Auto-teaching:
networks that develop their own teaching input

Stefano Nolfi Domenico Parisi

Institute of Psychology
National Research Council - Rome
E-mail: stiva@irmkant.Bitnet
domenico@irmkant.Bitnet

Abstract

Back-propagation learning (Rumelhart, Hinton and Williams, 1986) is a useful research tool but it has a number of undesiderable features such as having the experimenter decide from outside what should be learned. We describe a number of simulations of neural networks that internally generate their own teaching input. The networks generate the teaching input by trasforming the network input through connection weights that are evolved using a form of genetic algorithm. What results is an innate (evolved) capacity not to behave efficiently in an environment but to learn to behave efficiently. The analysis of what these networks evolve to learn shows some interesting results.

Introduction

In many supervised learning models of neural networks a vector of activity values is provided as an external teaching input to the network. Each activity value in the vector is compared with the computed value of the corresponding output unit and the resulting error is used to modify the connection weights.

Notwithstanding its usefulness for training networks, an external teaching input is not very plausible biologically and has a number of questionable properties.

(1) In nature it is very unlikely that an organism has an external teacher that decides what is the performance that is expected from the organism and describes this performance to the organism so that it can learn.

(2) More generally, it is unclear what is the origin of teaching input. In a sense, a teaching input is a statement of a goal performance that an organism is supposed to learn to perform. But one would like to have organisms develop their own goals without being told from outside what these goals should be.

(3) Teaching inputs are static objects that remain always the same for each given input (unless of course the experimenter changes them), are not generated by the network itself, and do not adjust themselves as a function of circumstances. On the other hand, goals emerge, develop, and are learned.

4) From a purely algorithmic point of view, often the experimenter does not know

the correct solution for a task. One would like the network itself to find a solution to the problem. The experimenter may be able to evaluate the appropriateness of a solution that the network has found but he or she cannot find good solutions himself or herself and neither can he or she specify what a network that incorporates such as solution should do in every situation. More concretely, in some cases the experimenter may not be able to provide the network with the appropriate set of input-teaching input pairs. Zipser (1990) considers two cases, one in which the function that generates these pairs is known and can be used to generate the teaching input, and the other in which "no function is known but where empirical input-output data is available" which "can be used to train the network" (Zipser, 1990 p.357). However, there is a third case in which neither the function nor the empirical data are available and still we may want the network to learn.

Learning algorithms vary in the amount and nature of the feedback required. Fully supervised paradigms such as backpropagation (Rumelhart, Hinton, and Williams, 1986) supply immediate and detailed correct answers as feedback. Reinforcement paradigms (Sutton, 1984) supply only judgments of right or wrong. Simulated annealing algorithms (Kirkpatrick, Gelatt and Vecchi, 1983) and genetic algorithms (Holland, 1975), applied to neural networks, supply still less, only a global evaluation of network's performace. Unsupervised paradigms (Hopfield, 1982; Kohonen, 1982) do not require supervision at all and therefore avoid these problems. On the other hand, unsupervised learning models cannot learn arbitrary functions and appear to be restricted to tasks in which what must be learned is basically the statistical properties of the inputs (Zipser, 1990, p.358).

Nolfi, Elman, and Parisi (1990) described networks that simulate organisms moving in an environment and looking for food elements. Using a population of networks and a genetic algorithm (i.e. selective reproduction and random mutation of some of the connection weights) we showed that these networks can find good solutions to the problem of approaching food. Although they were not told what the appropriate output (movement in the environment) was for each input (sensory information about food location), there was evolutionary emergence of a capacity to approach food, i.e. to respond with an appropriate output to each input. The only supervision here concerned the choice of which organisms were allowed to reproduce and which were not (organisms that ate more food elements during their life-time generated offspring) but this is exactly what natural selection provides to real organisms. In addition to being selected for eating ability, our networks were individually taught during their life to predict how the food location changed relative to the organism as the organism moved in space. This learning had a positive influence on the evolution of the food approaching capacity. We used backpropagation to teach these networks to predict since it is the environment itself that provides the teaching input in this type of prediction learning.

However, in other cases organisms appear to generate internally their own feedback for learning and to evolve autonomously a specification of what to learn. Ackley and Littman (1991) have simulated the evolution of very simple organisms similar to those just described using reinforcement learning in addition to evolution. The organisms are represented by neural networks whose behavior is evaluated during life just as right or wrong (positive evaluations result in a reinforcement and negative evaluations in a change of the current weights). What it is important to

notice is that the supervision required for networks' evaluation, in this model, is also evolved through the evolutionary process so that the life-learning process does not require any other supervision than the fitness-based selection of the pure evolutionary model.

In this paper we will present a similar approach in which fully supervised learning (and not just reinforcement learning) is obtained through evolution (i.e. without any external supervision other than networks' selection). Networks are not told from outside what is the desired performance they should learn to approximate, but they evolve the capacity to internally generate such "auto-teaching" input.

Teaching units

Consider a typical back-propagation network. The network is made up of a layer of input units, a layer of outputs units, and one or more layers of hidden units. A number of patterns are sequentially applied to the input units and for each pattern the network computes the corresponding pattern on the output units. When learning starts the connection weights are random so that an arbitrary output pattern is computed for each input pattern. For each input pattern, however, the network is provided with a desired pattern of activation values on the output units (teaching input). When an output pattern is generated each computed activation value is compared with the corresponding correct value and the resulting difference is called "error". The error is back propagated through the network leading to error-reducing modifications of the connections weights. After a number of learning "epochs" (runs of all the input patterns) each input pattern generates an output pattern which asymptotically approximates the desired performance. The network has learnt.

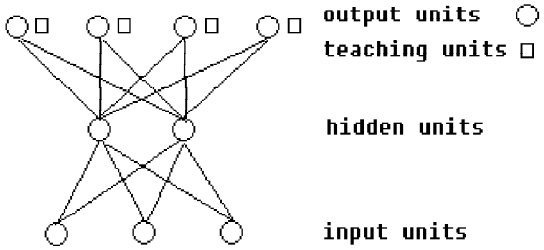

Figure. 1. Architecture of a feedforward network with one hidden layer. Teaching units are represented as little squares; each teaching unit is at the right of the output unit that it teaches.

