317.

[58] A. Robins & S. McCallum, Catastrophic forgetting and the pseudorehearsal solution in Hopfield type networks, *Connection Science, 10* (1998), 121 - 135.

[59] A. Robins & S. McCallum, The consolidation of learning during sleep: Comparing the pseudorehearsal and unlearning accounts, *Neural Networks, 12* (1999), 1191 - 1206.

[60] A. Robins & S. McCallum, A robust method for distinguishing between learned and spurious attractors, Submitted to *Neural Networks* (2003).

[61] N.E. Sharkey & A.J.C. Sharkey, Understanding Catastrophic Interference in Neural Nets, Technical Report CS-94-4, Department of Computer Science, University of Sheffield U.K., 1994.

[62] N.E. Sharkey & A.J.C. Sharkey, Interference and Discrimination in Neural Net Memory, in: Levy, J., Bairaktaris, D., Bullinaria, J. & Cairns, P., eds, *Connectionist Models of Memory and Language*, UCL Press, London, 1994.

[63] N. Sharkey & A. Sharkey, An analysis of catastrophic interference. *Connection Science, 7* (1995), 301 - 329.

[64] R. Shiffrin, R. Ratcliff & S. Clark, The list–strength effect II: Theoretical mechanisms, *Journal of Experimental Psychology: Learning, Memory and Cognition, 16* (1990), 179 - 195.

[65] D. Silver, Selective transfer of neural network task knowledge, PhD thesis, University of Western Ontario, 2000.

[66] D. Silver & R. Mercer, *The task rehearsal method of sequential learning*, Technical Report #517, Department of Computer Science, University of Western Ontario, 1998.

[67] J. van Hemmen, Hebbian learning, its correlation catastrophe, and unlearning, *Network, 8* (1997), V1 - V17.

[68] S. Wimbauer, N. Klemmer, & J. van Hemmen, Universality of unlearning, *Neural Networks, 7* (1994), 261 - 270.

	Standard			Unlearning		
Trained	Stable	Found	Spurious	Stable	Found	Spurious
10	9.8	9.75	117.95	9.1	8.25	42.7
11	9.85	9.85	95.05	9.7	8.95	51.05
12	10.85	10.65	127.1	10.25	9.4	56.3
13	11.05	10.75	123.8	11.35	10.2	69.1
14	11.9	11.3	146.1	11.7	10	79.25
15	11.05	9.2	153.7	11.95	9.7	104.25
16	9.95	7	150.8	12.25	10.25	121.1
17	9.6	6.15	171.9	12.55	10	136.15
18	9.35	4.9	211.6	13.05	10.2	167.65
19	7.35	2.4	173.65	13.7	9.45	177.25
20	8.25	1.7	217.6	14.35	9.9	200.3
21	6.75	1.3	199.75	13.25	8.9	231.8
22	6.9	1.2	245.15	14.15	9.1	257.85
23	4.9	0.65	226.5	14.15	8.4	270.15
24	5.5	0.35	261.05	13.7	7.6	252.75
25	4.316	0	250.053	13.1	6.8	260.3
26	3.316	0	269.526	12.95	7.4	301
27	2.722	0.333	270.056	14	7.1	319.05
28	2.5	0	329.111	12.55	6.85	332.7
29	1.8	0.067	298.733	12.3	6.55	336.8
30	1.688	0.062	387.938	12.45	6.6	363.2

Table 1: Standard Hebbian learning and Hebbian learning with unlearning.

