A framework to deal with interference in connectionist systems

Vicente Ruiz de Angulo, Carme Torras
Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Edifici NEXUS
Gran Capità 2-4
08034-Barcelona, Spain
E-mail: {ruiz,torras}@iri.upc.es

We analyze the conditions under which a memory system is prone to interference between new and old items. Essentially, these are the distributedness of the representation and the lack of retraining. Both are, however, desirable features providing compactness and speed. Thus, a two-stage framework to palliate interference in this type of systems is proposed based on exploiting the information available at each moment. The two stages are separated by the instant at which a new item becomes known: a) interference prevention, prior to that instant, consists in preparing the system to minimize the impact of learning new items and b) retroactive interference minimization, posterior to that instant, seeks to learn the new item while minimizing the damages inflicted on the old items. The subproblems addressed at the two stages are stated rigorously and possible methods to solve each of them are presented.

Keywords: interference, catastrophic forgetting, associative memory, neural networks

1. The interference problem

Suppose one would like to learn a set of input-output pairs $\{(X_i, D_i)\}_{i=1...N}$, where D_i is the desired output of the system to input X_i. In some applications not all the items to be learned are known at the same time. Instead, there is a sequential order of arrival, and the system must be operative before the last item is known with, desirably, the best possible performance. Therefore, new items must be quickly learned and integrated. However, when new memories are introduced in an associative system, the performance of the already stored items can be seriously degraded. This performance degradation is usually called *interference* or *catastrophic forgetting*. The effect of adding a new item depends basically on two factors: the type of memory used and the training scheme applied.

- Two types of **representations** can be distinguished:
  - Local representations, where the items are stored in such a way that they do not interfere with one other, like in a look-up table. As a consequence, in normal conditions, there is a perfect recall of both the old and the new items. However, when the system is full, there is no possibility of storing new items without destroying completely some old items.
  - Distributed representations, in which each element or parameter of the associative memory intervenes in the representation of several items. And reciprocally, each item is represented by several parameters, each of which supports only partially the storage of the item. When a new item is introduced, the recall of several stored items is degraded. The importance of forgetting raises very gradually with the number of new items learned. Distributed representations exploit memory resources much better than local ones, and are also reputed to interpolate more adequately (see Section 3.2 for details about this).
- There are two basic **training schemes**:
  - To introduce the new item isolately. In this case, the last item is quickly learned but, in many cases, the structure of the system is such that this learning causes a very important degradation of the previously learned items. That is, catastrophic interference appears.

* To train the new item jointly with the old ones. The problem is that with some kinds of systems, especially those having distributed representations, this can be a complex optimization process that requires lots of computation.

In the above account of types of memories and training schemes, we have only described the extreme possibilities. There are others in between representing a compromise between their properties.

2. Previous work

The phenomenon of interference has been noted and studied by the connectionist community: Ratcliff [14] and MacCloskey and Cohen [9] first unveiled the problem. Their purpose was to model by means of artificial neural networks some aspects of human learning and memory, such as forgetting and retroactive interference. They arrived to the conclusion that there is too much and too sudden interference in neural networks to be a valid model. Almost all the following papers on interference inherit this marked psychological character. We briefly comment on some of them, to next focus on the engineering aspects of the problem.

The different works will be placed in the framework outlined in the preceding section, namely an imaginary plane whose axes refer to the locality of the representation and the extent of retraining with old items (Figure 1).

Thus, French's approach [4] is situated at the extreme of localized solutions. He uses sigmoidal units (see the next section) with a $[0, 1]$ range, saturating them and favoring the zero states.

Hetherington and Seidenberg [5] observed that, although the old items could have large errors after the introduction of a new item, they could be quickly relearned. Thus, they suggest to retrain new and old items altogether after the introduction of the new pattern. This can be considered an approach far from the origin in the retraining axis, in which information about old items is used in a delayed mode.

In [15] the new item is trained with a (fixed or, preferably, randomly changing) *part* of the previously trained items. Thus, it can be situated in the middle of the retraining axis. In a variation of this strategy, he essays to produce the same result without requiring the availability of the old items; the associations that are jointly trained with the new items are not old items, but (input, network output) pairs, which we denote $(X, F(X))$, randomly extracted from the network before introducing the new item. Although in appearance similar to the former, this is a different approach (that Robins calls pseudorehearsal). Training with the old items constrains the shape of $F(X)$ much less than pseudorehearsal, because the latter implicitly aims to reproduce the previous $F(X)$ function, except at the discontinuous point where the new item is located. The repeated introduction of items through this process leads to very local representations. It must be said that pseudorehearsal has some intriguing points in common with brain processes during dreaming [16], which suggests that the brain could use some related strategy, as commented below.

McClelland et al. [8] claim that the brain avoids catastrophic interference by using a rapid -local- memory to store new items, which afterwards should be integrated slowly in the long-term, distributed memory[1]. This approach can be considered a combination of a temporal local solution and delayed retraining. To be successful it needs a method to discriminate when the output for an input must be generated using the local or the distributed memory. Other requirements are the availability of the old items, and free time to train the distributed memory with them and those stored in the local memory. Following McClelland et al., the cortex is the long-term memory in the human brain, and the hippocampus is the local memory, able also to recognize when a pattern is new. Dreaming could be, as suggested by Robins [16], the time for integration, and the information about the old patterns would be extracted by exciting the cortex randomly, in a way similar to the pseudorehearsal technique commented above.

3. A closer look at the interference problem

3.1. Associative systems and neural networks

At this point it is convenient to enter into details about the kind of systems that will be dealt with

[1] Note that this requires to know whether an association has been introduced recently in the quick memory.

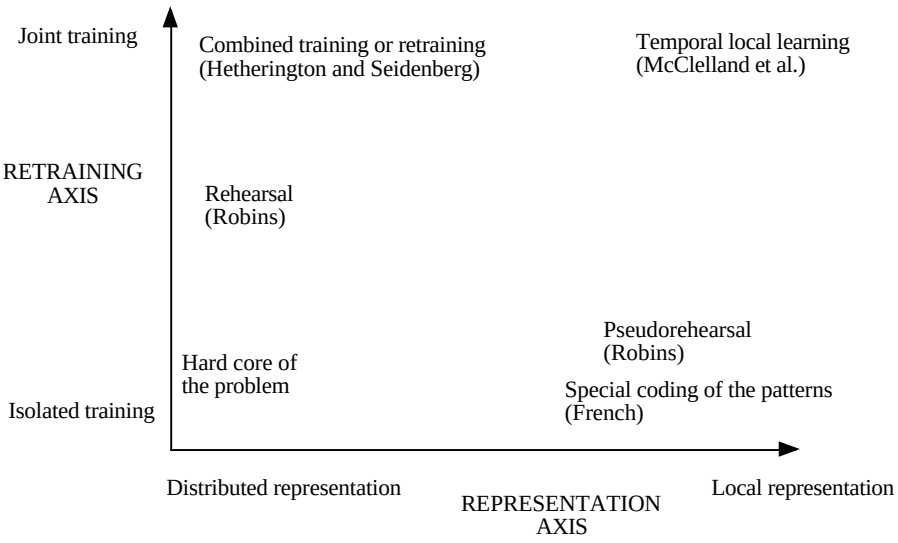

Fig. 1. Imaginary plane on which the different approaches to deal with catastrophic interference can be placed.

in this paper. Although we will try to keep the discussion at a rather general level, our results will be centered on neural networks.

First of all, we require the associative memory system to be *able to generalize*. This means that for any input X the system must yield a response or output $F(X;W)$ parametrized by a vector of parameters W. To store a set of items $\{(X_i, D_i)\}_{i=1...N}$, the system must minimize some function $E(W)$ enforcing the similarity of $F(X_i;W)$ and D_i, as for example $E(W) = \sum_{i=1...N} E_i(W) = \sum_{i=1...N}(F(X_i;W)-D_i)^2$. The exact shape of this function influences the way in which the network responds to inputs outside the training set and, for this reason, $E(W)$ could include terms aimed at determining the way the system should generalize.

Another condition we impose is the derivability of $F(X;W)$ and $E(W)$, this allowing the minimization of $E(W)$ with standard methods such as gradient descent using its derivatives.