If we represent the teaching input as a vector of activation values on a set of units each corresponding to one output unit and we call these units teaching units, it is correct to say that these units are "external" to the network (Figure 1). In fact, the teaching units are not linked via connections to the other units of the network. As a consequence, the activation value of these teaching units is decided from outside, i.e. by the experimenter.

In order to eliminate these properties of the traditional notion of teaching input we take two steps. The first step is to make the teaching units part of the network. We will assume that there are regular connections linking the other units of the network to the teaching units. This can be done in a number of different ways. The simplest case is that in which an additional set of hidden units is provided (Figure 2). We will call these units which are connected to a network and generate teaching input for the

network "auto-teaching" units and the teaching input they generate "auto-teaching" input.

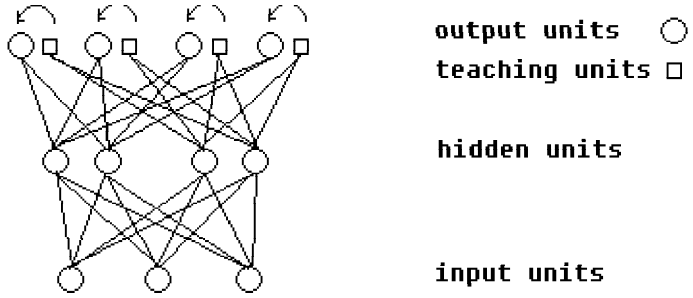

Figure 2. Auto-teaching network. An additional set of hidden units receive connections from the standard input units and send connections to the teaching units. This is the teaching sub-network.

Consider now what would happen in a network such as (2) if one applies an activation pattern to the input units. The network would compute activation values for the hidden units first (both for the hidden units of the standard sub-network and for the hidden units of the teaching sub-network), and then the activation values for the output units and for the teaching units. As soon as these latter activation values have been computed the network is ready to modify the connections weights of the standard sub-network on the basis of the discrepancy (error) between the activation value of each of the teaching units and the activation value of the corresponding output unit. Back-propagation can be applied as usual and will lead to new connection weights in the standard sub-network. The connection weights of the teaching network will remain the same.

However, it is clear that the network would not be learning. All the initial weights are randomly chosen and therefore the teaching input to the standard network, i.e. the computed activation values of the teaching units, is arbitrary. As a consequence, any resulting change to the connection weights of the standard sub-network would be similarly arbitrary and it wouldn't lead anywhere. We need to find out how to make our networks develop useful teaching input.

We think that an appropriate mechanism for selecting connection weights that can generate good teaching inputs in our type of networks could be a process similar to evolution. Training networks by using evolutionary methods is being actively studied at present (Hinton and Nowlan, 1987; Goldberg and Holland, 1988; Belew 1989; Nolfi, Elman and Parisi 1990; Belew, McInerney and Schraudolph, 1990; Miller and Todd, 1990). With evolutionary methods networks that are able to produce a given performance are not developed by individual training and weight modification, but by selective reproduction and random variation (or crossover). We will show that this method of genetic training can be applied to our networks with internal teaching units in order to develop networks that generate good auto-teaching.

The problem

Let us immagine that our goal is to create an organism (O) that is able to find and eat food in its environment. An O's environment is a two-dimensional square divided

up into cells. At any particular moment O occupies one of these cells. A number of food elements are randomly distributed in the environment with each food element occupying a single cell. O has a facing direction. We shall imagine it has a rudimentary sensory system that allows it to receive as input from the environment the angle (relative to where O is currently facing) and the distance of the nearest food element. We shall also equip O with a simple motor system that provides it with the possibility, in a single action, to turn any angle from 90 degrees left to 90 degrees right and then move from 0 to 5 cells forward. Finally, when O happens to step on a food cell, it eats the food element which disappears.

The basic network underlying O's behavior is a shown in Figure 3.

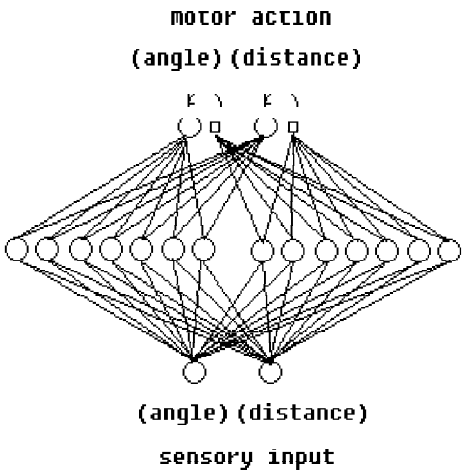

Figure 3. Os' Architecture. The 3-layer network is made up two sub-parts, a standard network and a teaching network. The two sub-networks share input units but each has a separate set of seven hidden units and two output units.

The output units of the standard network are interpreted as generating the motor actions while the output units of the teaching network (i.e. the teaching units) are interpreted as generating teaching input for the corresponding output units. Sensory input is encoded by the 2 input units representing the angle and the distance of the nearest food element (both values are scaled from 0.0 to 1.0). Movement is encoded in the 2 motor output units that specify amount and direction of turn and length of the step forward, respectively (these two values are also scaled from 0.0 to 1.0). Each network is initially assigned random weights, selected between -1.0 and 1.0, on all its connections .

When O is placed in the environment that has been described above, a sequence of events will occur. Sensory input is received on the input units. Activation flows from the input units to both the hidden units of the standard network and to those of the teaching network and then from these hidden units separately to the output units of the standard network (coding actual movements) and to the output units of the teaching network (coding teaching input for the standard network). The values on the two output units are then used to move O in the manner specified by these values, thereby changing the sensory input for the next cycle. The backpropagation algorithm uses the discrepancy between the two sets of output units to change the connection weights of the standard network.