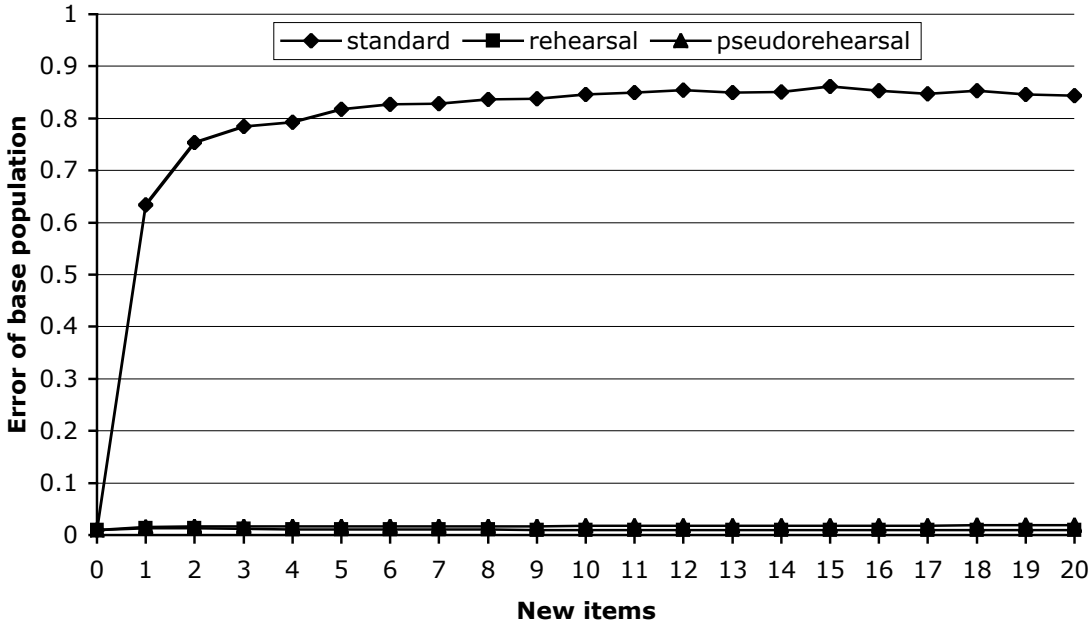

Figure 1. The standard condition illustrates the basic catastrophic forgetting effect. A base population is learned, and prior to any new learning ("0" on the *x* axis) has an error of less than 0.01. As each of a sequence of new items is learned the error of the base population increases. The rehearsal and pseudorehearsal conditions (see Section 3) show that these mechanisms eliminate the forgetting effect.

