Extracting Regularities in Space and Time Through a Cascade of Prediction Networks: The Case of a Mobile Robot Navigating in a Structured Environment

Stefano Nolfi[1]
Institute of Psychology, National Research Council
Viale Marx 15, 00137 Rome, Italy
Email:stefano@kant.irmkant.rm.cnr.it
http://kant.irmkant.rm.cnr.it/nolfi.html

Jun Tani
Sony Computer Science Laboratory Inc.
Takanawa Muse Building, 3-14-13 Higashi-gotanda, Shinagawa-ku, Tokyo, 141 Japan
Email: tani@csl.sony.co.jp
http://www.csl.sony.co.jp/person/tani.html

Abstract

We propose that the ability to extract regularities from time series through prediction learning can be enhanced if we use a hierarchical architecture in which higher layers are trained to predict the internal state of lower layers when such states change significantly. This hierarchical organization has two functions: (a) it forces the system to progressively re-code sensory information so as to enhance useful regularities and filter out useless information; (b) it progressively reduces the length of the sequences which should be predicted going from lower to higher layers. This, in turn, allows higher levels to extract higher level regularities which are hidden at the sensory level. By training an architecture of this type to predict the next sensory state of a robot navigating in a environment divided into two rooms we show how the first level prediction layer extracts low level regularities such as 'walls', 'corners', and 'corridors' while the second level prediction layer extracts higher level regularities such as 'the left side wall of the large room'. The extraction of these regularities allows the robot to localize its position in the environment and to detect changes in the environment (e.g. the presence of a new object or the fact that a door has been closed).

1. Introduction

From the point of view of a natural or artificial agent the external environment does not provide any direct cue on how the agent should act to attain a given goal. However, the environment provides a rich feedback: the sensory states. Such information, by depending both on the agent motor action and on the environmental structure, may be used to extract regularities[2] from the environment which in turn may be useful to achieve the agent's goal. For example, the agent may learn of the consequences of different actions in different environmental contexts or it may learn to classify sensory states also on the basis of the preceding and following sensory patterns.

A straightforward way to use sensory information to extract regularities from the environment is *prediction learning* i.e. to try to predict what the next sensory state will

[1] Most of the work described in this paper has been done while Stefano Nolfi was visiting SONY-CSL.

[2] By regularities we mean a set of sensory or internal states which can be easily separated from the others and which correspond to agent-environmental states which are stable over space or time (i.e. which are predictable). Hopefully, the meaning of the term will become progressively more clear in the next sections.

be given the current sensory state and the motor action that the agent is going to perform (Parisi & Cecconi, 1995). That prediction learning can extract high level regularities from time series was first shown by Elman (Elman, 1990, 1993). He showed how by training a simple recurrent neural network to predict the next word in sentences of a pseudo-natural language, the network was able to extract high level regularities such as 'nouns' or 'verbs'.

Regularities can also be extracted in different manners without relying on the information provided by the next sensory states. For example, Floreano & Mondada (1996) showed how a robot may extract a representation of the external environment by using an evolutionary process in which individuals are rewarded for their overall ability to achieve a given task. In this case a robot was evolved for its ability to explore a simple arena while periodically returning back to a recharging station. However, this technique seems to work only in relatively simple cases. In the case of this work for example, the recharging station was illuminated and the robot was able to directly detect its relative position in most of the cases.

In this paper we will investigate how prediction learning can extract regularities from the external environment in the case of a mobile robot that navigates in a simple environment divided into two rooms. As we will see, regularities extracted in this way can be used by the robot to localize its position in the environment and to detect changes in the environmental topology such as the presence of a new object. In doing so we will show why a simple prediction network such as that described above is not enough to solve such tasks. A more complex architecture based on a cascade of prediction and segmentation layers in which regularities can be extracted at different levels is needed (see below).

1.1 What can and cannot be predicted

In practical cases, it is not possible to predict all the sensory information coming from the external environment for two reasons:

(1) Some sensory states or a part of each sensory state may be completely unpredictable. Consider, for example, the case of predicting the next sounds. While driving we can predict the intensity of the noise produced by the engine of our own car to a good extent on the basis of the gear into which we have shifted, the slope of the road etc. However, the sounds coming from the other cars in a traffic jam cannot be predicted at all.

(2) In general, only some features of the next sensory state can be predicted. For example, "Hearing the first two words of the sentence 'Henrietta eats …' allows you to infer that the third word probably indicates something to eat but you cannot tell what. The class of the third word is predictable from the previous words - the particular instance of the class is not." (Schmidhuber & Prelinger, 1993, pp.625).