Genetic adaptation

We begin with 100 Os each having the same architecture and a different random assignment of connection weights. This is Generation 0 (G0). G0 networks are allowed to "live" for 20 epochs, where an epoch consists of 250 actions in 5 different environments (50 actions in each) for a total of 5000 actions. The environment is a grid of 40x40 cells with 10 pieces of food randomly distributed in it. The Os are placed in individual copies of these environments, i.e. they live in isolation.

At the end of their life (5000 actions) Os are allowed to reproduce. However only the 20 Os which have accumulated the most food in the course of their random movements are allowed to reproduce by generating 5 copies of their weight matrix. These 20x5=100 new Os constitute the next generation (G1). Mutations are introduced in the copying process by selecting at random 4 weights of the standard sub-network and 4 weights of the teaching sub-network. A random value between +1.0 and -1.0 is added to each selected weight value (this implies that weights can increase in size evolutionarily). Inheritance is strictly Darwinian. A reproducing network transmits its inherited weight matrix to its offspring, without any of the changes in the standard sub-network that result from lifetime learning.

After Os of G1 are created they are allowed to live for 5000 cycles. The behavior of these Os differs slightly from that of preceding generation (G0) as a result of two factors. First, the 100 Os of G1 are the offspring (copies) of a subset of the Os of G0. Second, the offspring themselves differ slightly from they parents because of the small mutations in their weights. These differences lead to small differences in mean food eaten by the Os in G1. At the end of their life the 20 best individuals are allowed to reproduce 5 times, forming G2. This process continues for 200 generations.

Simulation 1

In this first set of simulations we used a population of Os with the architecture shown in Figure 3. Mutations were applied both to the standard weights and to the teaching weights (4 standard weights and 4 teaching weights randomly selected are mutated). Each O had a lifespan of 250 movements in 5 different worlds (50 movements in each world) for a total of 5000 movements. At the end of a generation, the 20 Os which had gathered the most food were allowed to reproduce by generating 5 offspring. A learning rate of 0.15 was used.

One first question that this simulation was expected to answer was if the evolutionary process could cause the emergence of teaching weights generating non-arbitrary auto-teaching input. Can mutation and selective reproduction generate, generation after generation, teaching input so that learning based on it leads to a better eating capacity? In other words, can evolution be at the base of an auto-teaching capacity?

The answer to this question is Yes as is shown in Figure 4. There is no improvement in eating performance during life for the first 20 generations. However, later in the evolutionary process eating increases after the very first epochs of life and this increase is greater with successive generations of Os. This implies that evolution progressively selects networks that internally generate teaching input which can be

used by the networks themselves to learn during their life how to search for food efficiently.

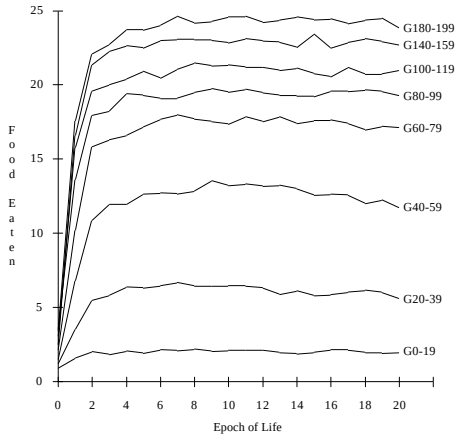

Figure 4. Eating performance of the best individuals of successive generations across the epochs of their life (performance at epoch 0 of life, i.e. at birth, is calculated by letting Os live for one epoch without having any auto-teaching). For pratical reasons we show the average results of the best Os of a group of successive generations by a single curve. Each curve represents the average performance of 10 different simulations with different random assignments of weights.

Figure 4 also implies that the evolutionary process, in addition to producing good auto-teaching capacity, can also cause an increase in the eating capacity with successive generations of Os. However, it is an open question how this increase would compare with the increase we would obtain without auto-teaching or, more generally, without any kind of learning during life. We then run a new set of simulations with a network architecture that only include the standard network. While the simulation without auto-teaching produces a better evolutionary curve in the first 60 generations (see Figure 5), after that there is a clear advantage of the Os with the more complex architecture and learning during life. It appears then that the simpler evolutionary process that operates on the standard weights only, i.e. directly on the eating capacity itself, is quicker in producing results but afterwards it reaches a lower local maximum. On the other hand, the more complex selection process that operates on both the standard weights and the teaching weights is slower but finally produces a higher maximum. (For an explanation of how learning can help evolution see Hinton and Nowlan (1987) and Nolfi, Elman and Parisi (1990)).

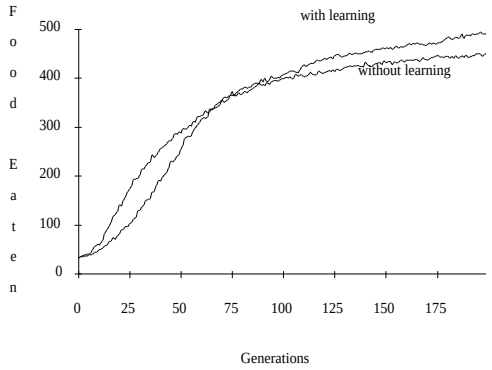

Figure 5. Increase in eating capacity of the best individuals across 200 generations for the simulations with auto-teaching versus a simulations without it. Each curve represents the average performance of 10 different simulations.

The evolutionary process operating on Os that learn during their life can produce more fit Os in two different ways: (a) it can develop Os with better and better standard weights, i.e. Os that eat more at birth with respect to the individuals of the preceding generations, and/or (b) it can develop Os with better and better teaching weights, i.e. Os that generate increasingly better auto-teaching input. This traslates in better learning and therefore in better eating performance.

To test which of these two possibilities is the correct one, i.e. what is the respective role or importance of the two sub-networks in the evolutionary process, we took the best individuals of each generation and we let them move in their environment without any learning. In other words, we tested the networks' performance at birth. If we rule out learning (see Figure 6), performance abruptly decrease. This means that the Os in this simulation are not selected for their ability to approach food. They are selected for their ability to learn to approach food, as also Figure 4 shows.