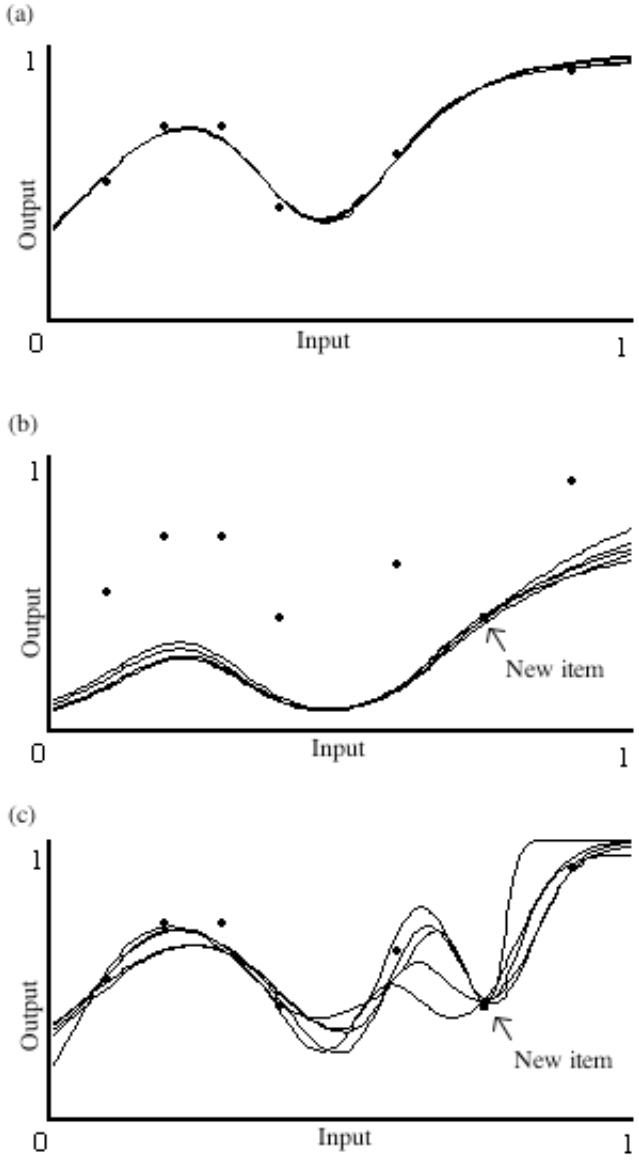

Figure 2. Functions learned by a network (five replications). Part (a) shows the function learned using six training items / data points. Parts (b) and (c) show the subsequent learning of a single new item using standard back-propagation learning and using pseudorehearsal respectively. (Adapted from [57]).