The usual models of multilayer neural networks satisfy the above specifications. For example, a typical regression[2] two-layer neural network has the form

$$F_i(X;W) = \sum_j w_{ij} y_j(X), \tag{1}$$

where F_i is the ith component of F, w_{ij} are modifiable parameters and y_j is a nonlinear function of the input that, in the terminology of neural networks, is known as the output of hidden unit j. Depending on the shape of the y_j's, feedforward neural networks can be classified into two types. The most frequent one uses

$$y_j(X) = sig(\sum_k v_{jk} X_k), \tag{2}$$

where X_k is the kth component of X, v_{jk}'s are modifiable parameters (also components of W) and sig is a function of sigmoidal shape, such as the hyperbolic tangent. The other type of networks is the Radial Basis Function (RBF) networks, whose y_j's are RBF units, typically gaussians:

$$y_j(X) = e^{-\frac{\sum_k (x_k - v_{jk})^2}{\sigma_j^2}}, \tag{3}$$

where v_{jk} and σ_j^2 are modifiable parameters (included in W).

3.2. *Distributed representations*

Up to now we have associated catastrophic forgetting with the existence of crossed dependencies of different outputs of the system on many parameters. This corresponds to systems with distributed representations, which seem to be at the heart of the problem. It is convenient to take a closer look at this relation and at the arguments against renouncing to distributed representations.

[2] By regression network we mean one whose output is a vector of real numbers.

The term "distributed representations" is often used in the connectionist literature ambiguously or with different meanings. Here, to avoid misunderstandings, we put forth a precise definition. We say that the response of the system to input X has *a distributed representation* if the majority of the derivatives $\frac{\partial F}{\partial w_i}(X;W)$ have a significant magnitude. In the opposite case, we say that it has a local representation. Note that the different items stored in a system may have representations with very different degrees of distributedness. Moreover, an arbitrary input which has not yet been associated to a desired output, but producing anyway an output response, can be said also to have a local or a distributed representation. The above definition is adequate to analyze the interference problem, while at the same time bearing resemblance to the common conception of distributed representations.

The easiest way to introduce a new item X_{new} is to change those parameters to which the current (erroneous) system response is more sensitive. If there is a great overlap between this set of parameters and those having a large $\frac{\partial F}{\partial w_i}(X_p;W)$ magnitude, the response of item p will be seriously perturbed (in a way approximately proportional to $\frac{\partial F}{\partial w_i}(X_p;W)\frac{\partial F}{\partial w_i}(X_{new};W)$). Thus, catastrophic forgetting appears when a) the stored items have distributed representations and b) the response of the system to the new item prior to learning has also a distributed representation.

Sigmoidal networks, due to the euclidean products in (1) and (2), tend to produce very distributed representations for all their possible inputs, unless the sigmoidal functions work in their saturated zones.

Instead, RBF networks can have distributed or local representations, depending on the relation between the distance among the RBF centers and their widths.

Then, why not simply use a very local representation of the memories to avoid interference? There are two main reasons for not doing this:

- Compactness. It is clear that local representations using an exclusive set of parameters for each item will require many resources to deal with large quantities of data. Indeed, purely local representations must always have at least as many parameters as data stored. Instead, when data are introduced by minimizing an error function without any locality constraint, the result is naturally distributed. With high data-to-parameter ratios, there are not local representations producing equivalent results.
- Generalization. To determine the output of an arbitrary input X, any good generalizer should take into account multiple stored items, especially in stochastic or noisy domains. When data are scarce and sparse, items far from X should influence the system response. This influence can be only realized through commonly dependent parameters. The sharing of parameters by several items implies, at least, a certain degree of distributedness in the representations. Note that in the sense we are using the term distributed, RBF networks composed of units with a limited "local" activation function can exhibit distributed representations, the determinant factor being the degree of overlap between the units.

3.3. Our goal

In this paper we intend to deal with the engineering aspects of the core problem. This core is for us at the origin of the retraining and representation axes (refer to Figure 1), i.e., catastrophic interference avoidance without requiring repeated presentations of the old items, and without explicitly imposing local representations. We coincide with [13] in pursuing this goal and in taking a pure engineering perspective.

However, theoretical reasons forbid a perfect solution to this problem. When a feed-forward network with a fixed number of units has enough capacity to encode a fixed set of items, there is a bound on how fast learning can take place, since this problem has been proven to be NP-complete [3] [6]. Therefore, we cannot aim at finding a procedure that in approximately constant time learns a new item while leaving the old ones intact, because by iterating this procedure learning could be carried out in time linear in the number of items. As a consequence, our goal must be limited to try to palliate interference as much as possible.

4. A two-stage approach to palliate interference

We consider that the interference problem must be handled in two stages separated by the arrival of the new item:

- A priori stage: encoding the old, known, items in the best way to prepare the network to minimize the impact of learning new items. We call this stage **interference prevention**.
- A posteriori stage: learning the new items while minimizing the damages inflicted on the old items stored with a given weight configuration. We call this stage **retroactive interference minimization**.

We will concrete the scenario in a set of items $1 \dots N-1$ which are stored prior to the arrival of the new item N. Let $E_{1\dots p}$ denote the error in items $1, 2 \dots p$. With this notation we formulate our problem as how to find efficiently a minimum of $E_{1\dots N}(W)$ with the constraint of having item N available in a moment posterior to the arrival of items $1 \dots N-1$. The optimization to be performed prior to this moment constitutes the interference prevention, whereas those optimizations posterior to that moment conform the retroactive interference minimization. We will enforce this division by reflecting the calculations of interference prevention in the variable W itself, fixing its value afterwards. The changes to be made to the weights during retroactive interference minimization will be reflected in the variable vector ΔW, so that the value of the weight vector after the introduction of the new item will be $W + \Delta W$. For clarity of explanation, we make all the derivations assuming a perfect encoding of the new item ($E_N(W + \Delta W) = 0$). However, the N-th item can also be introduced partially within this framework. Instead of using (X_N, D_N), we would take $(X_N, F(X_N, W) + \lambda(D_N - F(X_N, W)), 0 < \lambda < 1$ as the new item. Reducing only a fraction of the error can be useful when dealing with noisy data [21].

The minimization of $E_{1..N}(W)$ is approximately equivalent to:

$$\min_{W+\Delta W} E_{1\dots N-1}(W + \Delta W)$$
$$\textit{subject to } E_N(W + \Delta W) = 0.$$

And making the above distinction explicit:

$$\min_{W,\Delta W} [E_{1\dots N-1}(W) + \Delta E_{1\dots N-1}(\Delta W; W)]$$
$$\textit{subject to } E_N(W + \Delta W) = 0.$$

For clarity of explanation, we make all the derivations assuming a perfect encoding of the new item $E_N(\Delta W + W) = 0$. Let us abbreviate $E_{1\dots N-1}$ with E. We need to express $\Delta E(\Delta W + W)$ in a manageable way. Evaluating accurately $\Delta E(\Delta W + W)$ for a given ΔW can have a high computational cost. For example, in a supervised feedforward neural network, it would entail presenting the whole set of items to the network. If the optimum is to be found through some search process, this evaluation has to be performed repeatedly, leading to a high computational cost.

An alternative solution to accurate evaluation is to approximate the error function over the old items through a truncated Taylor series expansion, for example, with a second-order polynomial in the weights. A linear model is too rude and not feasible because it may turn the problem into an unsolvable one [21]. The coefficients of the polynomial have a direct interpretation, namely the first and second derivatives of E. Cross-terms are not included because the calculation of the Hessian requires computations very costly in memory and time. The Hessian diagonal is, therefore, the only information about old items that we require in principle.

Thus, the most faithful problem formulation we can reasonably aspire to deal with is:

$$\min_{W,\Delta W} \left[E(W) + \frac{1}{2} \sum_i \frac{\partial^2 E}{\partial W_i^2} \Delta W_i^2 + \sum_i \frac{\partial E}{\partial W_i} \Delta W_i \right]$$
$$\textit{subject to } E_N(W + \Delta W) = 0. \tag{4}$$

The problem is how to carry out this constrained minimization on W (interference prevention) and on ΔW (retroactive interference minimization). In the latter, W is by definition a constant already known, and the minimization takes place over ΔW. Interference prevention is more complex because the minimization must be performed over W, but the cost function is parametrized by ΔW, which is unknown at that moment. This leads to the minimization of the expected cost for the a priori distribution of ΔW. We begin by the easier one, namely retroactive interference minimization.