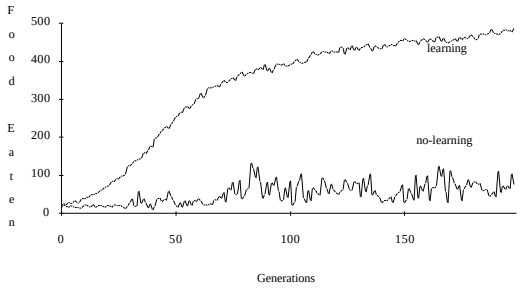

Figure 6. Performance with and without life-learning as a function of generations. The "learning" curve represents the performance of individuals with life learning and the "no learning" curve represents their performance at birth, i.e. without any learning. Each curve represents the average performance of 10 different simulations.

We have attributed so far the evolutionary increase in how much is learned during life across generations (see Figure 4) to the evolutionary emergence of better and better auto-teaching input. In other words, what is being selected are the weights of the teaching sub-network. At this point we might ask if the standard network plays any role at all in the evolution of the eating capacity which is shown in Figures 5. In fact, it could be the case that the standard weights are not selected in any way and therefore they have no role in such evolution. To test this further hypothesis we tested the same 5 best Os of each generation with life learning, but we first randomized their standard weights at birth. If these weights play no role at all, then the only thing that evolves in our Os would be the teaching weights generating better and better teaching input. In such a case we would have an equally good development of eating capacity with initially randomized standard weights. Figure 7 shows that this is not the case. The curve that is obtained demonstrates that by randomizing the standard weights at birth we destroy the O's capacity to learn to approach food. This result tells us that the standard weights also are selected by evolution and they play a crucial role from the point of view of Os' behavior. What appears to be happening is that the standard weights are not selected for directly incorporating good eating behaviors but they are accurately selected for their ability to let such a behavior emerge through life learning. In other words, the increased learning across generations that we observe in

Figure 4 is due to both an increased capacity to generate good auto-teaching input (selection of teaching weights) and to an increased capacity to learn from this auto-teaching input (selection of standard weights).

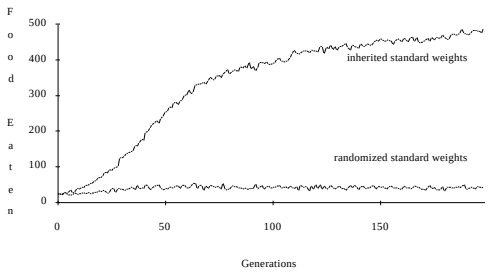

Figure 7. Performance with inherited and with randomized standard weights as a function of generations. Each curve represents the average performance of 10 different simulations.

Teaching internal units

In networks of type (2) auto-teaching units are provided for the output units, that is, for the units that interact with the external environment. Hence, these Os can evolve (and actually evolve) a capacity to learn to perform appropriate actions. On the other hand, we can suppose that Os could learn some other task which is not directly related to their external behavior but can indirectly help to develop such external behavior.

Biological organisms receive a huge amount of information that they can use to learn useful things. Perception of the world changes in time as a consequence of the Os' actions themselves and of intrinsic changes of the world. From those changes very complex things such as world maps and object classification can be learned using as teaching input the input pattern itself (auto-associative learning; see Willshaw, 1981) or the input pattern at time t+1 (prediction learning; see Elman, 1990; Weigend, Huberman and Rumelhart, 1990). Learning this kind of tasks require a full supervised learning, and not a reinforcement learning like that used by Ackley and Littman (1991), because in this case the environment provides a very detailed feed-back.

How can we obtain this type of detailed learning without external feedback? We would also like to find a way for not deciding in advance which is the correct task to learn. Consider the standard back-propagation model in which the teaching input is decided by the experimenter. In such a model it is impossible to teach internal units. The function of an internal unit (i.e. what it represents) is completely unknown before the network finds a useful set of weights, and even after that, such a function usually is not easily interpretable. Hence, we cannot know what kind of teaching input we should use for teaching these internal units. On the other hand, in networks that develop their own teaching input like our auto-teaching networks it makes sense to have internal units that have teaching inputs (i.e. internal units that have their own corresponding teaching units). What these internal units will eventually represent and what teaching input they must receive is not known but, since the kind of teaching input that these units receive changes as a consequence of mutation and selection, we might expect that a useful teaching input (and so a useful function) for these units can be found.

Let us consider the network shown in Figure 8. The network has 2 output units that codify the actions that the organism performs in the environment. In addition, there are three other units that are identical to the output units from the point of view of the connectivity pattern but that do not codify any action of the organism or, more generally, any desired or interpretable output. We call these units pseudo-output units. The activation values that these three units should have for each input pattern is simply the activation values of the three corresponding auto-teaching units. Because the networks start with random weights, these auto-teaching units, at the beginning, have completely random values for each input pattern. However, selection and mutation should produce more and more useful values for them. And after a certain number of generation the three teaching units might begin to teach some useful task, i.e. a task that helps Os to develop more appropriate external behavior.

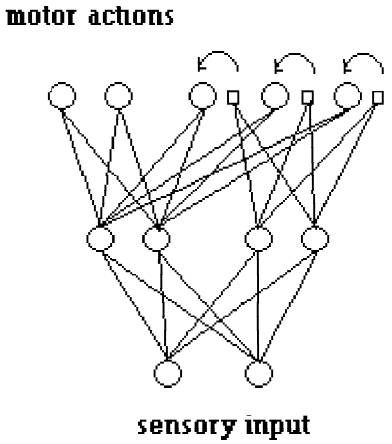

Figure 8. Auto-teaching network. Three internal units receive teaching input while the two output units do not.