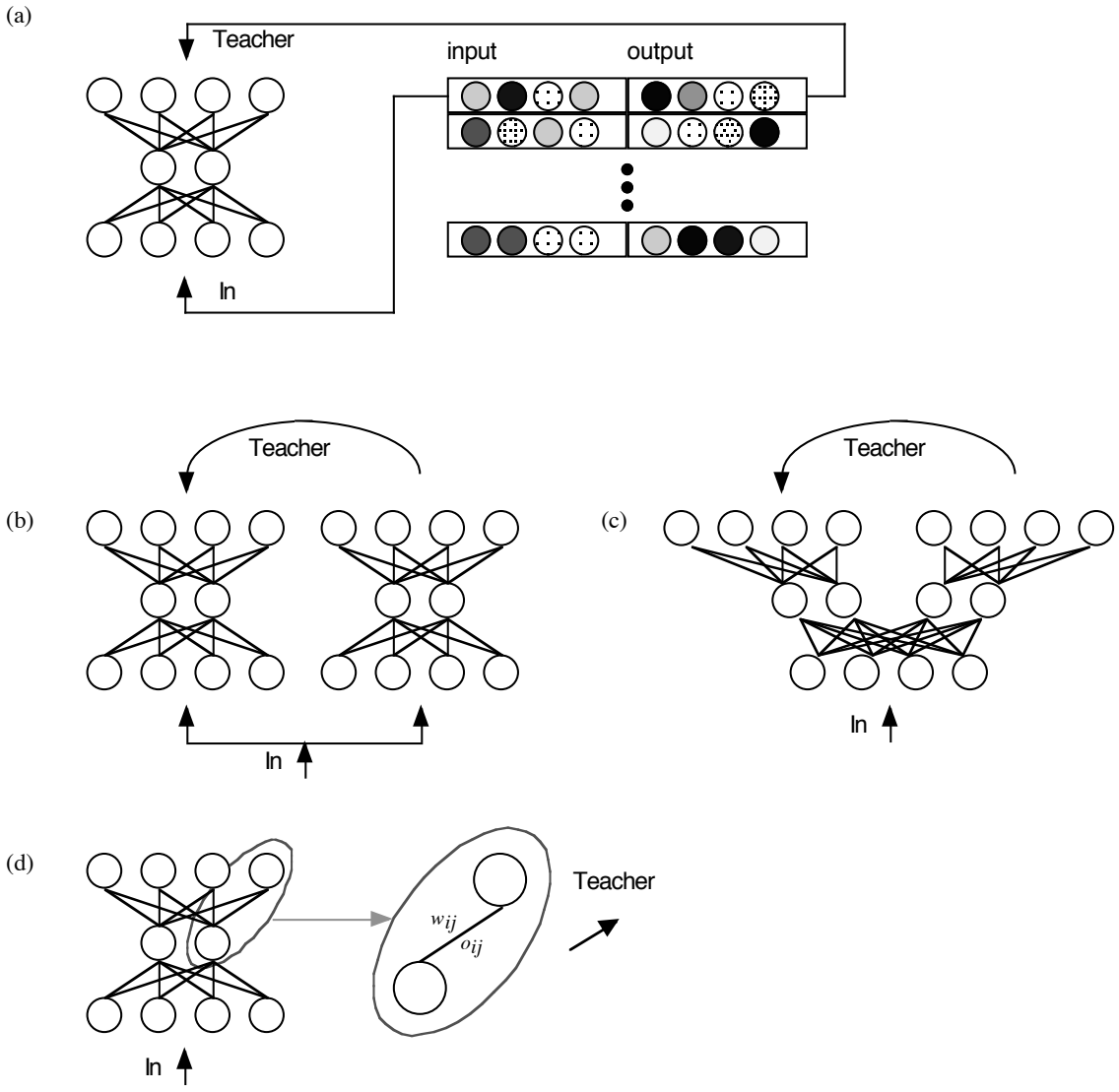

Figure 3. Implementing pseudorehearsal. Part (a) shows basic pseudorehearsal with a separate pseudoitem store. During new learning input pseudoitems are presented to the network along with the corresponding output pseudoitem which is used as a teaching signal. Parts (b) and (c) show variants of the dual network approach. Each network is presented the same random input and the pseudoitem output of one network is used as teaching signal for the other. Part (d) shows the dual weight approach. Here the network is presented a random input and generates an output using one weight, and a pseudoitem teaching signal using a separate weight on each connection.

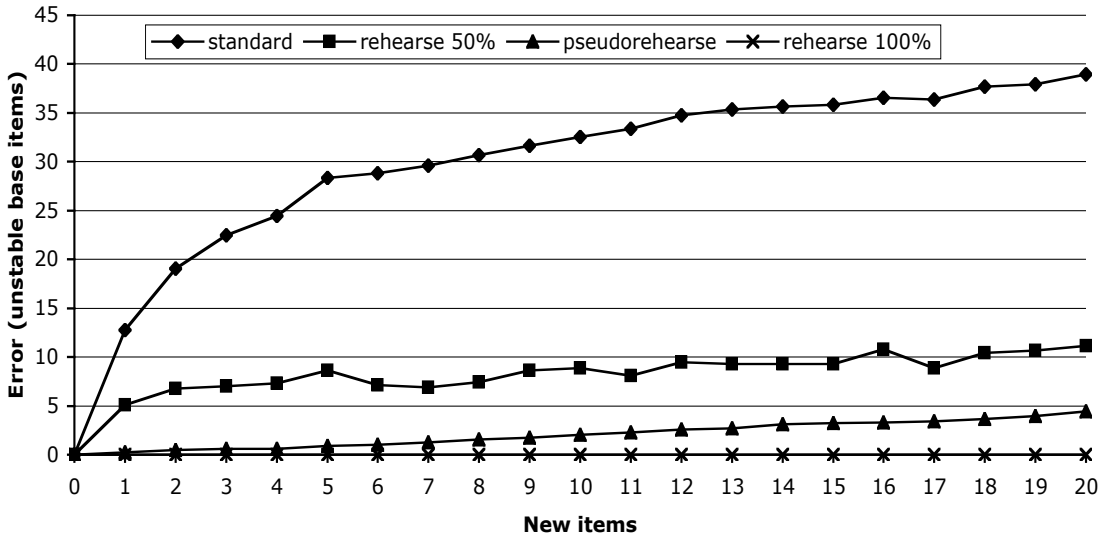

Figure 4. The standard condition illustrates basic catastrophic forgetting. A base population is learned, and prior to any new learning ("0" on the x axis) all items are stable. As each of a sequence of new items is learned the error (number of unstable base population items) increases. The two rehearsal, and the pseudorehearsal conditions, show that these mechanisms reduce the forgetting effect.