5. Minimizing retroactive interference

5.1. Problem formulation

Now we consider W as a constant, and therefore (4) becomes

$$\min_{\Delta W}\left[\frac{1}{2}\sum_i c_i\Delta W_i^2+\sum_i b_i\Delta W_i\right]$$
$$subject\ to\ E_N(\Delta W)=0,$$

with c_i assigned to $\frac{\partial^2 E}{\partial W_i^2}$ and b_i to $\frac{\partial E}{\partial W_i}$. Note that usually b_i will be close to zero because, by introducing the first $N-1$ items we should have minimized E and, therefore, also the absolute value of its first derivatives. In any case, this is part of interference prevention. Thus, we assume $b_i=0$, and our definitive formulation for the retroactive interference subproblem is:

$$\min_{\Delta W}\sum_i c_i\Delta W_i^2$$
$$subject\ to\ E_N(\Delta W)=0, \tag{5}$$

with $c_i=\frac{\partial^2 E}{\partial W_i^2}$.

5.2. Selection of coefficients

The above formulation is similar to that adopted by Park et al. [13], and also in previous works by the authors in [18], [20] and [21], the difference being in the value assigned to c_i. In [13] a modified second derivative of E, weighted by the error of the outputs of the network is used. In [20] a sensitivity measure dependent on the history of the first derivatives $\frac{\partial E}{\partial W_i}(t)$ and the changes made by the back-propagation algorithm $\delta W_i(t)$ in each iteration t is used,

$$c_i=\left|\frac{\sum_t \frac{\partial E}{\partial W_i}(t)\ \delta W_i(t)}{W_i^{fin}-W_i^{in}}\right|,$$

where W_i^{in} and W_i^{fin} are the initial weight and the final weight after the training, respectively. In [13], $c_i=\frac{\partial^2 E}{\partial W_i^2}$ and, $c_i=1$ are tested and compared. The latter option is interesting for its cheap computational cost, and because it is intuitively appealing, since (5) calculates in this case the nearest solution for the new item in parameter space.

We next show that the option $c_i=\frac{\partial^2 E}{\partial W_i^2}$ is the best in average to estimate $\Delta E(\Delta W)$ with a cost model of the form $\sum_i c_i\Delta W_i^2$, even outside a minimum for the old items. First, note that $\sum_i c_i\Delta W_i^2$ cannot distinguish between the effect produced by positive and negative increments of the same magnitude, i.e., the sign information is lost. The quantity $\sum_i c_i\Delta W_i^2$ could have been generated by 2^n vectors, denoted $\Delta W^{l)}$, n being the number of components of ΔW, by considering two opposite values for each component. The best one can do in this situation is to use as cost function the average of the error increments produced by these vectors:

$$\langle E(W)-E(W+\Delta W)\rangle =$$
$$=\frac{1}{2^n}\sum_{l=1}^{2^n}(E(W)-E(W+\Delta W^{l)}))=$$
$$=E(W)-\sum_{l=1}^{2^n}E(W+\Delta W^{l)})\frac{1}{2^n},$$

where $<\cdot>$ denotes expectation. Now we need to use the following result [22], [17]:

$$\int g(U+R)\ P(R)\ dR=$$
$$=g(U)+\frac{1}{2}\sum_i\frac{\partial^2 g}{\partial U_i^2}\ \sigma_i^2+\mathcal{O}(\mu_i), \tag{6}$$

where g is an arbitrary differentiable function, P is a symmetric, zero-mean probability density function, and σ_i^2 and μ_i are the variance and the fourth central moment of the marginal distribution of U_i, respectively. This formula can be particularized to discrete distributions, so that considering P_1 is the probability of $\Delta W^{l)}$ given the information about the square components, $P_1(\Delta W^{l)})=\frac{1}{2^n}$ and, therefore,

$$\langle E(W)-E(W+\Delta W)\rangle\approx$$
$$\approx E(W)-\left(E(W)+\frac{1}{2}\sum_i\frac{\partial^2 E}{\partial W_i^2}\ \sigma_i^2\right)=$$
$$=\frac{1}{2}\sum_i\frac{\partial^2 E}{\partial W_i^2}\ \frac{\Delta W_i^2+(-\Delta W_i)^2}{2},$$

and finally

$$\langle E(W)-E(W+\Delta W)\rangle\approx\frac{1}{2}\sum_i\frac{\partial^2 E}{\partial W_i^2}\ \Delta W_i^2. \tag{7}$$

Thus, we see that the absence of sign information leads naturally to a cost function of the form $\sum_i c_i\Delta W_i^2$ with the assignment $c_i=\frac{\partial^2 E}{\partial W_i^2}$, even in points far from the minimum. This result can be easily generalized to the estimation of an arbitrary function in the same conditions.

This conclusion is in agreement with the results of an experimental comparison of all the above-mentioned options for assigning c_i [20].

Another conclusion emerged from this comparison: the difference between using $c_i = 1$ and $c_i = \frac{\partial^2 E}{\partial W_i^2}$ decreases with the number of items stored in the memory system. This can be interpreted in the following way: the weights that are more changed by the solution of (5) are those with lower second derivatives. Thus, second derivatives can be considered as a measure of how much the encoding of the previously stored items are supported by the corresponding weights. Those weights with current lower derivatives change to support the new items and, then, their second derivatives grow. As the number of stored items increases, all weights tend to have similar, high-magnitude second derivatives and, thus, using $c_i = 1$ or $c_i = \frac{\partial^2 E}{\partial W_i^2}$ tends to yield the same results.

5.3. *Minimization method*

Now the question remains of how to solve (5) efficiently. A usual way to tackle a constrained optimization problem is to linearly combine the cost function with the deviation of the constraint from zero, and then minimize this new error function:

$$\min_{\Delta W} \left[\alpha \sum_i c_i \Delta W_i^2 + \beta \; E_N(\Delta W) \right]. \qquad (8)$$

In the minimization of this function, there is a tradeoff between E_N and $\sum_i c_i \Delta W_i^2$ that depends on α and β, and the error in the new item will not be 0 unless the $\frac{\beta}{\alpha}$ relation tends to infinite with time. In practice this is impossible and it is approximated through an appropriate schedule for changing α and β.

More sophisticated algorithms from the theory of constrained optimization can also be applied, as in [13], where the reduced gradient algorithm for nonlinear constrained optimization is used. The advantage of these methods is their generality, in the sense that in principle they can deal with general functional forms[3], but they can be complex and computationally expensive.

In [20] and [17], an algorithm to solve (5) that exploits the structure of multilayer neural networks is developed. The drawbacks of solving a constrained minimization problem are here avoided through the transformation of retroactive interference into an unconstrained minimization problem. The finding underlying this transformation is the existence of an one-to-one mapping between the hidden unit configurations and the best solution in the subspaces of weights that produce those hidden unit configurations. Besides allowing a non-constrained optimization, this transformation has other advantages, like a number of variables much lower than in the original problem, and an always perfect encoding of the new item. The algorithm derived from this transformation, called LMD (Learning with Minimal Degradation), is completely parallel, even among layers.

5.4. *Relation between retroactive interference minimization and pruning*

Pruning is a technique commonly used in connectionist systems consisting in trimming the network by eliminating the most superfluous weights.

To palliate retrograde interference, one of the more important issues is the determination of the less significant parameters for the encoding of a number of stored memories. These are the parameters that should support most of the necessary changes to introduce new information. Instead, pruning detects the less profited parameters to eliminate them. Indeed, we have used as relevance measure the second derivatives, also used in Optimal Brain Damage [7], the most popular pruning technique.

The relation with pruning suggests that advances in pruning techniques can be incorporated into retroactive interference minimization algorithms.

6. Interference prevention

6.1. *Problem formulation*

The interference prevention subproblem is formulated as the selection of a set of system parameters able to codify items $1 \ldots N-1$ in such a way that they are disrupted minimally when item N is introduced. Now, we have to solve (4) in W, taking ΔW as an unknown constant.

