How to learn multiple tasks

Raffaele Calabretta[1], Andrea Di Ferdinando[1], Domenico Parisi[1], Frank C. Keil[2]

[1] Institute of Cognitive Sciences and Technologies
Italian National Research Council, Rome, Italy
raffaele.calabretta@istc.cnr.it http://laral.istc.cnr.it/rcalabretta

[2] Department of Psychology Yale University
New Haven, CT, U.S.A.

Abstract The paper examines the question of how learning multiple tasks interacts with neural architectures and the flow of information through those architectures. It approaches the question by using the idealization of an artificial neural network where it is possible to ask more precise questions about the effects of modular versus nonmodular architectures as well as the effects of sequential vs. simultaneous learning of tasks. While prior work has shown a clear advantage of modular architectures when the two tasks must be learned at the same time from the start, this advantage may disappear when one task is first learned to a criterion before the second task is undertaken. Nonmodular networks, in some cases of sequential learning, achieve success levels comparable to those of modular networks. In particular, if a nonmodular network is to learn two tasks of different difficulty and the more difficult task is presented first and learned to a criterion, then the network will learn the second easier one without permanent degradation of the first one. In contrast, if the easier task is learned first, a nonmodular task may perform significantly less well than a modular one. It seems that the reason for these difference has to do with the fact that the sequential presentation of the more difficult task first minimizes interference between the two tasks. More broadly, the studies summarized in this paper seem to imply that no single learning architecture is optimal for all situations.

Keywords Neural networks, sequential learning, modularity, neural interference, backpropagation, multiple tasks, genetic algorithms, architecture, What and Where, development.

1. Neural interference

Neural networks are simulation models of the nervous system that can learn to exhibit various types of behavioral abilities. Real organisms in real environments are confronted with multiple tasks and therefore their nervous system must be able to learn multiple abilities. However, in most simulations using neural networks a neural network is trained in a single task and, if one is interested in studying different tasks, different neural networks are used in different simulations. Hence, if we want to understand real organisms there is a need for simulations in which one and the same neural network is trained to exhibit more than a single ability.

Learning many different abilities may pose a problem for neural networks and, presumably, also for real nervous systems. Abilities in neural networks are incorporated in the network's connection weights (LeDoux, 2001). A neural network can be said to possess some particular ability if the network is able to generate the appropriate output for each of a given set of inputs. Since, for any given network's architecture, the particular output with which the network responds to any given input depends on the nature (excitatory or inhibitory) and quantitative weight of the network's connections, the network's abilities or, more generally, the network's knowledge may be said to reside in the network's connections. When an ability is still not possessed, the state of the network can be captured by assigning random values to the network's connection weights. Hence, at this time the network will not be able to generate the appropriate output in response to each relevant input. The acquisition of the ability is a process of progressive changes in the network's connection weights so that at the end the appropriate connections weights are found, i.e., the connection weights that allow the network to respond appropriately to the inputs.

The problem of learning multiple tasks is that if one and the same specific connection inside the network is part of the neural circuit that is responsible for two distinct abilities, it can happen that acquiring one of the two abilities may require the connection to change its weight in one direction, for example by increasing the connection's current weight value, whereas acquiring the second ability may require the same connection to change its weight in the opposite direction, i.e., by decreasing the connection's weight value. We will call this problem "neural interference": if one and the same connection enters into the execution of different abilities, acquiring the different abilities may require changes in the connection's weight that interfere with each other.

In this paper we examine the problem of neural interference by describing various simulations that explore the underlying causes of the problem and propose various ways of solving it.

2. Solution 1: Modular networks

One solution to the problem of neural interference is modularity. If a nervous system must acquire the ability to execute not a single task but two or more different tasks, a modular architecture may work better than a nonmodular one. In learning two or more tasks with a modular architecture one particular set of neurons (module) is dedicated to each task so that the synaptic weights of each module can be adjusted without interfering with the other tasks. In contrast, in a nonmodular architecture, in which all the synaptic weights are involved in all the tasks, adjusting one weight to better perform in one task can result in less good performance in other tasks.

Rueckl et al (1989) trained neural networks to learn two different tasks requiring the extraction of two different types of information from the same input. The network's input is contained in a 'retina' in which different types of objects can appear, one at a time, in different positions. In each input/output cycle the network has both to recognize which object appears in the retina (What task) and to determine what is the position of the object in the retina (Where task). In each input/output cycle the network has both to recognize which object appears in the retina (What task) and to determine what is the position of the object in the retina (Where task) (cfr. Ungerleider and Mishkin, 1982; Milner and Goodale, 1995, 2005; Velichkovsky, 2007). The network has two separate sets of output units separately encoding the network's response for the two tasks.

Rueckl et al compared two different architectures, a modular architecture and a nonmodular one (Figure 1). Both architectures have a single layer of internal units, with the input units connected with the internal units through the lower connection weights and the internal units connected with the output units through the higher connection weights. In both architectures the input units that encode what is contained in the retina are all connected with all the internal units. The difference between the two architectures lies in the higher connections. While in the nonmodular architecture the internal units are also all connected with all the output units, i.e., with both the output units that encode the answer to the What task and the output units that encode the answer to the Where task, in the modular architecture a subset of the internal units are connected only with the What output units and the remaining internal units are connected only with the Where output units. Since the What task is more complex than the Where task (see below), Rueckl et al found that the best modular architecture is an architecture which assigns a greater number of internal units to the What task than to the Where task.

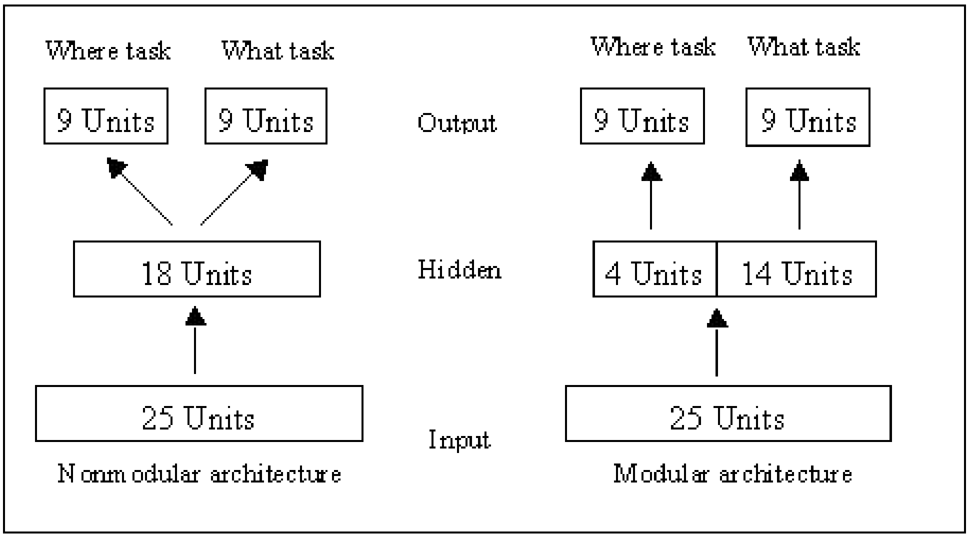


Figure 1. Modular and nonmodular network architectures for learning the What task and the Where task.