Moreover we should consider that some of those features which can be predicted in principle may be difficult to predict in practice. In particular:

(3) The longer the time lag between an event that can be predicted and the features that can be used to infer its occurrence the harder the corresponding prediction is (Schmidhuber, 1992). The reason for that is that "the longer the time lag between an event and the occurrence of a corresponding error the less information is carried by the corresponding back-propagated error signals" (Schmidhuber, 1992, pp. 234).

(4) The training process may end up in a local minimum in which some of the sensory states or some features of the sensory states which in principle are predictable

may not be predicted correctly. In particular, this tends to happen when most of the states can be predicted easily (e.g. can be predicted by taking into account only the previous sensory state or few previous sensory states) while few other states can be predicted with greater difficulties. In such cases, as we will see, the learning process tends to converge on a solution in which the states which are easy to predict and are frequent are predicted correctly while the states which are more difficult and are infrequent are not (on the role of the learning experience regarding the outcome of the learning process see Elman et al., 1996). This type of problem is very general and might affect the result of the learning process in very different circumstances. However, its negative effects may be particularly relevant in the case of prediction learning in realistic circumstances in which sensory states might not change significantly for a long time (i.e. in cases in which the prediction task is for the most part trivial).

In other words, it may be that regularities have only an attenuated existence in a body of training data. If this is the case we are facing a so-called type-2 problem (Clark & Thornton, 1997). As claimed by Clark and Thornton the only way to solve these hard problems is to re-code the data in a way that ensures that regularities will have a higher statistical visibility.

Different methods may be applied to solve these problems. To solve the problem that some states can be unpredictable and that, in other cases, not all the details can be predicted (i.e. to solve (1) and (2)) Schmidhuber & Prelinger proposed to train a network to produce the predictable class of the next states instead of the states themselves. Sensory states were classified into predictable classes by a second network that try to classify the sensory states in classes that are predictable and still as specific as possible (Schmidhuber & Prelinger, 1993). To alleviate the problem of the time lag between events to be predicted and the features that can be used to infer their occurrence and the problem that most of the states may be easy to predict (i.e. to solve (3) and (4)) Schmidhuber proposed focusing on unexpected inputs and ignoring expected ones. This may be obtained by using two networks: one that tries to predict all sensory states and another that tries to predict the unexpected sensory states. The inputs which turn out to be unpredictable from the very first network are sent to a higher level network which in turn predict its next input operating on a slower, self-organizing time scale (Schmidhuber, 1992a; 1992b; for a variation of this idea in which unexpected states are identified by a network that predicts its own error see Schmidhuber, 1991).

1.2 A self-organizing hierarchy of prediction and segmentation layers

In this paper we propose an approach based on a hierarchy of prediction layers (Figure 1) which try to predict the next internal states of the lower layers (or of the sensory-motor states in the case of the very first layer). In this architecture, as in Schmidhuber & Prelinger (1993), higher layers are asked to produce a classification of the next sensory states and not the next sensory states themselves. The classification in this case is produced by the lower level layers that try to predict the next input states at their level. Moreover, in this type of architecture the sensory-motor information will be progressively transformed going from lower to higher levels. What we expect is that during the learning process weights will be modified so that in each transformation: (a) information will be compressed in time (each internal state, in fact, will be a function both of the current and of the next input state); (b) features which can be predicted at the current level will be enhanced while other features will be attenuated. To understand why, we should consider that, after learning, the internal representation in each

predicting layer will be a function both of the input state and of the output state (the internal representation is an intermediate step between the input state and the output state). At the end of learning the output state will tend to approximate the teaching state. However, this will be true only for states which are correctly predicted. Therefore, the internal state will be more affected by states which can be predicted at the current level[3].