[3] But not always. For example, the reduced gradient method only works with linear constraints. Because of this, $E_N(W)$ is approximated linearly in [13].

$$\min_W \left[E(W) + \frac{1}{2} \sum_i \Delta W_i^2 \frac{\partial^2 E}{\partial W_i^2} + \sum_i \Delta W_i \frac{\partial E}{\partial W_i} \right]$$
$$\textit{subject to } E_N(W + \Delta W) = 0.$$

Before knowing the new item, we must solve this problem without assuming any particular value for ΔW. The ideal solution is that which, in average, best solves the problem, taking into consideration the distribution of ΔW:

$$\min_W \left[E(W) + \left\langle \frac{1}{2} \sum_i \Delta W_i^2 \frac{\partial^2 E}{\partial W_i^2} + \sum_i \Delta W_i \frac{\partial E}{\partial W_i} \right\rangle \right].$$

And developing the second term:

$$\min_W \left[E(W) + \int \left(\frac{1}{2} \sum_i \Delta W_i^2 \frac{\partial^2 E}{\partial W_i^2} + \sum_i \Delta W_i \frac{\partial E}{\partial W_i} \right) \mathcal{P}(\Delta W)\, d\Delta W \right], \qquad (9)$$

where $\mathcal{P}$ is the density function of ΔW. Let us assume that $\mathcal{P}(\Delta W)$ is equal for each of the components ΔW_i, or at least that they have equal variances (although this assumption can be easily relaxed). It is also very natural to assume a zero mean for this distribution: otherwise item N would not be really new and the information of the mean of $\mathcal{P}(\Delta W)$ could be used to train the system. We can apply (6) to (9) by making $g(U) = \frac{1}{2}\sum_i U_i^2 \frac{\partial^2 E}{\partial W_i^2} + \sum_i U_i \frac{\partial E}{\partial W_i}$, so that the integral in (9) becomes the expectation of $g(0 + \Delta W)$. Since $g(0)$ is null and $\frac{\partial^2 g}{\partial U_i^2}(\Delta W) = 2\frac{\partial^2 E}{\partial W_i^2}$, we get

$$\min_W \left[E(W) + \frac{\sigma^2}{2} \sum_i \frac{\partial^2 E}{\partial W_i^2} \right], \qquad (10)$$

without any error[4].

This result can be derived in another way, without supposing a particular shape for $\Delta E_{1\ldots N-1}(\Delta W; W)$. The problem of interference prevention can be understood as the search for a W such that, after being modified by the introduction of the new item, it would be still able to reproduce old ones. As the new items are by definition unknown, they produce unknown modifications in the parameters when they are learned. Thus, the problem consists in getting a point of low $E(W)$ stable with respect to random perturbations of W. This resistance to perturbations can be expressed as

$$\min_W \int E(W + \Delta W)\ \mathcal{P}(\Delta W)\ d\Delta W. \qquad (11)$$

This expression is made exactly equivalent to (10) by applying (6). Thus, reassuringly, we have obtained the same result with two different reasonings.

6.2. Selection of coefficients

Unfortunately the above discussion did not suggest which is the density function $\mathcal{P}(\Delta W)$. This amounts to decide the variance σ^2, which is the main parameter of the distribution influencing the minimization (10) or (11)[5]. The variance of the weight changes produced by future items could be approximately deduced from their expected error, which in turn may be estimated from the error that the most recently introduced items had at their arrival. However, this is a context-dependent hypothesis, which can lead to significant errors. Moreover, there is another important issue about the selection of σ^2 to be dealt with: it influences not only the error increments produced by new items, but also the way in which the non presented items are interpolated. In fact, the way in which the items $1 \ldots N-1$ are encoded determines the answer of the system to all possible inputs. The quality of these answers is often more important than exact storage of the presented items. In this case, σ^2 should be tuned in accordance to the former desideratum.

In conclusion, either because of ignorance of the appropriate value or because it is tuned for other purposes, the parameter σ^2 used in (10) could be significantly different from real weight variances. What are the consequences of this parameter imprecision? Could it have effects opposed to those desired, i.e., in these conditions, the introduction of new items can be worst after minimizing (10) than after minimizing $E(W)$? A detailed mathe-

[4] The term in (6) disregarded here implies fourth or higher-order derivatives, and these are null for $\frac{1}{2}\sum_i \Delta W_i^2 \frac{\partial^2 E}{\partial W_i^2} + \sum_i \Delta W_i \frac{\partial E}{\partial W_i}$.

[5] The terms disregarded when approximating (11) with (10) are significant in general, but are close to zero in the minimum of (10).

matical analysis of this question is carried out in [17]. The conclusions can be summarized in a few words. If σ^2 is smaller than the real variance, the minimization (10) is always beneficial. The opposite case is also safe if the remaining error $E(W)$ in the minimum of (10) is not much higher than in the minimum of $E(W)$. $E(W)$ in the minimum of (10) has a sigmoidal shape when considered as a function of σ^2 and, therefore, σ^2 can be increased until the error begins to grow quickly.

6.3. Minimization method

There is a direct way of minimizing (10) or equivalently (11). It consists in adding noise to the weights while minimizing $E(W)$, so that a sample of the gradient distribution of $\frac{\partial E}{\partial W_i}(W + \Delta W)$ is calculated in each iteration. Lots of samples of ΔW must be extracted from its distribution to get a good estimate of the average derivatives (or alternatively, very small minimization steps must be done in the direction of $\frac{\partial E}{\partial W_i}(W + \Delta W)$). This noise addition during training has been already used for other purposes, such as ameliorating fault-tolerance in neural networks [11] or improving their generalization properties [12], [1].

Again, this method is very general, it being valid for any type of $E(W)$ and $\mathcal{P}(\Delta W)$. However, it is extremely inefficient for systems such as neural networks, which have a high-dimensional parameter space to be sampled in order to obtain the averages in the optimization steps. In [19] and [22], a method based on the gradient of (10), especially adapted for feed-forward networks, is developed. It has the advantage of being deterministic and much more stable. In addition, it is easily computable with an algorithm of the same order as the back-propagation of the gradient of $E(W)$.

6.4. Relation between interference prevention and generalization

There was an implicit assumption in the derivation of our interference prevention method: the basic features of $\mathcal{P}(\Delta W)$ and especially its variance are independent of the W used to encode the old items. We have supposed that this variance, which is directly related to the expected error for the new item, does not change while performing the minimization (11). In other words, we minimize future interference assuming an expected error for the new items that is independent of W.

But the error in the new items (or equivalently, the variance of $\mathcal{P}(\Delta W)$) is another factor determining the interference that these new items will produce and, of course, it is also controlled by the selection of W. Thus, there exists an alternative way to prevent interference, namely reducing the error in the new items. This is nothing more than improving generalization.

The point we want to make next is that the minimization (11) is also useful to control generalization. This can be understood in two ways:

- First, by reducing second derivatives of the parameters, their information content[6] with respect to the encoding of the items is also reduced. Controlling the information content of a model is a general way of controlling its generalization.

- The other way is considering the term $\frac{\sigma^2}{2} \sum_i \Delta W_i^2 \frac{\partial^2 E}{\partial W_i^2}$ as a regularizer that constrains the system to be simple or smooth. The use of regularizer terms is another classical technique that controls the smoothing of $F(X)$ by means of a regularization coefficient that regulates the importance of the regularizer. In our case this regularization coefficient is σ^2.

7. Experimental results

We show now results obtained by combining the two complementary algorithms for catastrophic interference avoidance. First the robustness against changes is enhanced by the minimization of (9), and then the new patterns are introduced while minimizing retrograde interference by means of the LMD algorithm. Plain back-propagation and retrograde interference minimization with LMD have been extensively compared [18], [20], [21].We concentrate on studying the benefits of complementing LMD with the interference prevention algorithm.

[6] The information content of a parameter can be approximated by $log\left(\frac{\partial^2 E}{\partial W_i^2}\right)$, assuming a uniform a priori distribution for it [17].

7.1. A first example

The following experiment used a neural network architecture with seven hidden units. Fourteen random samples of the function $sin(x_1 + x_2)$ were chosen as an initial training set for the network. The network was trained using our *interference prevention* method, i.e., by minimizing (10) following its gradient. This process was repeated eleven times using different σ^2, producing eleven different networks. For each of these networks, we tested the effect of introducing twelve more random samples of the same function, using the standard ($c_i = \frac{\partial^2 E}{\partial W_i^2}$) and the coarse ($c_i = 1$) versions of the LMD algorithm for *retrograde interference minimization*. Note that, adhering to our simplified formulation, the LMD algorithm always encodes perfectly the new patterns. Thus, the state of the network after their introduction is entirely characterized by the error increment in the old patterns.