The modular architecture is in fact two separate architectures, with two non-overlapping subsets of connection weights each dedicated to only one task. Therefore, in the modular architecture there is no interference between the two tasks. In each cycle, on the basis of the task-specific teaching input, each connection weight always receives a single message for increasing or decreasing its weight value without interference from the teaching input for the other task. In contrast, in the nonmodular architectures the two tasks use two separate subsets of weights at the higher level (connections between hidden layer and output layer) but they share the same weights at the lower level (connections between input units and hidden units). Hence, there may be interference between the two tasks at the level of the lower connection weights in that the same lower connection weight can receive conflicting messages from the two teaching inputs. The teaching input of the What task may ask the weight to increase its current value while the teaching input of the Where task may ask the same weight to decrease its value, or vice versa. This predicts that modular architectures will work better than nonmodular ones for learning the two tasks.

In fact the results of Rueckl et al's simulations show that this is the case. Starting from random connection weights Rueckl et al use the backpropagation procedure to progressively adjust these connection weights. In each cycle the network is provided with two distinct teaching inputs which specify the correct answer for the What task and for the Where task, respectively. The network compares its own answer with the correct answer and on the basis of this comparison it modifies its connection weights in such a way that the discrepancy (error) between the network's answer and the correct answer is progressively reduced. At the end of learning the total error is significantly lower for neural networks with a modular architecture than for networks with a nonmodular architecture.

As we have said, in Rueckl et al's simulations the Where task is computationally less complex than the What task. This depends on the fact that "the difficulty of an input/output mapping decreases as a function of the systematicity of the mapping (i.e., of the degree to which similar input patterns are mapped onto similar output patterns and dissimilar input patterns are mapped onto dissimilar output patterns)" (Rueckl *et al., 1989),* and systematicity is higher in the Where subtask than in the What sub-task. As a consequence, in modular networks, that effectively are two separate networks, the Where task is acquired earlier than the What task although the terminal error is equally low for the two tasks. In nonmodular networks, the terminal error separately computed for the two tasks is lower for the Where task than for the What task. The reason appears to be that when, after acquiring the Where task which is less complex and is learned first, a nonmodular network turns to the more complex What task the network's connection weights that are shared between the two tasks have already been recruited for incorporating the knowledge about the Where task and, as a consequence, the What task cannot be acquired as effectively as the Where task.

In Rueckl et al (1989) the network architectures are hardwired by the researcher. Di Ferdinando et al. (2001) have used a genetic algorithm to evolve the most appropriate network architecture in a population of neural networks that learn, via backpropagation, the Where task and the What task during their life (see also Calabretta et al., 2003). An individual network's architecture is encoded in the inherited genotype of the network. At the beginning of the simulation a population of random genotypes is generated resulting in a number of different neural architectures. Each individual learns the What task and the Where task during its "life" and only the individuals with the best learning performance are allowed to reproduce by generating a number of "offspring" with the same neural architecture of their (single) "parent" except for some random mutations in the inherited genotype. After a certain number of generations most individuals in the population have a modular architecture with more internal units assigned to the What task and fewer internal units assigned to the Where task. This confirms that modular architectures are better at learning the two tasks than nonmodular architectures.

3. Interference occurs especially in the early stages of learning

We interpret the less good performance of nonmodular networks in learning two tasks at the same time as due to interference, that is, the possible arrival to one and the same connection of conflicting messages for changing the connection's current weight value. In backpropagation learning how much the weight of some particular connection has to be changed is proportional to the error of the unit to which the connection arrives. This error (E) is the discrepancy between the unit's observed and desired activation value:

$$E = t_i - a_i \qquad (1)$$

where t_i is the desired activation value and a_i is the observed activation value.

In general, connections are told to change more substantially their current value when the neural network's errors are larger. On the other hand, when the network's errors are smaller, connection weights are required to change less. One consequence of this is that the phenomenon of interference in nonmodular networks is greater when a neural network's errors are larger and therefore the network's connections are asked to change more substantially their weight value. The more a connection has to change its current weight value, the more serious the interference if the required changes go in opposite directions. For example, the conflict between a message that asks a connection to change its weight value by adding 0.1 and another message that asks the same connection to change its weight by subtracting 0.2 is less strong than the conflict between a message to add 1.0 and another message to subtract 2.0. In the first case the weight will be decreased by 0.1, in the latter case by 1.0. Therefore, in the latter case the conflict creates a more serious problem since the damage with respect to the first task is greater.

If we ask when a neural network's errors are greater, the obvious answer is that it is in the early stages of learning. Therefore, we should expect that the phenomenon of interference in nonmodular networks that have to learn two tasks at the same time will be greater at these early stages. This is difficult to discern from the summed squared error curve during learning. The summed squared error (SSE) is the sum of the squared errors of all the output units for all the input patterns:

$$SSE = ½ \Sigma E_i^2 \qquad (2)$$

If we look at the SSE curve, what we see is that the total error of the neural network decreases very rapidly in the early stages of learning and then much more gradually in the later stages. This in fact is what we observe in nonmodular networks that learn the What and Where task, as in Rueckl et al.'s simulations (Figure 2). Notice that these and all the other data shown in this article belong to our analyses of replications of Rueckl et al.'s simulations.

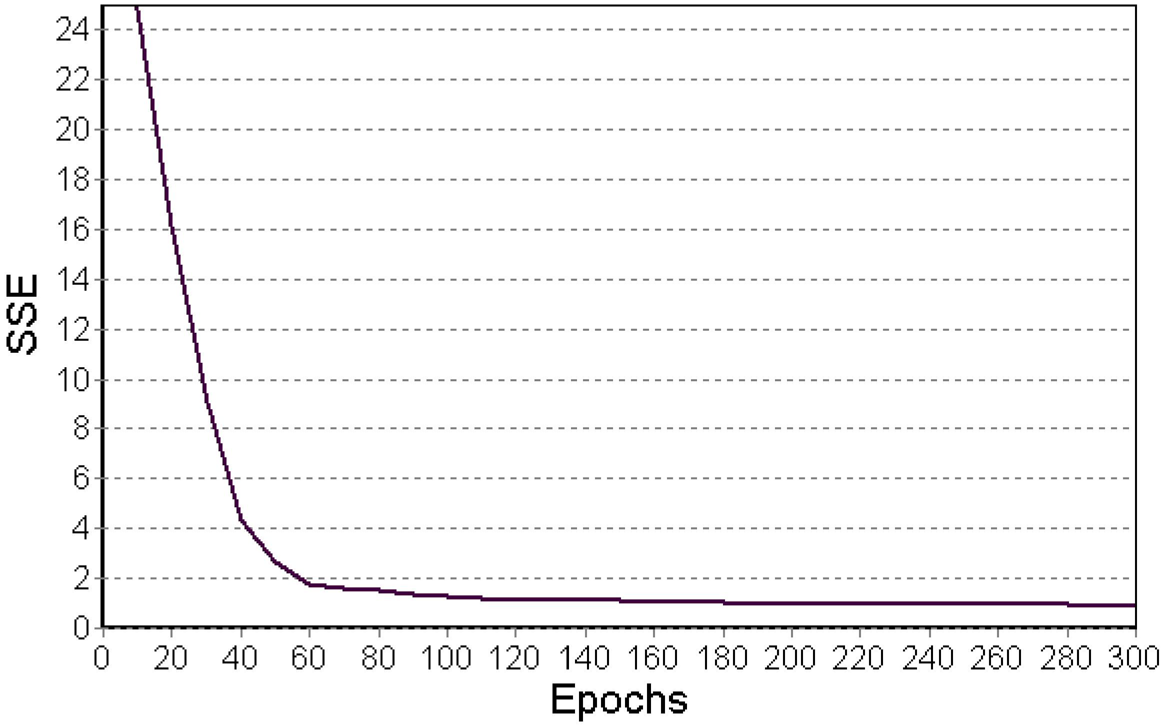


Figure 2. SSE curve for nonmodular networks that learn the What and Where task. The curve represents the average of 10 replications of the simulation.