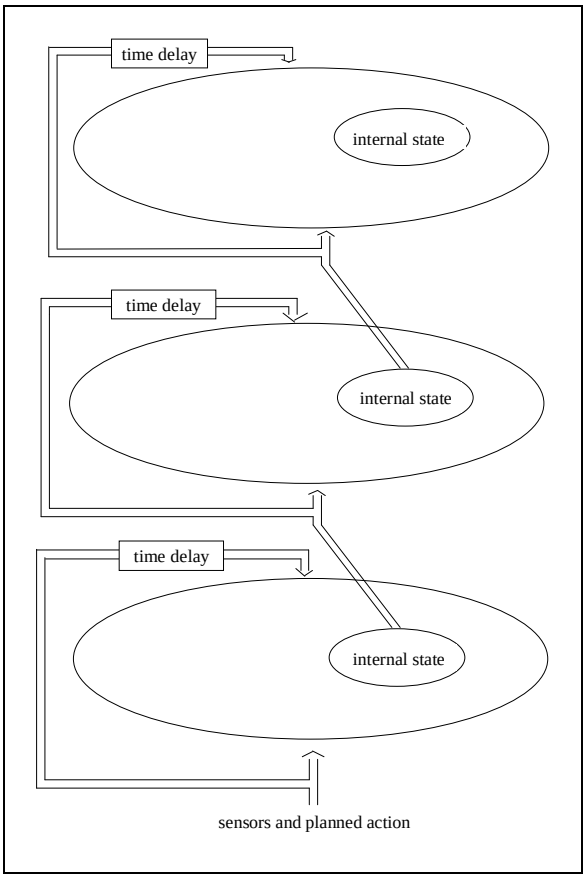

Figure 1. Hierarchy of prediction layers. The first level prediction layer (at the bottom) predicts the state of the sensors at time t_{+1} by receiving as input their state at time t and the planned motor action. The higher level prediction layers predict the internal state of the lower level layers at time t_{+1} by receiving as input their state at time t.

Moreover, in order to focus on relevant states and reduce the time lag between the events and the features that can be used to infer their occurrence we propose an architecture in which, at each level, a segmentation layer compresses sub-sequences of homologous states into a single state (Figure 2). These segmentation layers, by identifying sequences of homologous states at each level, allow the network to progressively reduce the frequency of states to be predicted going from lower to higher

[3] Consider, for example, the case of a state S_0 which is followed 50% of the times by a state S_1 and 50% of the times by the state S_2. Moreover imagine that the distribution of S_1 and S_2 is random (i.e. we cannot predict if S_0 will be followed by S_1 or S_2 even by taking into account the previous states). If S_1 and S_2 are orthogonal (i.e. if they are completely different) the next state is totally unpredictable. In this case S_1 and S_2 will not affect the internal representation of the network because the network will learn to produce a neutral state S_3 equally distant from S_1 and S_2. On the contrary, if S_1 and S_2 are similar, the next state can be predicted to a certain extent. In this case S_1 and S_2 will affect the internal representation because the network will learn to produce a state S_3 which will be similar to both S_1 and S_2.

prediction layers. Therefore, while low level prediction layers, by being asked to predict long sequences of states may only learn to detect short term regularities, higher level layers, by being asked to predict shorter sequences, may be able to detect long term regularities. The role of the segmentation layers is, as in Schmidhuber (1992a; 1992b), to compress information in time by focusing on relevant states. The difference is how relevant states are identified. In this case, relevant states are internal states, at a certain level, which differ significantly from the previous states. In Schmidhuber (1992a; 1992b) instead, relevant states are states which cannot be predicted from the lower level prediction layer.

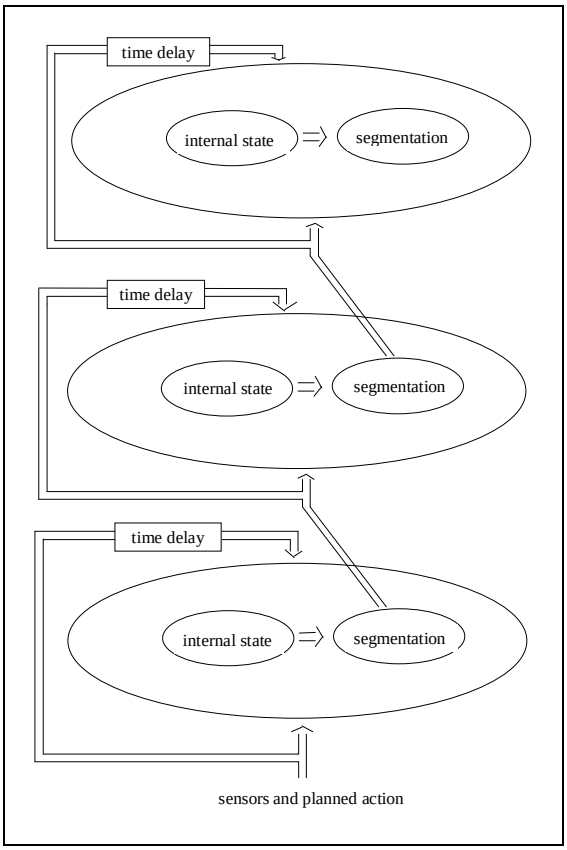

Figure 2. Hierarchy of prediction and segmentation layers. Each layer includes a prediction and a segmentation sub-network. The frequency of the spreading of activation is reduced going from lower to higher levels because high level layers predict only when the lower segmented state changes. So higher level layers should predict the internal state of the lower level layers at time t_{+n} where 'n' corresponds to how many steps the internal states of the lower level do not change significantly from the state at time t.