Figure 2 shows the average error increments produced by the introduction of the new items. An important fall of catastrophic interference can be observed, especially in the second half of the graph. This reduction is due in part to the improvement in generalization, which is reflected in Figure 3, where the coarse version distances suffer a small drop located more or less in the same place, due to the lower error of the new items at arrival time, which requires smaller weight modifications. Observe that the origin of abcissas corresponds to coding the old patterns with plain backpropagation (i.e., no interference prevention is applied). Applying also plain back-propagation instead of LMD to code the new pattern (i.e., no retrograde interference avoidance) produces error increments that go beyond the scale of the graph.

The distances in general are greater in the standard LMD than in the coarse LMD, because the latter explicitly minimizes $||\Delta W||$, while the former looks for privileged directions suggested by the second derivatives. When the network is trained with the classical backpropagation method, i.e., following the gradient of $E_N(W)$ in discrete steps, the results depend on the length of these steps. The total distance covered in weight space tends to decrease with the shortening of the length of the steps (at the cost of longer training times). In the infinitesimal limit, the solution tends usually to approximate the coarse version of LMD [20].

Note that the distances covered by the standard version grow with σ^2. The reason is that not all second derivatives decrease in the same way when σ^2 increases. The minimum of (10) does not make a pressure proportional to the second derivative's value, but an equal pressure for large and small ones. Therefore some of them become almost null, while others remain large.

This has consequences for the retroactive interference problem formulation: the cost coefficients are the second derivatives and, thus, weights with null second derivatives can be changed arbitrarily. This problem is similar to the excessively large steps that optimization methods based on the Hessian (like Newton or Pseudo-newton) perform when the second derivatives are small. We solve it in the usual way, namely by adding a constant k to the coefficients so that $c_i = \frac{\partial^2 E}{\partial W_i^2} + k$. In [17] we argue that a sensible value for this constant in the case of feedforward neural networks is the square of the maximum possible activation of the hidden units.

7.2. Experiments using the Pumadyn datasets

The next series of experiments have been performed using data from the Pumadyn family datasets [24](loaded from the Delve database [25]), which come from a realistic simulation of the dynamics of a Puma 560 robot arm. The inputs in the datasets chosen consist of angular positions, velocities, torques and other dynamic parameters of the robot arm. The output is the angular acceleration of one of the links of the robot arm. We have used the two datasets in the family labelled with the attributes: 32-dimensional input, fairly linear, and medium noise in one case, and high noise in the other case. We made the same type of experiments shown in Figure 2, but only with the standard version of LMD, since it is the one that works best. Moreover, we have added a very important information to the graphs: the generalization error obtained for each of the values of σ^2, evaluated over 2000 untrained patterns. Networks with forty hidden units were first trained with 200 or 400 patterns and then the average error increment produced by introuducing 200 new patterns separately was evaluated. The results of all combinations of number of previously trained patterns and degrees of noise are displayed in Figures 4 through 7.

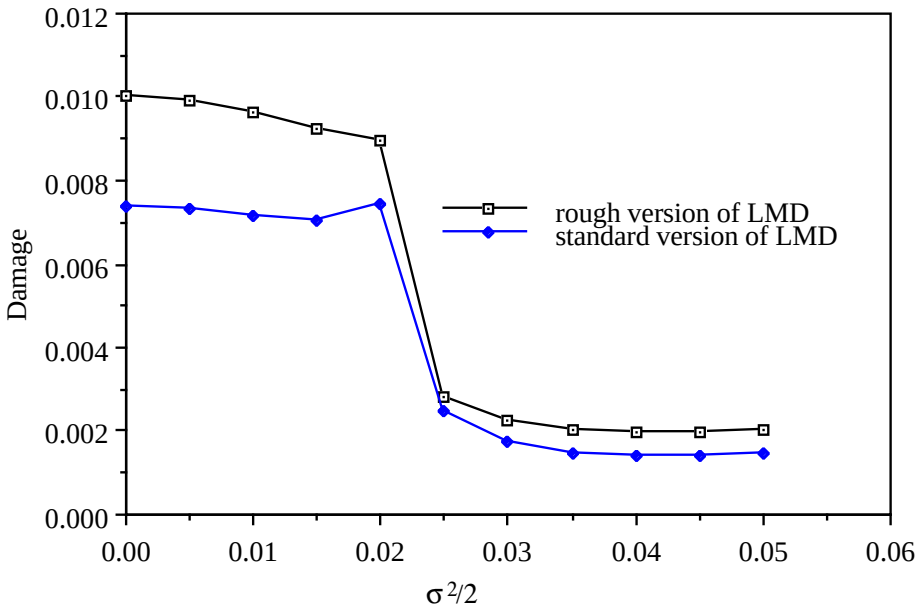

Fig. 2. Average error increments produced by the application of LMD to eleven networks trained with our interference prevention method. The networks have resulted from minimizing (10) for $\frac{\sigma^2}{2}$ between 0 and 0.05, as represented in the axis of abscissas.

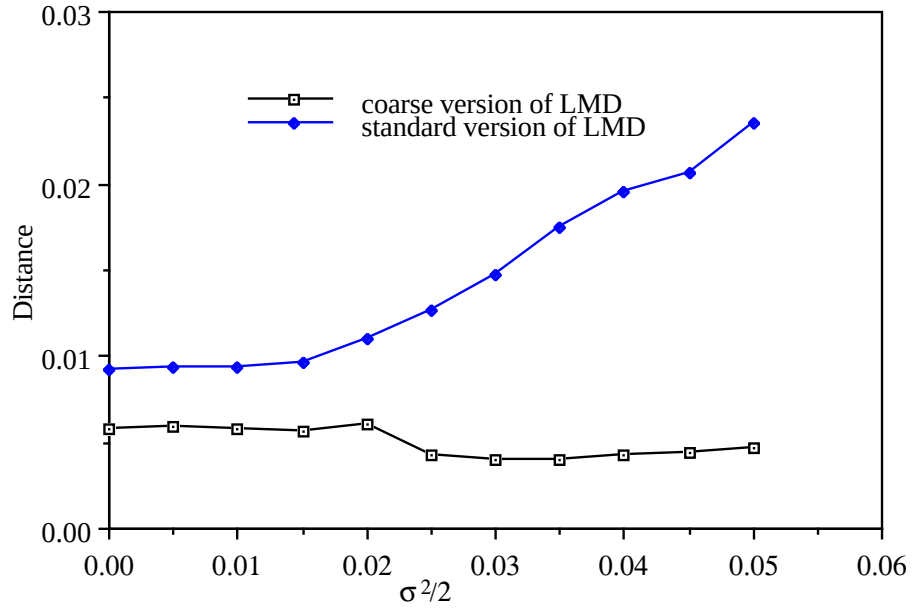

Fig. 3. Average $||\Delta W||$ produced by the application of LMD to the networks in Figure 2.