The curve in Figure 2 represents the average of 10 replications of the simulation. If we look at the curves of the single replications, we observe that in some replications the network succeeds in learning the task (SSE approaches zero), but in most cases it does not. Again, looking at the SSE curve we cannot understand why this happens (Figure 3).

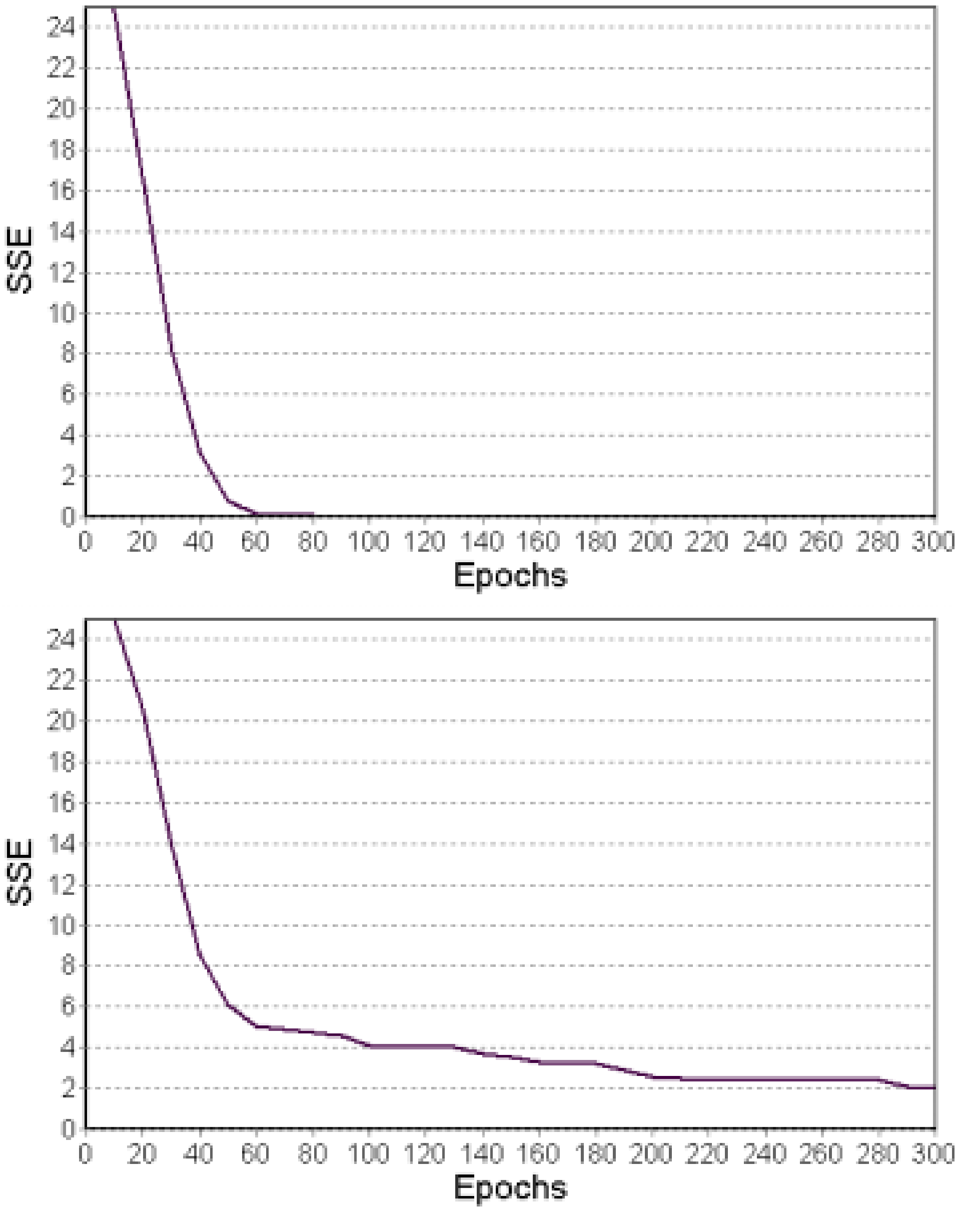


Figure 3. SSE curve for two replications of the simulation with nonmodular networks. In the first replication SSE approaches zero, in the second replication it remains still large after 300 epochs.

We have the same problem if we graph the average absolute error (AAE), that is, the discrepancy between the actual activation value of the output units and the desired activation value (with no plus or minus sign) averaged for all input patterns and for all output units:

$$AAE = \text{Mean}\ (|E_i|) \qquad (3)$$

Again, we might think that the observed decrease of AAE during learning holds for all input patterns and for all output units. However, this is not so. This becomes clear if we consider the single maximum absolute error, that is, the absolute error which is the largest among all absolute errors of the input/output mappings:

$$MAE = \text{Max}\ (|E_i|) \quad (4)$$

If we separately graph MAE and AAE, we see that while AAE decreases very quickly in the early learning stages, the value of MAE actually increases, equally quickly, during the early learning stages approaching its maximum possible value (0.9). In some replications of the simulation, after the initial increase MAE starts to decrease at some point before it reaches its maximum possible value (Figure 4a), but in most replications MAE rises very close to its maximum value and then it does not decrease and is never eliminated (Figure 4b). (Notice that MAE ranges between 0.0 and 0.9 because, following Rueckl et al, we used as training values 0.1 and 0.9.)