Unfortunately, the process of compressing information in time at each level may be strongly affected by the presence of unpredictable information. At each level, in fact, there is no way to discriminate between relevant states (i.e. states which are significantly different from the previous states or which are unpredictable from the lower level prediction layer) and unpredictable states. States which are totally unpredictable, in fact, will probably be different from the previous states and will be certainly unpredictable from the lower level prediction layer. As a consequence, the process of focusing on relevant states risks ending up in a process that focuses on

unpredictable states as well. This may have serious consequences on the higher level prediction layers.

For this reason it is important that, as we have mentioned above, each prediction layer progressively re-codes information while filtering out unpredictable information. It should be noted that at a given level we cannot discriminate between features that are completely unpredictable and features which are unpredictable at the current level but which may be predictable at higher levels. This implies that, by filtering out information which is unpredictable at the current level, we risk eliminating not only information which is really unpredictable but also information which could be predicted by the higher levels. However, as we will see, information which is predictable and which is stable throughout time does not risk being eliminated. Consider for example the occurrence of the feature *x* which occurs at time *t* and lasts *n* cycles. Moreover, imagine that the occurrence of *x* can be predicted only by taking into account the presence of a feature *y* which occurred several time steps before. The first level prediction network will probably fail to predict the occurrence of *x* at time *t* and as a consequence the internal state of this network will not be affected by *x*. However, at time t_{+1} the sensory state will include *x*. If *x* tends to last for a significant number of time steps after the first occurrence, the first level prediction network will probably predict correctly that *x* will occur also in the next sensory state. As a consequence the feature *x* will be filtered out at time *t* but not in the next time steps in which it will be both present in the input and in the output state.

1.3 Extracting regularities at different levels

This work has been inspired by a set of experiments described in Tani (1996). In these experiments a mobile robot explored the workspace by means of collision-free maneuvering. When the robot encountered a fork in the road, the robot decided either to take an alternative direction or to maintain the same direction. At these points (i.e. at the points of intersection) a recurrent neural network would attempt to predict the next sensory input which the robot will encounter at the next intersection having as input the current sensory state and the decision taken at the intersection. These intersections thus play the role of landmarks and involve a temporal segmentation of the sensory-motor flow. However, which are the landmarks and how the sensory motor flow should be segmented is the result of a decision of the experimenter (see also Mataric, 1992). In this paper we will try to eliminate these types of predefined mechanisms in the hope that the robot itself will be able to extract them by using the feedback of the environment.

Deciding in advance what will be landmarks and how the sensory-motor flow should be segmented has at least two drawbacks: (a) if the environment changes significantly the experimenter has to re-design, at least in part, the control system of the robot; (b) the difficulty in detecting the nearest landmark and in predicting the next depends on which the landmarks are selected by the experimenter. We should consider that landmarks which are easily perceived by a human observer may be hard to perceive by a mobile robot that has a sensory motor system which is very different from our own. Moreover, landmarks should be selected by taking into account several factors such as how different they are from the other selected landmarks and how stable they are, given the current motor behavior of the robot. What a good landmark is depends on several factors: the task of the robot, the environment, and the sensory-motor structure of the robot. From the point of view of the prediction task a good landmark is not only a set of

states which can be easily identified but also a set of states which are useful in predicting the next landmarks. For these reasons it is clear that it would be better to have the robot learning what landmarks are and how they are arranged in sequence at the same time.

In this paper we will use the hierarchical architecture described in Figure 2 both to extract a set of landmarks from the environment and to predict the next landmark that the robot will encounter. We will refer to extracted regularities (landmarks) as *segmented states* to indicate that: (a) they are not identified in advance; (b) they may correspond to sub-sequences of similar states at different levels (at the sensory-motor level, at the internal level of the first level prediction layer, or at the internal level of higher layers); (c) they do not necessarily correspond to environmental circumstances which can be described by using a single or few words of our own language (such as the name of an object or of a topological structure such as an intersection point). A segmented state, for example, may correspond to a similar sequence of sensory-motor states which the robot experiences while it is moving along a wall. Or it may correspond to the states that the robot experiences by following the walls of a square room which is followed by a narrow corridor. Or, it may correspond to the states which the robot experiences by encountering an object from a particular angle or in a particular context, for example after following a wall of a certain length.