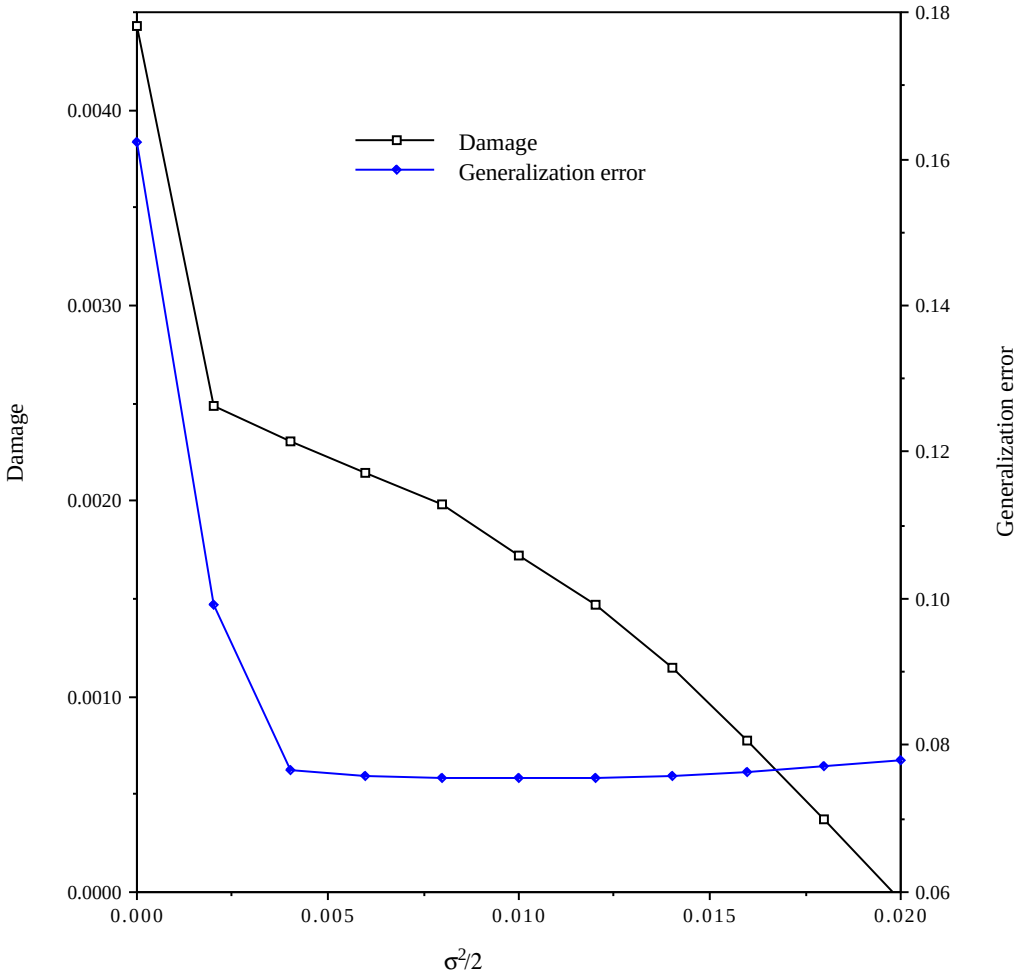

Fig. 4. Same experiments as in Figure 2 but using networks with 40 hidden units and, as training set, the 400 patterns of the high-noise Pumadyn dataset. Damage is measured as the average error increments for the old patterns, whilst the generalization error is evaluated over 2000 untrained patterns.

These figures show that *interference can be alleviated while at the same time improving generalization*. This is in contrast with other strategies for interference avoidance based on a special coding of patterns (e.g., saturating the hidden units to get more local representations), which produce input-output functions $F(X;W)$ (e.g., piece-wise step functions) of a different nature from the function being approximated, thus resulting in high generalization errors. However, we are forced to use the same regularization coefficient for controlling generalization and prevention of interference, the best values for each of these purposes being usually different. Therefore, there is a trade-off that should be considered depending on the application.

If generalization takes priority, the potential benefit of the interference prevention procedure depends on several factors. One of such factor is the number of patterns already stored in the network. The more information a network has stored in, the more its approximation power is well directed and, therefore, the less it requires to be regularized. This can be seen by comparing the curves in Figures 4 and 5: with double number of trained patterns, the generalization curve is more squashed against the left vertical axis. It can also be seen in how the generalization curve of Figure 7 increases more slowly when compared to Figure 6. Another factor is the amount of noise in the examples. The more noisy the training patterns are,

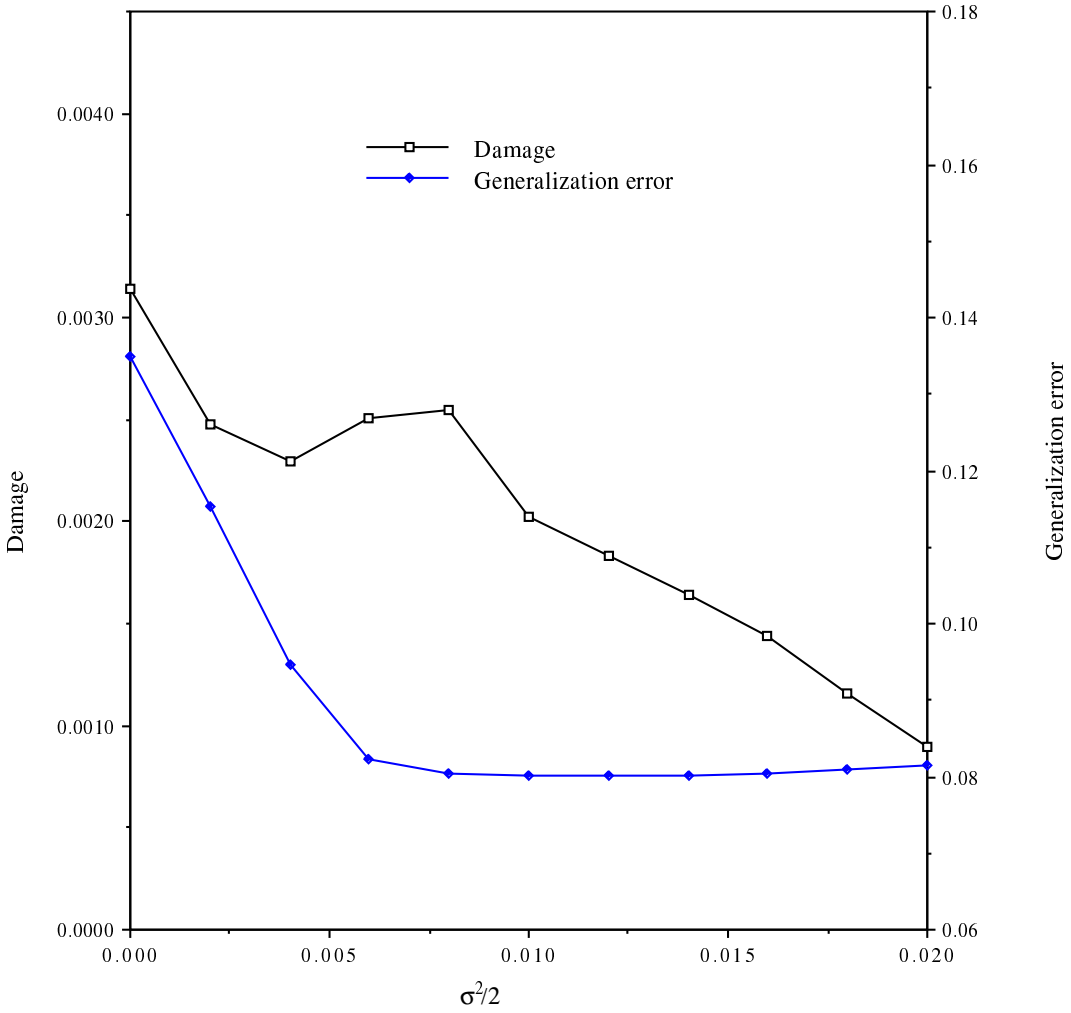

Fig. 5. As in Figure 4, but using the 200 training patterns of the high-noise Pumadyn dataset.

the more convenient it is to smooth $F(X;W)$ by increasing the regularizer coefficient. This is very evident when comparing Figures 6 and 7 (medium noise) with Figures 4 and 5 (high noise), which exhibit generalization error minima at higher values of σ^2. Moreover, the error raises very gently with σ^2 in these figures, allowing for large reductions in interference without paying a high cost in the generalization account.

Therefore, when priority is given to generalization, the narrowest margins for benefits in interference prevention occur for networks trained intensively with a large number of noiseless patterns. But this is precisely the case in which interference is less serious, since the new patterns are better predicted and their introduction produces less damage. This can be checked by observing that the generalization minimum for the network trained with 400 medium-noise patterns (Figure 6) has an associated damage that is an order of magnitude lower than that of the opposite case (200 high-noise patterns) displayed in Figure 5.