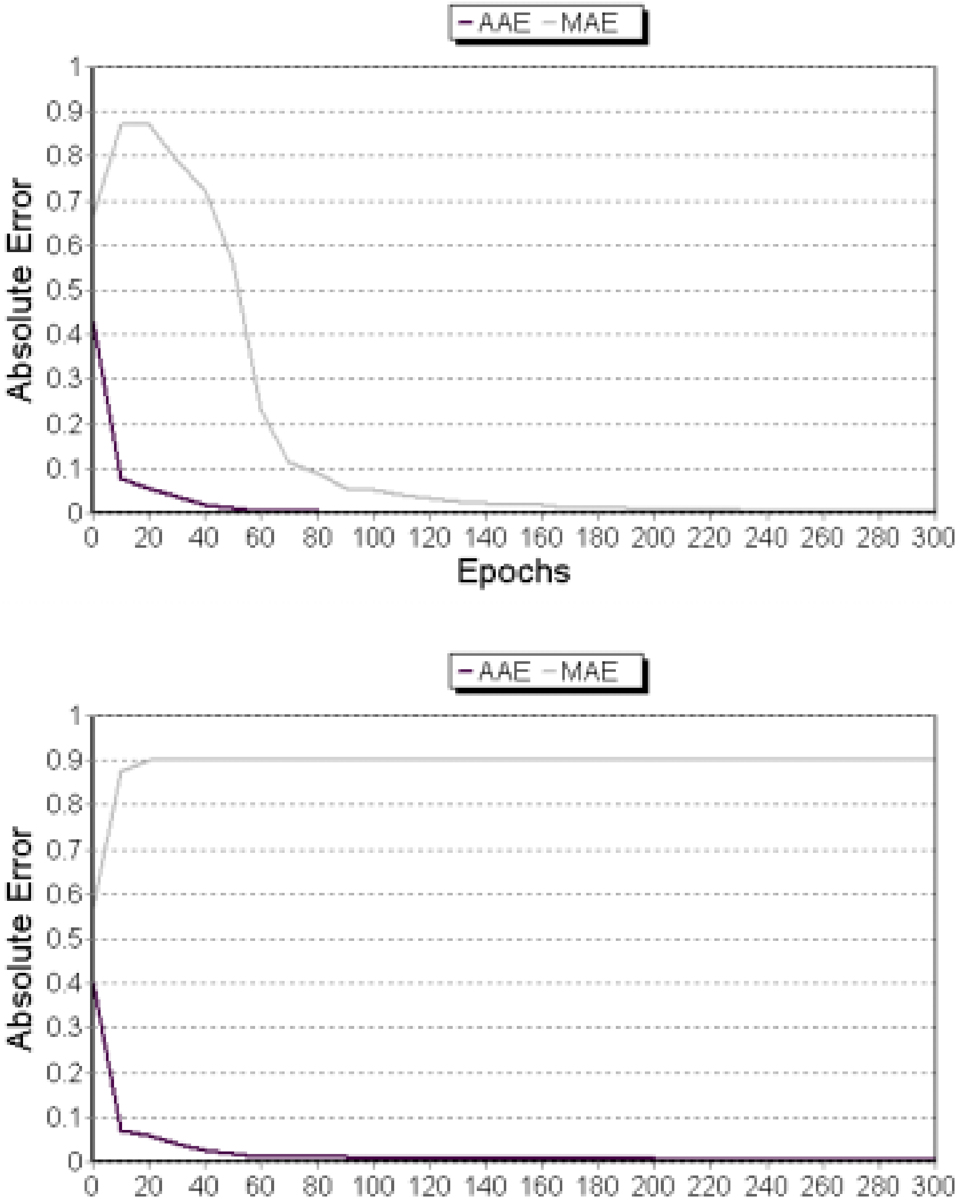


Figure 4. Changes in MAE (gray) and in AAE (black) during learning, for two replications of the simulation with nonmodular networks.

What the graph of Figure 4 shows is that, while the nonmodular network is learning the two tasks, interference causes three effects: (a) at least one of the network's outputs becomes entirely mistaken; (b) this takes place in the very early stages of learning; (c) if MAE gets too close to its maximum value it becomes hard to eliminate as learning progresses.

We can now understand the difference between the two replications of Figure 3. In the first replication MAE doesn't get too close to its maximum value, and thus the network succeeds in solving the task, whereas in the second replication it gets too close to its maximum value and the network is unable to solve the task. The reason why the more the MAE approaches its maximum value, the more it is hard to eliminate, is that in computing the error signal (ES) of a unit (whether output unit or internal unit), the backpropagation procedure uses the first derivative of the logistic function. For example, for an output unit ES (for a given input pattern) is computed as follows:

$$ES = E \text{ x } Deriv_i \qquad (5)$$

where

$$Deriv_i = a_i \text{ x } (1 - a_i) \quad (6)$$

where a_i is the activation state of the unit. The derivative tends to zero for values of a_i that tend to either 1 or 0. Hence, a unit's ES tends to zero when the unit's activation value either tends to 1 or to 0. If, for example, the desired activation value is 1, the unit's ES tends to zero both when the unit's activation state is near 1 (which of course is obvious) and when the activation state is near 0 (which is less obvious) (Figure 5).

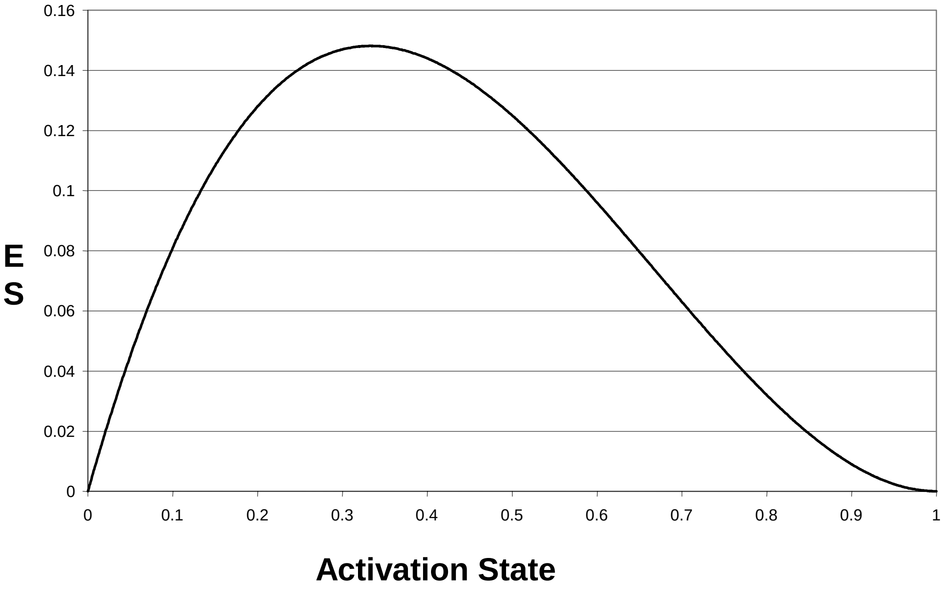


Figure 5. The ES of a unit as function of its activation state.

In other words, rather paradoxically, the formula for computing ES is such that if a unit responds in a completely mistaken manner, its ES is very small. Therefore, the network modifies the weights of the connections arriving to that unit by a very small quantity, which is insufficient to eliminate the error.

As we have already noted, the phenomenon of neural interference occurs especially in the early stages of learning. Rueckl et al (1989) observe that in the early learning stages nonmodular networks are more exposed to a pressure to change in order to learn the easier task, the Where task. After these changes in the network's connection weights have been made, the second, more complex, task, the What task, can be learned only "within the confines of the constraints placed on the hidden node connections by the local computation", which means that it is too late to learn the What task.

However, the ES formula does not imply that, once some particular unit responds in a completely mistaken way, it is impossible to make the unit respond correctly. If the ES for a given unit is very small, the weights that determine the unit's activation state are also changed by very small quantities, and therefore the learning process can take many epochs. But, at the end, it should be possible for the learning algorithm to bring even these units with a very mistaken activation state to their correct activation state. In fact if, instead of discontinuing learning after 300 epochs, as in Rueckl et al (1989), we continue learning until epoch 10,000, the terminal SSE turns out to be near zero even for nonmodular networks in all the 10 replications of the simulation (Figure 6). Therefore, as suggested by our analysis, interference in nonmodular networks that try to learn two tasks at the same time slows down learning but it does not make learning impossible.