Why should auto-teaching of internal units be useful? Why should learning to perform a task that does not have any direct effect on the network actions make the network more fit? The input stimuli from the environment are elaborated by the weights that go from the input units to the teaching network's hidden units and from these hidden units to the auto-teaching units. The teaching input which is generated produces a change in the weights of the standard sub-network and, what is critical, in the weights that go from the input units to the standard network's hidden units. As a consequence, the internal representation of the input stimuli changes during life and, because of the evolutionary process, these changes may have a tendence to improve Os' behavior. (For a more detailed analysis of how learning a task that does not have a direct effect on the network's output could increase performance see: Nolfi, Elman, and Parisi (1990), Parisi, Cecconi, and Nolfi (1990)).

Simulation 2

In a second set of simulation we used a population of Os with the architecture shown in Figure 9. These Os have a standard network with two additional output units which don't have any effect on the world. These two pseudo-output units have associated two auto-teaching units that generate teaching input for them. The teaching sub-network shares, as usual, the input units with the standard sub-network and has

its own set of seven hidden units.

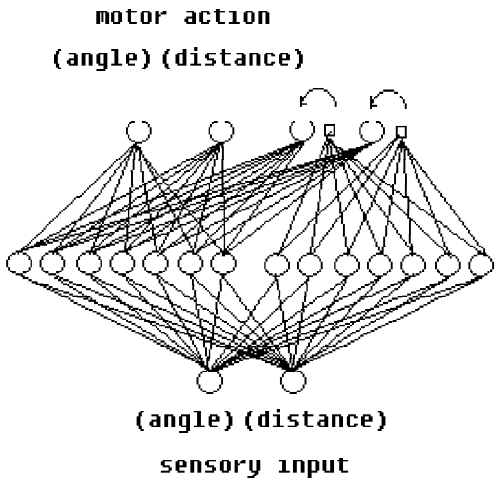

Figure 9. Architecture of Os of Simulation 2. Auto-teaching is provided for two pseudo-output units instead that for the two motor output units as in Simulation 1.

Because of this architecture, the weights that connect the input to the standard hidden units but not the weights that connect these hidden units with the two (real) output units are changed during life. In other words, the auto-teaching changes the internal representation of the input stimuli within the standard network.

All other parameters remained the same as those of the previous simulation. Mutations were applied both to the standard weights and to the teaching weights (4 standard weights and 4 teaching weights randomly selected are mutated). Reproduction was based on the parent's weight matrix at the time of the parent's own creation. Each O had a lifespan of 250 movements in 5 different worlds (50 movements in each world) for a total of 5000 movements. At the end of a generation, those Os which had gathered the most food were allowed to reproduce by generating 5 offspring. A learning rate of 0.15 was used.

If we look at Os' performance during their life (see Figure 10) we see that in this case, in contrast with Simulation 1, there is only a very small increase in the eating performance during Os' life (such an increase concerns the very first epochs of life and reaches the largest amount from the 40th to the 59th generation). This fact seems to suggest that, in this simulation, learning has a very indirect effect on evolution (see paragraph 8 for a more detailed interpretation of this result).

As we already noted, if we compare this simulation with the first one we note an important difference. In the first simulation we know what the auto-teaching units are teaching to the standard network: they are teaching the correct movements in the environment since it is the movement-coding output units of the standard network that are taught by the auto-teaching units. By contrast, it is not clear at all what the auto-teaching units are teaching to the standard network in this simulation. We cannot directly interpret what is taught by the auto-teaching units since the standard network's pseudo-output units that are taught by the auto-teaching units in this simulation do not have any clear interpretation.

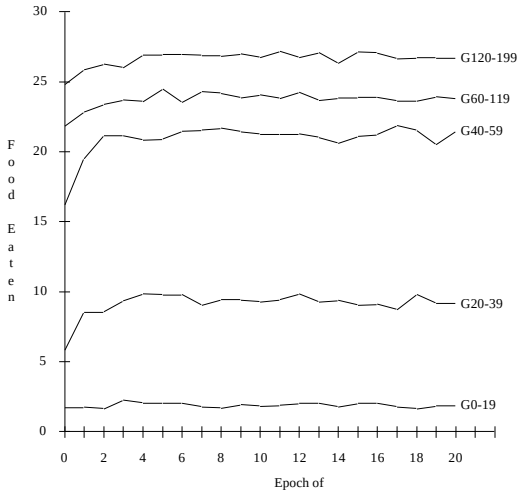

Figure 10. Eating performance of the best individuals of successive generations as a function of epochs of life. For pratical reasons we show the average results of the best Os of a group of successive generations by a single curve. Each curve represents the average performance of 10 different simulations.

However, we can make an attempt at determining in a more indirect way what teaching function is gradually developed by the auto-teaching units in the course of evolution. One first thing that we can do is to ascertain how much the activation values of the two auto-teaching units vary as a function of the different input patterns, generation after generation. At the beginning, since the random weights simply blur the input, we should expect the two auto-teaching units to have always approximately the same values for every input pattern. After a certain number of generations, if the teaching units have evolved some useful teaching function, we should expect an increase in the variability of the activation values of these units.

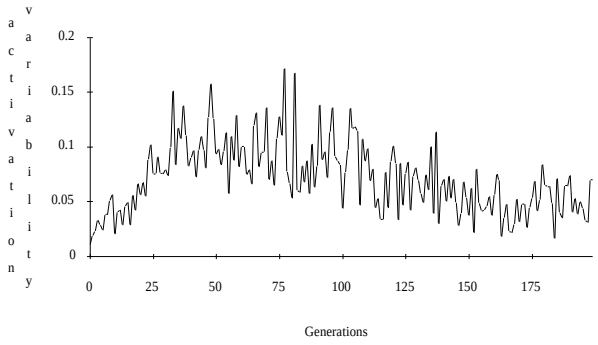

Figure 11. Variability (range of variation) in the activation values of the auto-teaching units of Simulation 2 across 200 successive generations. Average result of 10 different simulations.