Finally a few words about an aspect of Figures 6 and 7 that could seem strange: the damage curve quickly drops to zero and apparently continues with negative values. In fact, interference takes negative values; that is, the encoding of old patterns is improved (rather than disrupted) by the introduction of the new patterns. Note that this happens when the network is highly over regularized, so that the smoothing constraint pushes the interpolating curve $F(X;W)$ far from the trained

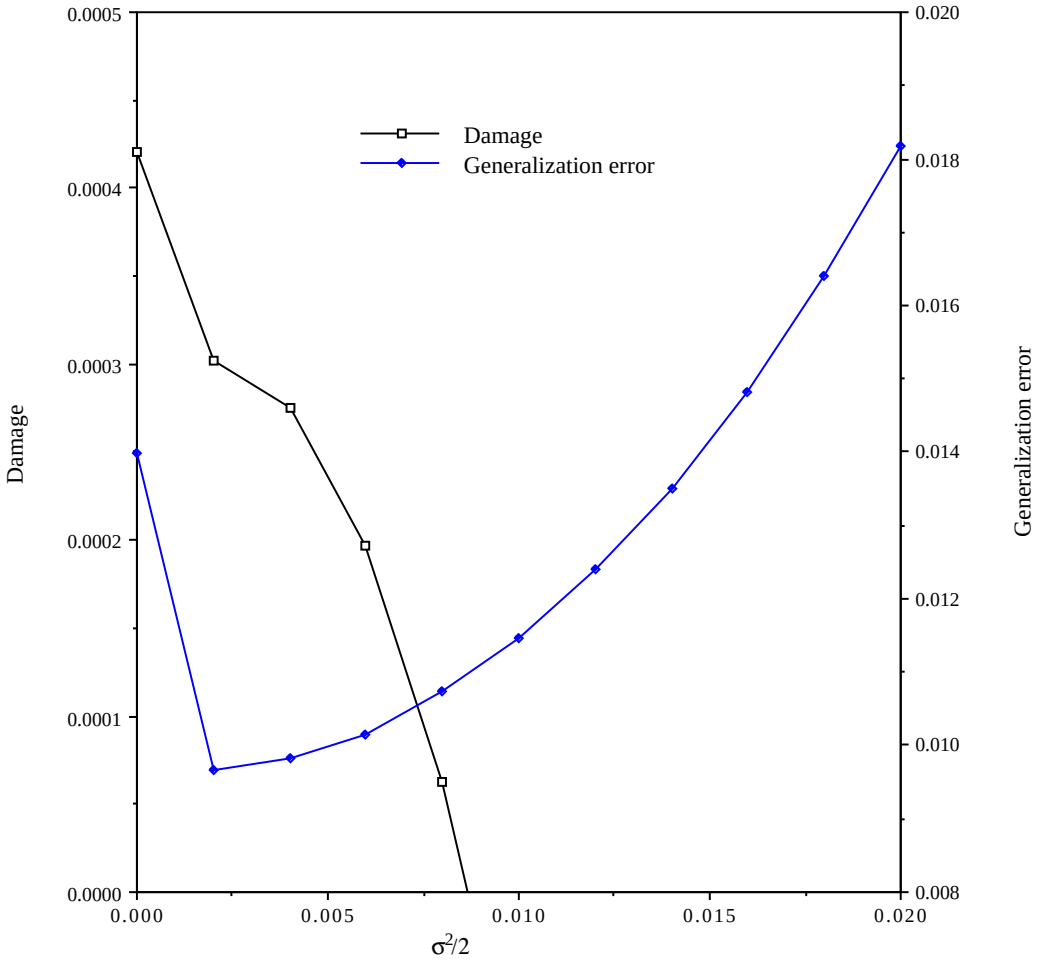

Fig. 6. As in Figure 4, but using the 400 training patterns of the medium-noise Pumadyn dataset.

patterns. Then the introduction of new patterns (that in Figures 6 and 7 have not much noise) without such constraint brings the interpolation curve nearer to the old patterns with high probability.

7.3. Limitations of the proposed algorithm

Together with the benefits above, we must also point out the limitations in the application of our method for interference prevention. We said the drop in the distances for the coarse LMD in Figure 3 was due to generalization. This is true, but the fall that would correspond to the improvement in generalization should be greater. This means that, although the errors are lower, the weights have had to be modified almost the same. The reason is that the algorithm minimizing (10) makes the network output insensitive to changes in the weights for the stored items, but this insensitiveness is transmitted or generalized to the rest of the input space. Because of this, it is also necessary to modify more the weights to introduce the new items, and the potential benefits of the strategy get limited. Like in generalization, the greater the number of items, the greater and more likely the insensitiveness of the items outside of the learning set will be. When the items are few, the results are irregular, as the network has become insensitive for the new items located near a group of old items.

8. Conclusions

Two conditions are required for catastrophic interference to occur:

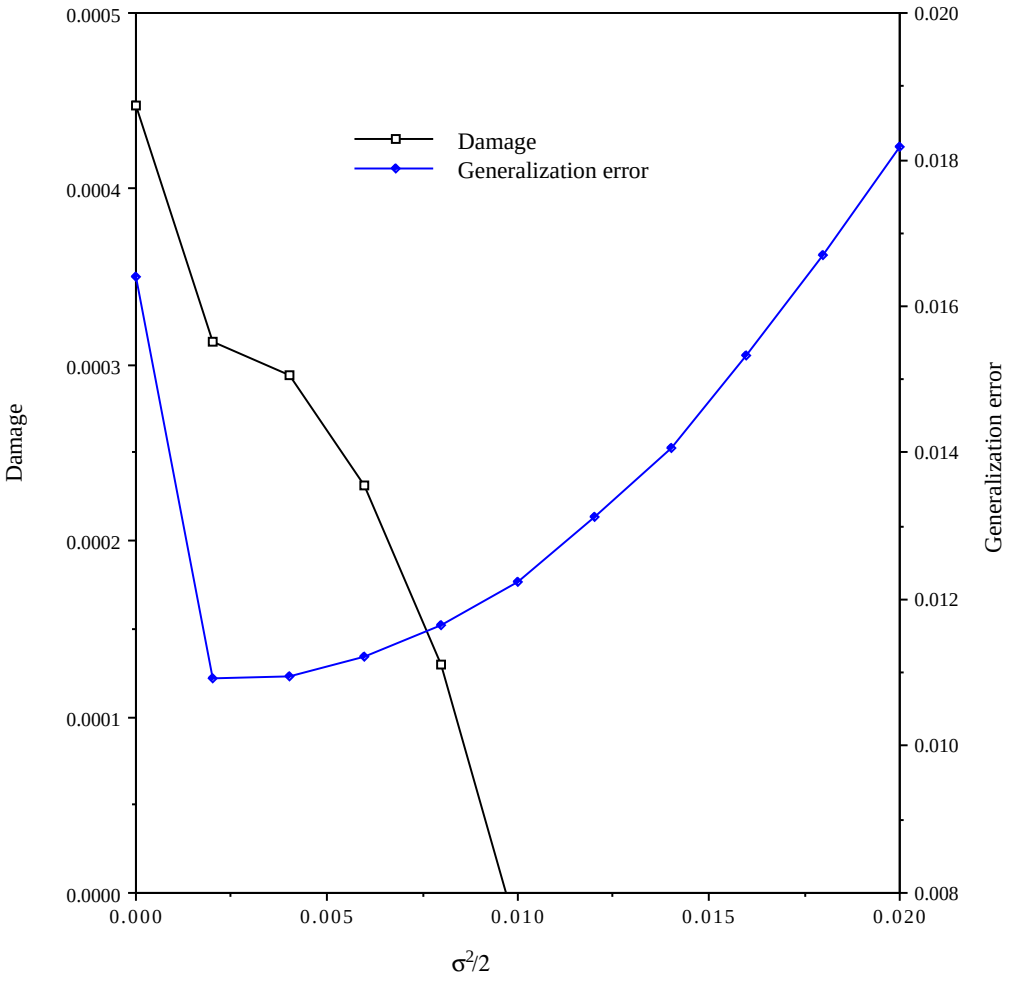

Fig. 7. As in Figure 4, but using the 200 training patterns of the medium-noise Pumadyn dataset.

- The isolated training with new items without reminding the old ones, and
- The use of distributed representations.

We have typified the approaches to solve the interference problem by their degree of retraining with old items, and by the locality of their representations.

We have proposed a two-stage framework to deal with the interference problem based on the information available at each moment. Retroactive minimization deals with interference when the new item is already known. It can be formulated as the search for the weights minimizing the error increments of the already stored items subject to the encoding of the new information. In practice, the best model affordable for a highly dimensional system is a weighted sum of squares of the changes in the parameters. We have shown that the best coefficients are in average the second derivatives of the parameters, even outside of the minimum. For feedforward neural networks, a very efficient algorithm can be used to solve this constrained minimization.