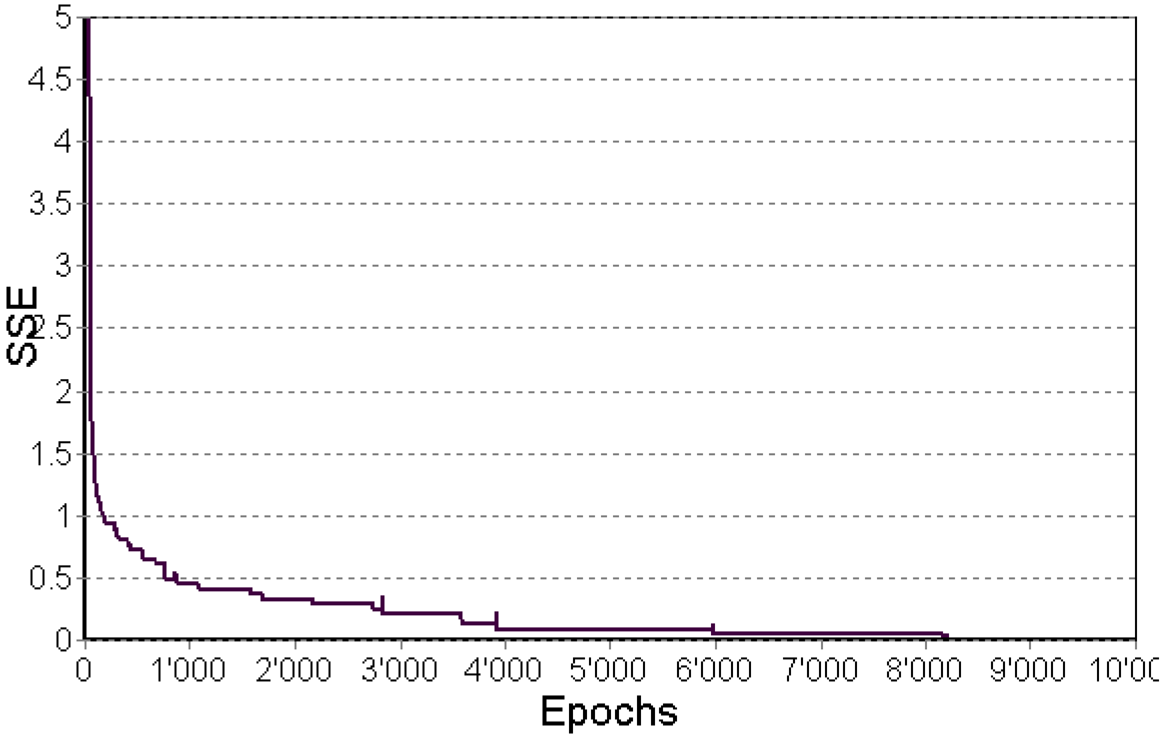


Figure 6. SSE curve for nonmodular networks that learn the What and Where tasks for 10,000 epochs (average of 10 replications of the simulation). Note that the error scale is different than in the preceding figures.

Another way to avoid the problem shown in Figure 5 is to prevent the activation state of a unit from reaching extreme values (i.e., 0.999 and 0.001). In this way, the derivative of the logistic function cannot reach a value very close to 0 and, when the response is completely mistaken, this produces a bigger modification of the weights, thus *reducing* the number of epochs needed to solve the two tasks. In fact, by doing so, we have been able to obtain perfect learning of the What and Where task even in nonmodular networks in Rueckl et al's 300 epochs. This method, however, although efficient, has the limitation that it appears to be tied to the specific learning algorithm used, the backpropagation algorithm. The method does not eliminate the interference that characterizes nonmodular networks but it only succeeds in eliminating its negative effects that we have discussed.

As Rueckl et al's results indicate, interference can be eliminated by using modular networks, in which distinct set of connections, i.e., modules, take care of separate tasks and therefore it is impossible that contradictory instructions for weight change arrive to one and the same connection. (A form of intra-task interference exists also in learning a single task but it is much less strong than when two tasks have to be learned at the same time. Cf. Plaut and Hinton, 1987.) Consider that the MAE curve only tells us that *at least* one output becomes entirely mistaken, but the actual number of outputs entirely mistaken can be more than one. To quantify the number of output units with an entirely mistaken activation state, we can introduce a threshold value: when a unit's error (E) reaches a value larger than 0.899, the unit's activation state is considered as entirely mistaken. Using this measure we have determined the number of outputs that are completely mistaken and we have observed that this number is smaller in modular than in nonmodular networks, i.e., 0.3 vs. 1.9 (see Table 2b). This seems to confirm our analysis in terms of MAE.

Modularity is one way to eliminate neural interference. Another way, which allows even nonmodular networks to learn two or more tasks, is sequential learning: the network learns the two tasks not at the same time but one after the other.

4. Solution 2: Learning two tasks one after the other

In all the simulations described sofar neural networks start learning the two tasks, the What task and the Where task, at the same time. From the beginning of the simulation the performance of the network in the two tasks is evaluated in each cycle using the two teaching inputs and all the network’s connection weights are modified accordingly. After 300 cycles the SSE has reached a stable value which is 0.16 for modular architectures and 0.90 for nonmodular architectures (average of 10 replications of the simulation with randomly selected initial set of connection weights). This is the SSE of the network. Since, as we have observed, the What task is more difficult to learn than the Where task, while for modular networks the two separate components of SSE for the What and the Where tasks are both very low at the end of the simulation, for nonmodular networks the SSE component for the What task is significantly higher (0.90) than the SSE component for the Where task (0.00).

Now imagine that a neural network learns the two tasks not both at the same time but one after the other. The network starts learning one task, i.e., only the teaching input for this task is provided to the network. Only after this first task has been learned, i.e., the SSE for the task is near zero, the second task is introduced and the network starts learning the second task. Notice, however, that when the network starts learning the second task, the teaching input for the first task continues to be provided to the network. In other words, when the learning of the second task begins the learning of the first task is not discontinued.

When this form of sequential learning is applied to modular networks the results are not different from the case in which both tasks are learned at the same time. As already observed, modular networks are actually made up of two separate sub-networks (modules) that do not share any single connection weight (cf. Figure 1). Therefore, whether the two tasks are learned together or sequentially is irrelevant and the SSE for both tasks goes to zero in both cases. The interesting question is: can sequential learning allow nonmodular networks to acquire both tasks equally well and therefore to avoid their handicap with respect to modular networks which is apparent when the two tasks are learned together?

The answer seems to be Yes. We have trained a nonmodular network by providing the network with the What teaching input for 100 cycles, omitting the Where teaching input. One hundred learning cycles are sufficient for the network to reach an SSE value of near zero in the What task. From this point on, i.e., starting from cycle 101, both the teaching input for the What task and the teaching input for the Where task are provided to the network. At the end of the simulation (300 cycles) SSE is 0.04 (average of 10 runs of the simulation). In other words, if the learning of the two tasks is sequential rather than simultaneous, a nonmodular network can acquire both tasks equally well as a modular network (terminal SSE: 0.16; the difference between these two conditions is not significant).

However, the situation appears to be somewhat more complex. In the simulation just described the nonmodular network first learns the more complex What task and subsequently the easier Where task. We have run another simulation in which the order is inverted. The network first learns the easier Where task and then the more difficult What task. Notice that since the Where task is easier than the What task the network learns the Where task in just 20 cycles (SSE near zero) compared with 100 cycles which are necessary to learn the What task. Therefore, the new task, the What task, is added after only 20 cycles since the beginning of the simulation. At the end of the simulation (after 300 cycles) the SSE is 0.53 and this error is almost entirely due to the What task which is learned after the Where task has already been learned. This error is larger than the terminal error for modular networks (0.16) but smaller than the terminal error the nonmodular networks that start learning the two tasks at the same time (0.90). Apparently, then, learning two tasks in sequence rather than at the same time is beneficial for nonmodular networks in all cases but it is better to learn the more complex or difficult task first and the easier task later than the other way round. With +equential learning, if the more difficult task is learned first nonmodular networks can learn both tasks as effectively as modular networks (see Table 1). But if the easier task is learned before the more difficult task the terminal error turns out to be larger and the second, more difficult, task is not learned as effectively as in modular networks.

	modular	nonmodular	diff. m/nm
Both	0.16	0.90	signif.
Where -> Both	0.16	0.48	signif.
What -> Both	0.16	0.04	not signif.

Table 1. Error in modular and nonmodular networks, in the three conditions (the data refer to the average of 10 replications of the simulation). For each of the three conditions, we performed a one way ANOVA in which the network architecture was manipulated between the seeds (where each seed corresponds to a different subject).