Figure 11 shows that our expectations were correct. The variability in the activation values of the teaching units is rather low in the first generations and then it increases and reaches a peak after 75 generations. What was not expected is that such variability shows a decrease after that. If we look at the activation values of the two auto-teaching units in this later part of the evolutionary process, we find that often they have an activation value very close to 0.0 or 1.0 for every input pattern. At the same time also the activation values of the two pseudo-output units of the standard network which are taught by these auto-teaching units go to 0.0 or 1.0 since birth. We

can conclude that in the later part of the evolutionary process the auto-teaching units of most simulations have ceased to teach anything new to the standard network. The overall picture is one in which our networks take some time to evolve an auto-teaching capacity but then, after the auto-teaching units have helped in the evolutionary emergence of good movement strategies, these same units do not have any additional role to play and evolution cuts them off.

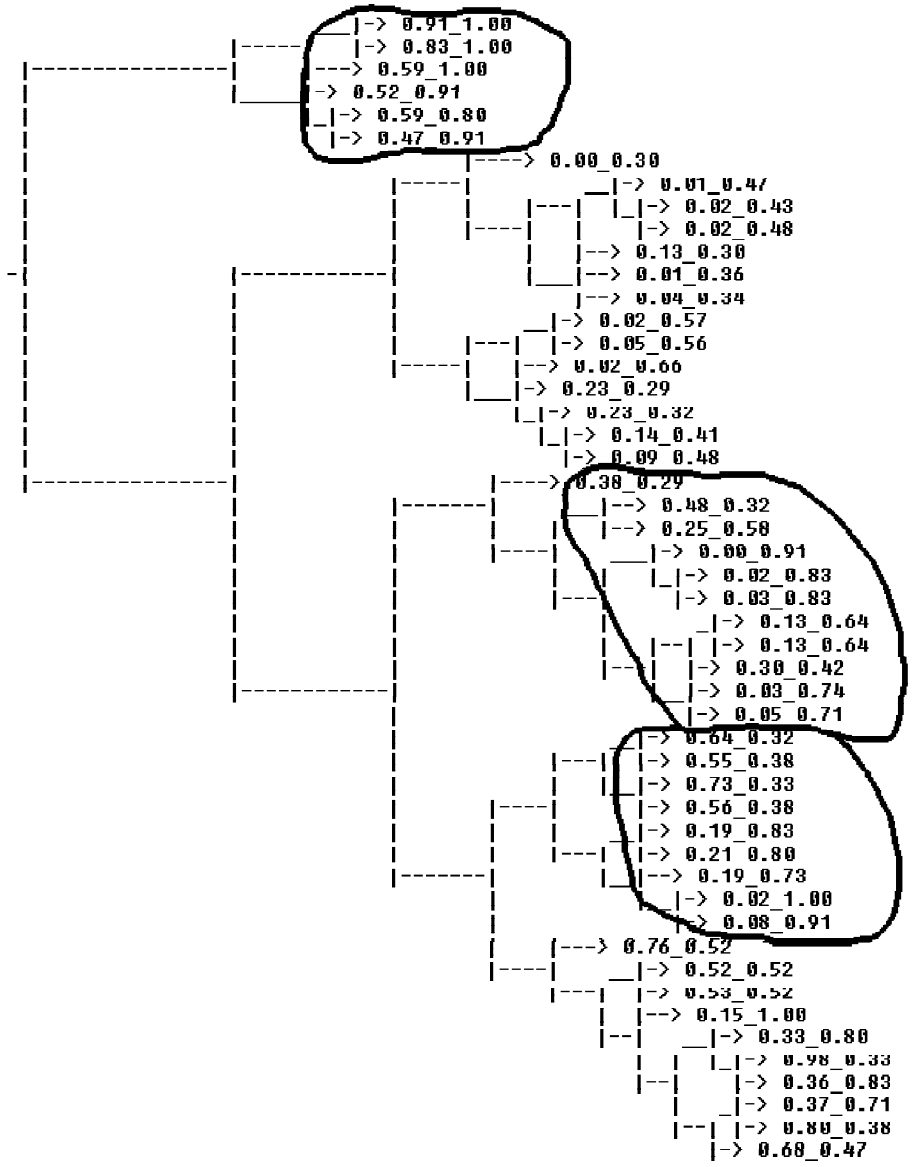

Figure 12. Cluster analysis (based on euclidean distance) of auto-teaching patterns of the best O of the 90th generation. Similar teaching patterns are represented by closer branches of the hierarchical tree. The first 50 patterns, corresponding to the first 50 spreadings of activation (cycles of life), were used. Notice that each teaching pattern is identified by the corresponding input pattern that has generated it. The first number refers to the activation value of the first input unit (coding the angle of the nearest food element) and the second number to the activation value of the second input unit (coding the distance of the food element).

These results concerning the variability of the activation values of the auto-teaching units are interesting but they do not say anything about the actual content of

what is taught by these units. One possibility is that the function of the auto-teaching in this simulation could be to help the standard network to develop good internal representations. In order to test this hypothesis we made a hierarchical cluster analysis of the teaching patterns generated by the teaching network of the best organism of several generations. One of the most interesting results is the one we obtained analyzing the best O of the 90th generation.

As Figure 12 shows, it is not always the case that similar input patterns generate similar teaching patterns. More particularly, the clusters indicated by circles contain dissimilar input patterns that generate similar teaching patterns. At this point we can ask if there is some relation between the manner in which different input patterns are classified together by the teaching network (in terms of the similarity of the teaching output they generate) and the type of correct movement output that these input patterns require. The response is yes.