Instead, at the earlier stage of interference prevention, when the new item has not yet arrived, the corresponding weight changes are also unknown. Thus, the only reasonable way of minimizing in anticipation the cost function is minimizing the coefficients, i.e., the second derivatives, jointly with the error. The effect of this is to make the stored items insensitive to future changes. When tested experimentally, we have found a limited success of this strategy due to an unexpected reason: the insensitiveness to which old items were trained gets

"generalized", especially to nearby zones. If a new item has to be introduced in one of these insensitive zones, larger weight changes are required, and most of the expected benefits are lost. When the old items cover densely the input space, there is no possible gain. There is a solution for this situation: to accept and assume that the average sensitivity is the same for old and new items. Thus, the weight increments for the new items will depend on the sensitivity (second derivatives) of the old items. This is reflected by expressing σ^2 as a function of the second derivatives for the old items, and the cost function (10) becomes [17]:

$$E(W) + 2(N-1) < E_N > \frac{\sum_i \left(\frac{\partial^2 E}{\partial W_i^2}\right)^2}{\left(\sum_i \frac{\partial^2 E}{\partial W_i^2}\right)^2}, \quad (12)$$

where $< E_N >$ is the expected error function for the new item. Unfortunately, this function is more complex than (10) and its gradient is harder to calculate. This failure to avoid interference completely with a simple procedure was previously expected, as explained in Section 3.3.

Moreover, there are many scenarios in which the blind application of the hypothetically best possible algorithm could be inappropriate. In fact, as mentioned in Section 4, usually it would be better to introduce only partially the new item, leaving a certain error that is exchanged for a minor error increment in the old items. The appropriate balance point in this trade-off depends critically on several factors:

- The capacity of the memory system, i.e., in what measure it is able to assimilate all the items.
- The amount of noise in the data.
- The number of already stored items. As it grows, the comparative importance of the new item error decreases.
- The variability in time of the function to be approximated. If the function changes quickly, the comparative importance of the errors in new items becomes more important, and more interference should be allowed.

A rule of thumb that is generally correct when the objective function is static or changes steadily is that the error in the new item should not be made lower than the average error in the old items.

We have argued, with others, that distributed representations are indispensable for good generalization. But, is this completely true? Think in this extremely localist representation: the items themselves as a list of input-output pairs, with no other structure or parameter. But, each time an answer to an arbitrary input is required, one can make some very complicated process, for example, building a sigmoidal feedforward network, training it with the stored items, and producing as answer the output of the network. When a new item is introduced there is no catastrophic interference, because it is just added to the list. Thus, the key point is shifting the processing time from training to the generation of answers by the system. More practical methods than the one above could be imagined and some work in the literature [2] can be considered as other, more practical examples of moving computational cost to the recall phase.

So, under this point of view, the question is where to put the burden of processing. Putting it in the learning phase is advantageous if there is enough time for it and one continuously has to generate outputs and react very quickly to the inputs. This is the scenario for animals in their environments. Thus, based on engineering principles, we think that there are two ultimate reasons for which the cortex uses distributed representations that constrain it to slow learning. The first is that, being the residence of long-term memory, it must be able to store great quantities of information, which implies a high degree of compactness that can only be reached using distributed representations, as explained in Section 3.2. The second reason is the requirement of very quick responses to the stimuli (on which life or death can depend) that must be however "optimal" in the sense of well generalized form past experiences. This can be obtained only if the influences of past memories required to respond to new stimuli are already calculated during a previous learning phase and explicited as distributed representations, as explained before.

References

[1] G. An, The effects of adding noise during backpropagation training on generalization performance, *Neural Computation* **8** (1996), 643-674.

[2] C.G. Atkenson, A.W. Moore, and S. Schaal, Locally Weighted Learning, *Artificial Intelligence Review* (in Press).

[3] A. Blum, and R.L. Rivest, Training a 3-node neural network is NP-complete, in: *Proc. of the Workshop on Computational Learning Theory*, Morgan-Kauffman Publishers, San Mateo, CA, 1988, pp. 9-18.

[4] R.M. French, Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference, in: *Proc. of the 16th Annual Conf. of the Cognitive Science Society*, Erlbaum, Hillsdale, 1994, pp. 335-340.

[5] P.A. Hetherington and M.S. Seidenberg, Is there catastrophic interference in connectionist networks?, in: *Proc. of the Eleventh Annual Conf. of the Cognitive Science Society*, Erlbaum, Hillsdale, NJ, 1993, pp. 26-33.

[6] S. Judd, *Neural Network Design and the Complexity of Learning*, MIT Press, Cambridge, 1990.

[7] Y. Le Cun, J.S. Denker, and S.A. Solla, Optimal Brain Damage, Advances in Neural Information Processing Systems 2. *Neural Information Processing Systems 2*, Morgan Kauffman Publishers, San Mateo, CA, 1990.

[8] J. McClelland, B. McNaughton, and R. OReilly, Why there are complementary learning systems in the hippocampus and the neocortex: Insights from the successes and failures of connectionist models of learning and memory, *Psychological Review* **102** (1995), 419-457.

[9] M. McCloskey, and N.J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: *The psychology of learning and motivation*, G. H. Bower, ed., Academic Press, New York, 1989.

[10] K. McRae and P.A. Hetherington, Catastrophic interference is eliminated in pretrained networks, in: *Proc. of the Fifteenth Annual Meeting of the Cognitive Science Society*, Erlbaum, Hillsdale, NJ, 1993, pp. 723-728.

[11] A.F. Murray and P.J. Edwards, Synaptic weight noise during multilayer perceptron training: Fault tolerance and training improvements, *IEEE Transactions on Neural Networks* **4**(4) (1993), 722-725.

[12] A.F. Murray and P.J. Edwards, Enhanced multilayer perceptron performance and fault tolerance resulting from synaptic weight noise during training, *IEEE Transactions on Neural Networks* **5** (1994), 792-802.

[13] D.C. Park, M.A. El-Sharkawi, and R.J. Marks II, An adaptively trained neural network, *IEEE Transactions on Neural Networks* **2**(3) (1991), 334-345.

[14] R. Ratcliff, Connectionist models of recognition memory: constraints imposed by learning and forgetting functions, *Psychological Review* **97**(2) (1990), 235-308.

[15] A. Robins, Catastrophic forgetting, rehearsal and pseudorehersal, *Connection Science* **7**(2) (1995), 123-146.

[16] A. Robins, Consolidation in neural networks and in the sleeping brain, *Connection Science* **8**(2) (1995), 259-275.

[17] V. Ruiz de Angulo, Interferencia catastrófica en redes neuronales: soluciones y relación con otros problemas del conexionismo, Ph. D. Thesis, Universidad del Pais Vasco, 1996.

[18] V. Ruiz de Angulo and C. Torras, The MDL algorithm, in: *Proc. of the Int. Workshop on Neural Networks (IWANN91)*, A. Prieto ed., Lecture Notes on Computer Science, Vol. 540, Springer-Verlag, 1991, pp. 162-172.

[19] V. Ruiz de Angulo and C. Torras, Random weights and regularization, in: *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN94)*, G. Marinaro and P. Morasso eds., Springer-Verlag, 1994, pp. 1456-1459.

[20] V. Ruiz de Angulo and C. Torras, On-line learning with minimal degradation in feedforward networks, *IEEE Transactions on Neural Networks* **6**(3), (1995) 657-668.

[21] V. Ruiz de Angulo and C. Torras, Learning of nonstationary processes, in: *Optimization Techniques*, C.T. Leondes ed., Neural Network Systems Techniques and Applications series, Vol. 2, Academic Press, 1998, pp. 175-207.

[22] V. Ruiz de Angulo and C. Torras, Averaging over networks: properties, evaluation and minimization. Tech. Report IRI-DT-9811, Institut de Robòtica i Informàtica Industrial, Barcelona, Spain, 1998.

[23] A.S. Weigend, D.E. Rumelhart and B.A, Huberman, Generalisation by weight-elimination with application to forecasting. *Neural Information Processing Systems 3*, Morgan Kauffman Publishers, San Mateo, CA, 1991, pp. 885-882.

[24] http://www.cs.utoronto.ca/~delve/data/Pumadyn/desc.html, Detailed documentation file.

[25] http://www.cs.utoronto.ca/~delve/