The hierarchical architecture described in Figure 2 may extract regularities at different levels of organization. The first level prediction layer, by being exposed to low level information (sensory-motor states) and to long sequences of states can only extract low level regularities such as, for example, 'an obstacle is approaching from the frontal direction'. Higher level prediction layers however, by being exposed to higher level internal states and to shorter sequences may be able to extract higher level regularities such as, for example, 'I am leaving the big room'.

It should be noted that although in this paper we discuss the implications of this hierarchical architecture in the case of a mobile robot navigating in an structured environment, the architecture is quite general and can be applied to very different domains. For example, one can imagine applying the same architecture to extract regularity at different levels of organization (e.g. phonemes, syllables, words etc.) in a sound wave corresponding to a piece of speech. In other words the strategy of forcing the neural network to learn to re-code the sensory information by extracting regularities in space and time can be seen as a general strategy for solving different type-2 problems.

A final remark concerns the role of the motor actions produced by the robot. In this paper we will assume that the behavior of the robot is predetermined and fixed. On the other hand it is clear that the ability to predict the next sensory states might be strongly affected by the behavior of the robot. Indeed, by evolving plastic controllers which were allowed to modify their behavior, it was shown that evolved individuals displayed an initial behavior that enhanced their ability to learn to navigate in their environment (Nolfi & Parisi, 1997). For a first attempt to develop a control system which extracts regularities from the environment through prediction learning and uses this information to modify its own behavior see (Ito & Tani, 1998).

2. The experimental setup

In this section we will describe the experimental setup and the detailed architecture which we used in our experiments.

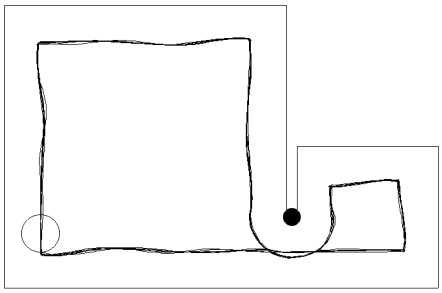

Figure 3. An environment made of two rooms joined by a short corridor. The straight lines represent the walls and the full circle represents a cylindrical object. The circle on the left-bottom side represents the robot and the trace on the terrain represents the trajectory of the robot during a few laps in the environment.

Let us consider the case of a mobile robot that navigates by performing a wall following behavior in an environment like that described in Figure 3. The robot (see Figure 4) is a miniature mobile robot developed at E.P.F.L. in Lausanne, Switzerland (Mondada, Franzi, & Ienne, 1993). It has a circular shape with a diameter of 55 mm, a height of 30 mm, and a weight of 70g. It is supported by two wheels and two small Teflon balls. The wheels are controlled by two DC motors with an incremental encoder (10 pulses per mm of advancement by the robot). The robot is provided with eight infra-red proximity sensors (six sensors are positioned on the front of the robot, and the remaining two on the back). The infrared sensors can detect obstacles within a range of about 3cm.

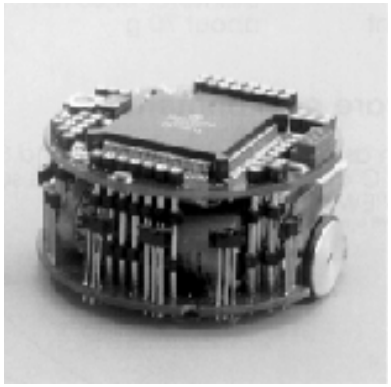

Figure 4. The Khepera robot.

The Khepera robot, which is programmed to produce a wall following behavior[4], is placed in an environment made of two rooms of 40x40 cm and 20x20 cm respectively joined by a corridor of 10 cm in width. The robot is asked to predict what the state of the 8 infrared sensors will be after each movement, given the current state of the sensors and the selected speed of the two motors. The state of the motors and sensors is updated each 100ms (step) and the robot is asked to predict the sensory pattern it will experience

[4] As we said above the behavior of the robot might strongly affect its ability to learn to predict the next sensory states. We choose this behavior because it is easy to implement and because it allows the robot to explore environments with different topologies. A similar approach has been used by Nehmzow (1992).

after each cycle (a cycle may correspond to one or more steps depending on the level of the layer involved in the prediction, see below).