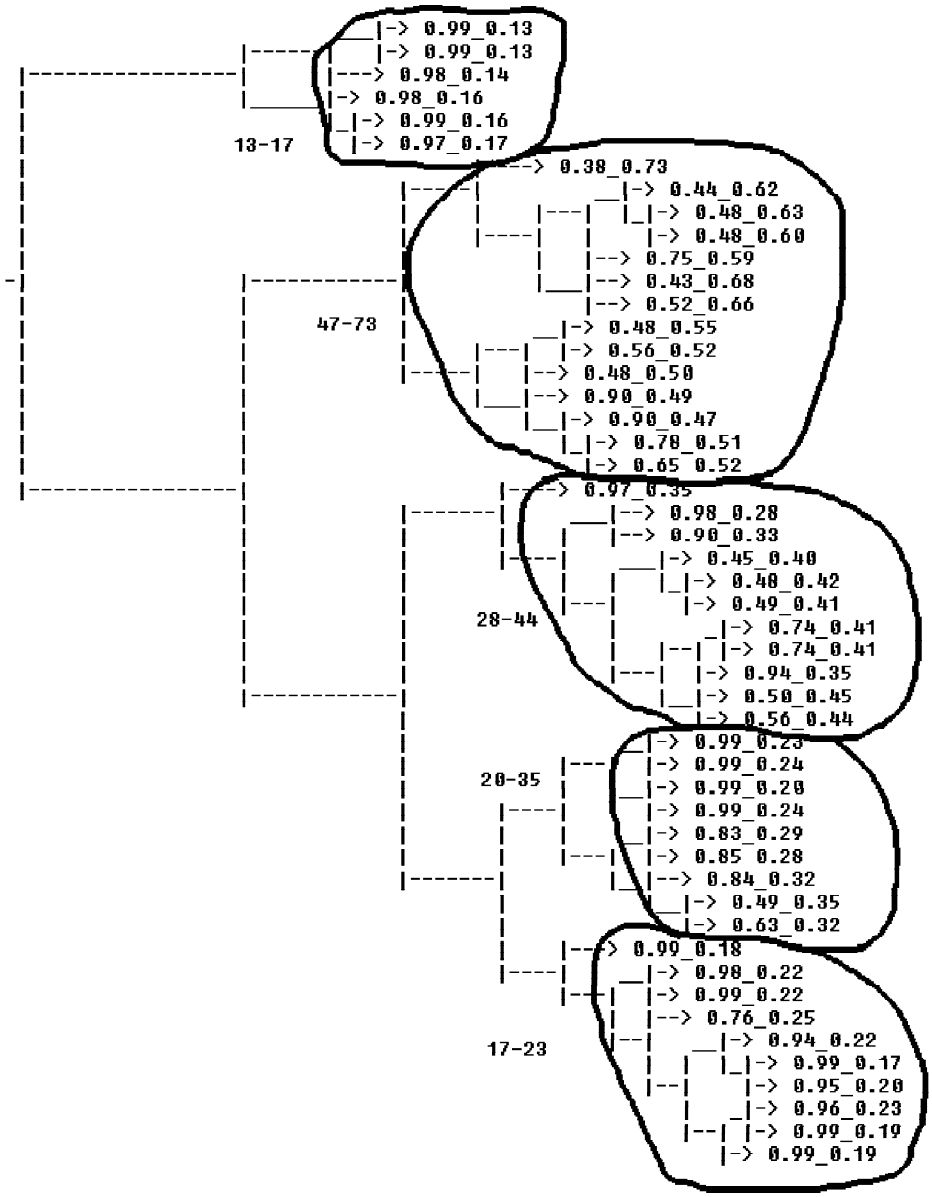

Figure 13. Cluster analysis of auto-teaching patterns of the best O of the 90th generation. The same first 50 patterns of Figure 12 were used. Each teaching pattern is identified by the corresponding

movement output pattern generated by the network. The first number designates the angle of the movement and the second number the length of the step forward.

As Figure 13 shows, input patterns that generate similar movement responses in an efficient organism such as the best individual of generation 90 also generate similar teaching patterns. More specifically, the circles show that sensory states that must generate steps forward of the organism of approximately the same length are classified together in the cluster analysis of the corresponding teaching patterns. We can conclude that the function of the auto-teaching mechanism in this simulation is to teach an individual organism to develop good internal representations of the sensory information so that appropriate movement responses to this information can be more easily selected by the organism.

Discussion

The general result of the simulations described in this paper is that we have been able to evolve networks with auto-teaching units. The selection and mutation process of evolution can produce networks that can internally generate their own teaching input. The individual learning that results from this teaching input during life has a facilitatory effect on the evolution at the population level of the capacity - food approaching - that constitutes the criterion for selection.

Furthermore, some more specific results appear in our data:

(1) What is inherited at birth is often interpreted as a capacity to behave, that is, to respond in effective ways to incoming stimuli. Networks of simulation 1 possess no such capacity at birth although the evolution of such a capacity is possible as is shown in simulations without life-learning. The connection weights they inherit from their parents do not allow them to respond to sensory information from the nearest food element in such a way that they approach the food. However, the evolution process has left an important trace in their inherited weights: a capacity to learn during life.

That the learning capacity is an evolved one is shown by the fact that there is no improvement in performance (no learning) during life in the first generations. However, as soon as the selection process has had an opportunity to demonstrate its power, the connection weights of both the standard and the teaching sub-networks are so selected that individual networks are born with an innate capacity to learn (their ability to approach food improves during their life). This increased learning capacity appears to be due to two complementary factors. One is the evolving capacity of the teaching network to generate useful teaching input for the standard network; the other is the evolving capacity of the standard network to take advantage of this teaching from the teaching network. No one factor without the other can explain the results that are obtained.

(2) Simulation 2 shows that the notion of auto-teaching can have a very abstract interpretation. In Simulations 1 what is taught by the auto-teaching units and what evolves across generations is the same thing: the capacity to approach food. This results from the fact that the movement output units of the standard network are directly taught by the teaching units. On the other hand, in Simulation 2 the notion of pseudo-output units that are taught by the teaching units allows us to detach whatever

capacity is taught by the teaching units from the actual output of the standard network, and therefore from the actual capacity in terms of which selection occurs. This makes the auto-teaching mechanism a very general and powerful one. In fact, we don't have to specify what is taught by the auto-teaching units. Any unit receiving activation from within a network can be taught by a corresponding auto-teaching unit in such a way that the learned changes in the connection weights leading to that unit can make the contribution of the unit to the network's final output a more appropriate one. In fact, the auto-teaching units of Simulation 2 could directly teach the hidden units of the standard network, without the indermediation of the pseudo-output units.