Notice that in the simulations we have described the two tasks are acquired sequentially but the learning of the task which is acquired first is not discontinued when the network starts learning the second task. What happens if the nonmodular network first learns one task and after this task has been acquired the network starts learning the second task but the teaching input for the first task ceases to be provided to the network? The answer is catastrophic forgetting (French, 1999). After 300 learning cycles the SSE is 17.15 for the simulation in which the What task is learned first and then the network learns the Where task, and is 16.65 for the simulation in which the Where task is learned first and then the network learns the What task. As the literature on catastrophic forgetting shows, sequential learning is possible only if, after learning the first task, the first task continues to be rehearsed while the network is learning the second task.

How can we explain the pattern of results we have obtained? Why sequential learning with rehearsal of the first task allows even a nonmodular network to learn two tasks? Why is learning the more difficult What task first and then add the simpler Where task better than learning the two tasks in the opposite order?

The answer is that with sequential learning we are able to reduce, even though not completely, the neural interference between the two tasks. (Intra-task interference, which however is weaker, continues to operate). Figure 7 shows how this happens. As explained before, when the network learns two tasks at the same time, it is in the first stages of learning that interference reaches its maximum strength because in these first stages the error messages are larger (Figure 7a). On the contrary, in the first stages of sequential learning there is only one task to be learned, and therefore there is no interference (Figure 7b). When the first task is learned (SSE approaches zero), the second task is added but there is still no interference between the two tasks in that the first task has already been learned and therefore it does not send any error message (Figure 7c). It is true that when the learning algorithm starts to modify the weights to learn the second task, the performance in the first task can somehow decrease. Since the lower connection weights are shared by the two tasks, the changes in connection weights that result from the teaching input for the second task can influence the network's performance on the first task (and vice versa). And, as soon as the performance in the first task decreases, the first task starts again to send error messages. However, these error messages will be very small in comparison with those of the second task (Figure 7d). As learning proceeds, the performance in the first task can continue to decrease for a while, but in the meanwhile the performance in the second task increases, and therefore the error messages from the two tasks will never both be very large at the same time and therefore there will never be the strong interference which occurs when the two tasks are learned at the same time (Figure 7e).

In summary, sequential learning avoids the contemporary presence of two large error messages, and therefore it reduces, even if it does not eliminate, neural interference. A consequence of this is that, given that the first task is learned without any interference whereas the second task is learned with some (reduced) interference, it is better to learn first the more difficult task than the easier task.

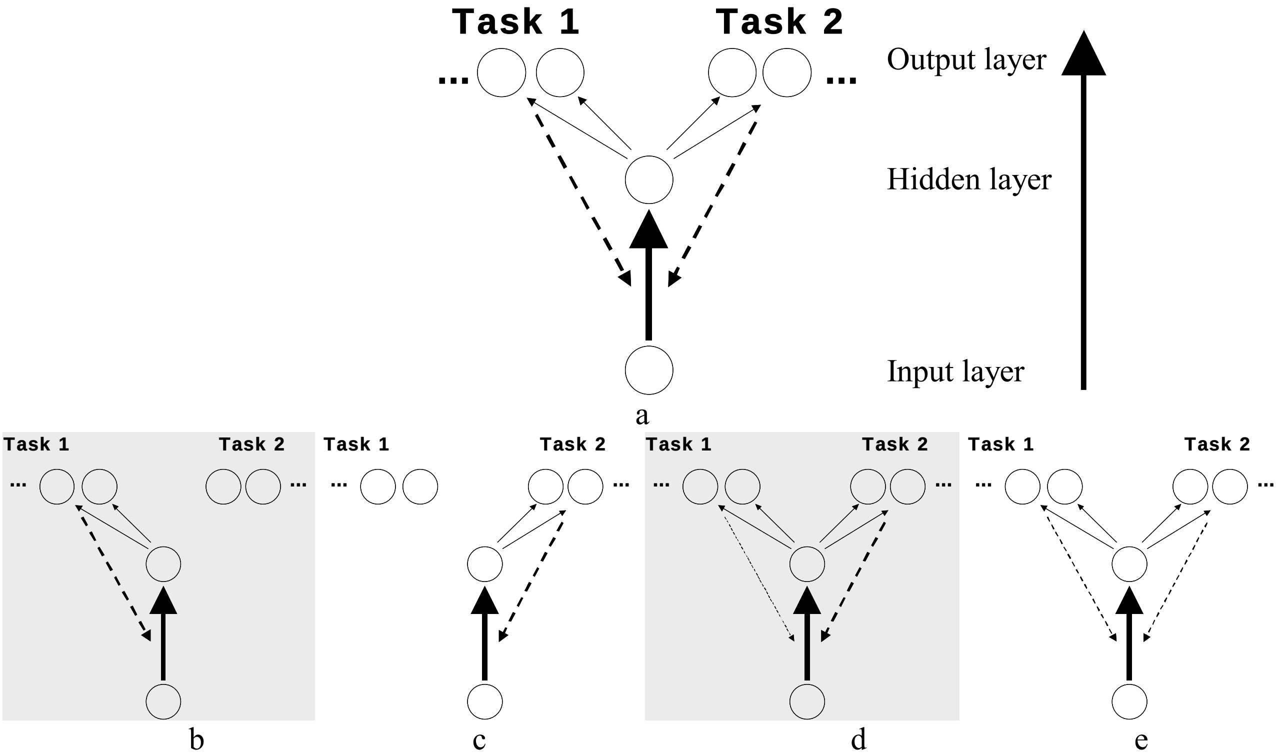
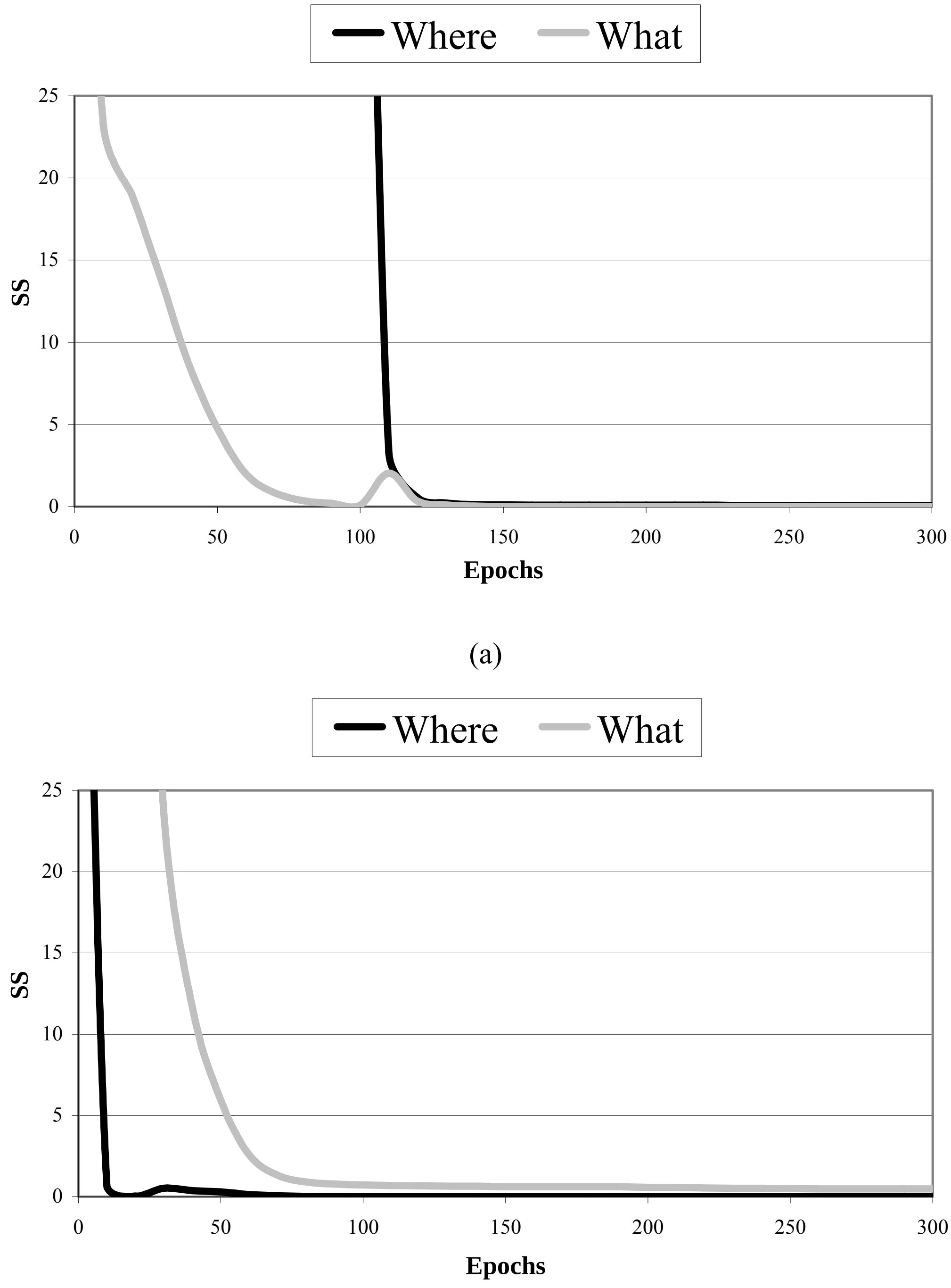


Figure 7. Neural interference between the What task and the Where task in nonmodular network architecture. The solid lines represent connection weights, while the dotted lines represent error messages (the thicker is the dotted line, the bigger is the error message). Figure 7a shows an early stage of simultaneous learning of the two tasks, while Figures 7b, 7c, 7d and 7e show four succeeding stages of sequential learning (for more details, see text).

Figures 8 show the SSE curve separately for the Where task and for the What task for both the simulation in which the What task is acquired first and the Where task is added later (Figure 8a) and for the simulation in which the order of acquisition is reversed (Figure 8b). In both simulations, when the second task is added after the first task has been learned (SSE near zero) the performance in the first task is initially somewhat damaged (small increase in SSE) as a result of the introduction of the new task, but very quickly the damage is repaired and neutralized (SSE is again near zero).

Neural interference occurs during this period. However, for the reasons explained before, its size is not as big as it would be if the two tasks were learned at the same time. Moreover, neural interference is bigger when the task which is learned first is the easier task.



(a)

(b)

Figure 8. SSE curve in nonmodular networks shown separately for the Where task and for the What task, for both the simulation in which the What task is acquired first and the Where task is added later (Figure 8a), and for the simulation in which the order of acquisition is reversed (Figure 8b). The curves represent the average of 10 replications of the simulation. Notice that, given the architecture of nonmodular networks, changes in connection weights that result from the teaching input for a task can influence the network's performance on the other task even if no teaching input

is provided for the other task. In fact, the lower connection weights are shared by the two tasks so that a change in these weights affects the network's performance in both tasks.

We can better understand this last point if we look at MAE. When the easier task is learned first, the number of outputs which are completely mistaken increases (see Table 2a e 2b).

	modular	nonmodular	diff. m/nm
Both	0,40	2,30	signif.
Where -> Both	0,40	1,40	signif.
What -> Both	0,40	0,10	not signif.

Table 2a. Maximum value of the number of outputs completely mistaken (error larger than 0.899) in modular and nonmodular networks in the three conditions during learning process (notice that this value is reached during the early phases of learning). The data refer to the average of 10 replications of the simulation. For each of the three conditions, we performed a one way ANOVA in which the network architecture was manipulated between the seeds (where each seed corresponds to a different subject).

	modular	nonmodular	diff. m/nm
Both	0.30	1.90	signif.
Where -> Both	0.30	1.10	signif.
What -> Both	0.30	0.00	not signif.

Table 2b. Number of output completely mistaken (error larger than 0.899) in modular and nonmodular networks in the three conditions at the end of the learning process (the data refer to the average of 10 replications of the simulation). For each of the three conditions, we performed a one way ANOVA in which the network architecture was manipulated between the seeds (where each seed corresponds to a different subject).

From the comparison of the two tables it can be seen that neural interference pushes some network outputs to become completely mistaken during the early phases of learning, and that when this happens it is very difficult for the network to carry back these outputs to the right values.

5. Neural interference and backpropagation.

One could think that our results concerning the effects of neural interference are due to the specific learning algorithm used, the backpropagation algorithm. In fact, although some of the consequences of neural interference are specific to the backpropagation learning algorithm, neural interference is a more general phenomenon that exists every time a neural network has to learn more than one task at the same time. More specifically:

1- We have seen that the error measure typically used in connectionist simulations, that is, the summed squared error (and, more generally, all error measures that take into consideration a network's global performance), does not show the real nature of the error due to neural interference. This error is not the sum of many small errors distributed on many patterns and many units but it is rather due to a few large errors distributed on a very few units. This is a general result that also applies to other algorithms such as the unsupervised genetic algorithm.

2- We have seen that these few but large errors arise during the early stages of learning and tend to last until the end of learning (compare Table 2a and Table 2b). As we have shown, in the case of the backpropagation procedure this phenomenon is explained by the fact that a unit's error signal (ES) tends to zero both when a particular unit responds correctly and when it responds in a completely mistaken way. In this last case learning is very difficult and slow because, as ES is very small, the network connection weights will be modified by a very small quantity. However, the phenomenon is not specific to the backpropagation algoritm but is more generally due to neural plasticity and how it changes during learning. At the beginning of learning all the network's connection weights are assigned a quantitative value which is randomly chosen in a very small interval (in our simulation between plus/minus 0.30). This choice confers an initial large amount of plasticity to the neural network since the network can improve its performance significantly even by modifying each time very slightly its connection weights. As is well known, during the course of learning the absolute value of the connection weights generally tends to increase and therefore neural networks loose plasticity. If in later learning stages the network's performance is not good, errors will be very difficult to correct because the network has lost much of its plasticity. This is why early learning stages are very important.

Given the importance of early learning stages, it becomes critical to find methods that can help a network to avoid big mistakes during these early stages. To the extent that these errors are due to neural interference, a way to solve the problem consists in reducing neural interference. There might be different methods for reducing neural interference, such as introducing new units in the course of learning or introducing noise. One method we tried, i.e., preventing the activation state of units from reaching extreme values, does not eliminate interference but it can avoid its negative consequences in the specific case of the backpropagation procedure. On the contrary, sequential learning is a general method that can prevent in all cases the negative effects of neural interference.

6. Discussion

Neural interference may prevent a neural network to learn two or more distinct abilities or to learn them well. Modularity is one way to solve the problem of neural interference. In fact, training neural networks to exhibit both an ability to recognize the identity of an object and an ability to recognize the object's spatial location results in better learning with modular than with nonmodular networks.

An analysis of neural interference in backpropagation learning shows that neural interference is greater during the early learning stages. Backpropagation learning implies that connection weights are told to change, either increasing or decreasing their current weight value, as a function of the quantitative discrepancy (error) between the actual and the desired activation level of the units that are activated by them. In the early stages of learning errors tend to be larger and therefore pressures to change (error messages) have larger absolute value. As a consequence, if one and the same connection subserves two distinct abilities and there is neural interference between contradictory instructions to change, the negative effects of neural interference will be greater during the early than during the later stages of learning. This implies that if a neural network must acquire two distinct abilities at the same time and the two abilities are of different complexity and take different time to learn, the less difficult ability will be learned better than the more difficult one.