Our analysis of what the auto-teaching units of Simulation 2 actually teach points to a general interpretation of the function of auto-teaching units in this type of architecture. Auto-teaching units may be an evolved mechanism for helping a network to develop during life more appropriate internal representations for input data, that is, internal representations (activation values on the hidden units) that are more easily mapped onto the final output than the original input representations. Developing good internal representations when the input-output mapping is a complex one, i.e. when "the similarity structure of the input and output patterns is very different", is the fundamental function of hidden units (Zipser and Rumelhart, 1990, pag. 193). Auto-teaching units might be an evolved mechanism to accelerate the learning of good internal representations.

(3) Another interesting result of Simulation 2 is that the auto-teaching mechanism almost extinguishes itself in the last generations. In other words, there appear to be two phases in the evolution of the auto-teaching units. In the first phase, after a number of initial generations connection weights at birth appear to be so selected that an innate capacity to learn results. This is testified by the rapid increase in performance in the very first epochs of life of individuals belonging to these middle generations. However, this innate learning capacity does not last for the entire course of evolution that we have studied. There is a second phase of the evolutionary process in which the innate learning capacity seems to disappear.

This result is in contrast with what was obtained in Simulation 1, where learning during life continues throughout the evolutionary process. The reason appears to reside in the different network architectures in the two simulations. In Simulation 1 all the weights of the standard network are subject to learning through the auto-teaching mechanism. Hence, it is possible to gradually select teaching weights that can increase the standard weights' performance from null to a more fit value. In Simulation 2, on the contrary, only the lower connection weights of the standard network, those from the input to the hidden units, learn on the basis of the auto-teaching input. The higher weights, those from the hidden units to the output units, are not influenced by this learning since error is not backpropagated through them. Therefore, the higher weights must be selected to directly incorporate an appropriate mapping from the internal representations in the hidden units to the final motor output. One result of this different architecture and manner of functioning might be that, in Simulation 2, after a certain number of generations the standard network inherits "higher" weights that are appropriate and sufficient for executing the task (approaching food) and the learning mechanism associated with the auto-teaching units becomes superfluous or perhaps even conterproductive. As a result, the auto-teaching mechanism is estinguished by evolution. This might be an interesting feature

of an auto-teaching mechanism compared with the more usual learning based on an external teaching input. Auto-teaching can emerge evolutionarily when it is useful and it can annihilate itself when it becomes dysfunctional.

An important advantage of an innate learning capacity in natural organisms might emerge when organisms must adapt during life to varying and changing environments. We completely ignored this aspect of learning in our simulations since we used simulated environments that do not change either at the evolutionary scale or at the life-time scale. In fact, one possible extension of the present work on auto-teaching could be to demonstrate the importance of this type of mechanism in simulated organisms that live in changing environments or that themselves change their environment.

References

Ackley, D.E. and Littman, M.L. 1991. *Proceedings* of the Second Conference on Artificial Life. Addison-Wesley: Reading, MA.

Belew, R.K. 1989. Evolution, learning and culture: computational metaphors for adaptive algorithms. CSE *Technical Report* CS89-156. University of California, San Diego.

Belew, R.K., McInerney, J., Schraudolph, N. 1990. Evolving networks: using the genetic algorithm with connectionist learning. CSE *Technical Report* CS89-174. University of California, San Diego.

Elman, J.L. 1990. Finding Structure in Time. *Cognitive Science*, **14**, 179-211.

Goldberg, D.E., Holland, J.H. 1988. Genetic Algorithms. *Machine Learning*, **3**, 95-245.

Hinton, G.E., Nowlan S.J. 1987. How Learning Guides Evolution. *Complex System*, **1**, 495-502.

Holland, J.J. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: University of Michigan Press.

Hopfield, J.J. 1982. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*, U.S.A., **79**, 2554-2558.

Kirkpatrick, C.D., Gelatt, M.P., Vecchi, M.P. 1983. Optimization by Simulated Annealing. *Science*. **220**.

Kohonen, T. 1982. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 59-69.

Miller, G.F. and Todd, P.M. 1990. Exploring adaptive agency I: theory and methods for simulating the evolution of learning. In D.S. Touretzky, J.L. Elman, T.J. Sejnowski and G.E. Hinton (eds.), *Proceedings of the 1990 Connectionist Models*

Summer School. San Matteo, CA: Morgan Kaufmann.

Nolfi, S., Elman, J, and Parisi, D. 1990. Learning and evolution in neural networks. CRL *Technical Report* 9019. University of California, San Diego.

Parisi, D., Cecconi, F., Nolfi, S. 1990. Econets: Neural Networks that Learn in an Environment. *Network*, **1**, 149-168.

Parisi, D., Nolfi, S. and Cecconi, F. 1991. Learning, behavior, and evolution. In: Varela, F, Bourgine, P. *Toward a pratice of autonomous systems*. MIT Press.

Rumelhart, D.E., Hinton G.E., and Williams, R.J. 1986. Learning internal representations by error propagation. In D.E. Rumelhart, and J.L. McClelland, (eds.), *Parallel Distributed Processing*. Vol.1: Foundations. Cambridge, Mass.: MIT Press.

Sutton, R.S. 1984. Temporal credit assignment in reinforcement learning. University of Massachusetts. Departement of Computer and Information Science. *Technical Report* 84-2. Amherst, MA.

Weigend, A.S., Huberman, B.A., Rumelhart, D.E. 1990. Predicting the Future: a Connectionist Approach. *International Journal of Neural Systems*, **1**, 193-209.

Willshaw, D. 1981. Holografy, associative memory and inductive generalization. In Hinton, G. and J. Anderson (Eds.) *Parallel Model of Associative Memory*. Hillsdale: Lawrence Erlbaum Associates.

Zipser, D. 1990. Modeling cortical computation with backpropagation. In M.A. Gluck and D.E. Rumelhart (eds.) *Neuroscience and Connectionist Theory*. Hillsdale, N.J., Erlbaum.

Zipser, D. and Rumelhart, D.E. 1990. The neurobiological significance of the new learning models. In Schwartz, E.L. (Ed.) *Computational Neuroscience*. Cambridge, Mass.: MIT Press.