Are nonmodular networks absolutely prevented from learning two or more distinct abilities? The answer is no. Another solution to the problem of neural interference is sequential learning. Sequential learning can allow even nonmodular networks to learn two or more distinct abilities. In sequential learning a network first learns only a single ability and therefore without interference from the other ability. When the first ability has been learned, the network starts learning the second ability. Notice however that when the network starts learning the second ability the network still receives error messages with respect to the first ability and therefore the learning of the first ability is not discontinued. Otherwise, if the learning of the first ability is entirely discontinued when the network starts learning the second ability, catastrophic forgetting occurs: the second ability is acquired but the first ability is lost. The reason for catastrophic forgetting is that in a nonmodular network in which one and the same connection weight may subserve both abilities the changes in connection weights that are required to learn the second ability tend to disrupt the first ability. This disruption is avoided if the network continues to receive error messages also with respect to the first ability which are taken into consideration in deciding how to change the value of the connection weights. This of course implies that there may be some neural interference between error messages when the network starts learning the second ability. However, the advantage of sequential learning is that this neural interference tends to be weak. When a neural network starts learning two distinct abilities at the same time, i.e., nonsequentially, neural interference tends to be great because both abilities are in their early learning stages and therefore error messages are quite large for both abilities. When a network learns two abilities one after the other, there is no neural interference during the learning of the first ability and, furthermore, neural interference is weak when the network starts learning the second ability without discontinuing the learning of the first ability. In fact, what makes learning two tasks at the same time difficult is that in the first stages of learning error messages are large and therefore in these early stages the network tends to receive potentially contradictory error messages that are both large. The situation is different if the network learns the two abilities sequentially. In the first stages of learning of the second ability the error messages for the second ability will be large but those for the first ability will be small. In the later stages both error messages will be small. Hence, the negative effects of neural interferences are reduced and the network can learn both tasks.

This analysis also explains our result according to which in learning two tasks sequentially it is better to learn the more difficult task first and then the more difficult task, than the other way round. Task difficulty is reflected in the size of error messages, with more difficult abilities being intrinsically associated with larger error messages. If the more difficult ability is learned first, this means that the ability which generates larger error messages can be learned without interference. Furthermore, when the network starts learning the second, less difficult, ability, this easier ability will generate for intrinsic reasons smaller error messages and there will be little interference with the small error messages associated with the already learned more difficult ability.

7. Conclusions

Virtually all organisms of some complexity must learn to do several different things. An organism might need to learn to recognize several different visual and auditory patterns, to learn where and when certain events or entities are likely to occur, or to learn how many things occur together. Sometimes these different learning tasks might occur in different biological substrates, but often the same neural circuits are learning to do different things. This paper has examined the question of how the presence of multiple tasks interacts with learning architectures and the flow of information through those architectures. It has approached the question by using the idealization of an artificial neural network where it is possible to ask more precisely about the effects of modular versus nonmodular architectures as well as the effects of sequential vs. simultaneous learning of tasks.

While prior work has shown a clear advantage of modular architectures when the two tasks must be learned at the same time from the start, this advantage may disappear when one task is first learned to a criterion before the second task is undertaken. Nonmodular networks, in some cases of sequential learning, achieve success levels comparable to those of modular networks. In particular, if a nonmodular network is to learn two tasks of different difficulties and the more difficult task is presented first and learned to a criterion, then the network will learn the second easier one without permanent degradation of the first one. In contrast, if the easier task is learned first, a nonmodular task may perform significantly less well than a modular one.

It seems that the reason for these difference has to do with the fact that the sequential presentation of the more difficult task first minimizes interference between the two tasks. Because fewer weight changes and less overall restructuring must occur when the simpler task occurs second, the network that has developed to execute the first more difficult task is able to recover from the perturbations introduced by the second task and reach a criterion level in both tasks. By way of analogy, imagine that one is trying to learn how to both juggle three balls and ride a unicycle. One can either learn to juggle first and then add unicycle riding, or learn to ride first and then add in the juggling. The two tasks clearly interfere and the unicycling is generally much more difficult to master than simple juggling.

Assume further that early in the learning process for difficult tasks such as unicycling there are points where learning is very difficult. It is clear that it would be almost impossible for learners to pass these points if they have to continue to learn another task, though a more simple one, such as juggling a few balls. In other words if one learn juggling first and then tries the unicycle, the interference created by the juggling may increase the difficulty of the unicycle in the initial phase up to an impossible level. But if one is riding the unicycle first, once that difficult learning points are crossed and the task becomes easier, it is not hard to then add in the juggling.

What are the implications of serial versus simultaneous learning in actual organisms? How often, for example, does an organism encounter one task first, learn it to mastery and then start on a separate task? Is it more common for the organism to have to learn both tasks together from the start? In the case of the what versus where task it seems likely that both tasks occur together from the start. One rarely is receiving just one form of information without the other. If so, then it may be difficult to envision a naturalistic situation in what and where learning where a nonmodular network could do as well as a modular one. But, with other tasks, sequential presentations of the tasks may not only be plausible, but the norm.

A second issue concerns the need for the first task in the two task sequence to be the more difficult one for the nonmodular networks to succeed. This constraint poses an interesting problem of how networks might learn in the course of an organism's development. It is normally assumed that the more immature an organism, the lower the level of task difficulty or complexity that it is able to master. Thus, naturalistically in development, it might seem that organisms as they develop tend to learn simpler tasks before more complex ones. If this is indeed a general pattern, it suggests that modular networks might have a distinct advantage for the learning problems confronted by young organisms, but that non modular networks might thrive in more mature organisms where the natural order of task difficulties might be reversed.

More broadly, the studies summarized in this paper make it clear no one learning architecture is optimal in all situations. Modular architectures, whether pre-wired, or acquired through genetic algorithms, can have distinct advantages over nonmodular ones in some multitask environments, but not all. The challenge now is to describe in more precise terms those cases where modular architectures have an advantage and to understand the implications of such findings in artificial systems for natural systems both in their mature and developing forms.

References

Calabretta, R., Di Ferdinando, A., Wagner, G. P. & Parisi, D. (2003). What does it take to evolve behaviorally complex organisms? *BioSystems* 69/2-3, 245–262.

Di Ferdinando, A., Calabretta, R. & Parisi, D. (2001). Evolving modular architectures for neural networks. In R. French & J. Sougné (Eds.), *Proceedings of the Sixth Neural Computation and Psychology Workshop Evolution, Learning, and Development*, pp. 253-262, London: Springer Verlag.

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

LeDoux, J. (2001). The synaptic self. Vicking, New York.

Milner, A. D. & Goodale, M. A. 1998. The visual brain in action. *PSYCHE*, 4(12) (http://psyche.cs.monash.edu.au/v4/psyche-4-12-milner.html).

Plaut D. C. & Hinton, G. E. (1987). Learning sets of filters using backpropagation. *Computer Speech and Language* 2, 35-61.

Rueckl, J. G., Cave, K. R. & Kosslyn, S. M. (1989). Why are "what" and "where" processed by separate cortical visual systems? A computational investigation. *Journal of Cognitive Neuroscience* 1, 171-186.

Ungerleider, L. G. & Mishkin, M. 1982. Two cortical visual systems. In Ingle, D. J., Goodale, M. A. & Mansfield, R. J. W. (Eds.), *The Analysis of Visual Behavior*. The MIT Press, Cambridge, MA.

Velichkovsky, B.M. (2007) Towards an Evolutionary Framework for Human Cognitive Neuroscience. *Biological Theory 2*(1), 3-6.