In doing so we expect that our robot will extract a dynamical representation of the environment that will incorporate the topological structure of the environment and the relative position of the robot in the environment. This type of representation may be useful in achieving several different goals such as, for example, navigating to a desired part of the environment[5].

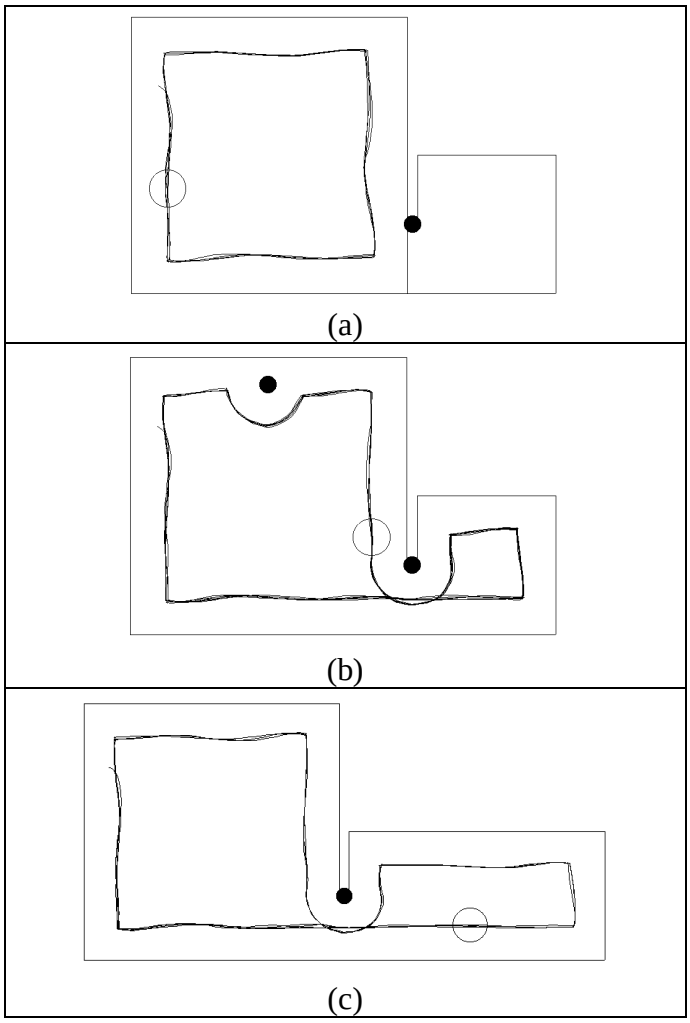

Figure 5. Three variations of the environment described in Figure 3.

In this paper we will first consider the simplest use of the representation extracted through prediction learning: detecting if something changed in the environment. Imagine a sort of vigilant-robot which would check and report on any significant change occurring in a delimited environment. We place in such an environment a robot which is programmed to explore the environment itself and we train the robot to predict its next

[5] The ability to directly navigate to a target area requires that the robot modifies its own motor behavior (which is something that will not be discussed in this paper). On the other hand, a similar result can be obtained by imagining that the robot stops to perform the wall-following behavior when he reaches the target area. As we will show in section 3.3, in fact, the robot can be trained to identify when it is traveling along a certain area of the environment.

sensory states. Hopefully, during the training, the prediction error will be progressively reduced up to a point in which it will remain relatively low. After that we can use the margin of error as an indication of the environmental stability. If the margin of error remains low, we may conclude that the environment is stable because the sequence of sensory-motor states experienced by the robot confirms the robot expectations which have been developed during the training phase. On the contrary, if the prediction error suddenly increases, we may conclude that something has been changed in the environment.

For instance, we want a robot which, after being trained in the environment described in Figure 3, is able to detect if the corridor between the two rooms has been closed, if a new obstacle has been placed in the environment, or if the extension of one of the two rooms has been altered (see Figure 5 a, b, and c, respectively).

The experiments which will be described have been conducted in simulation (for more details on the simulator see Miglino, Lund, & Nolfi, 1995). The simulator has been designed by taking samples of a real environment using the real robot sensors and motors[6]. Moreover, it has been shown that quite similar forms of behavior can be obtained by testing the same control system on the simulated and on the real robot (see Miglino, Lund, & Nolfi, 1995; Nolfi, 1997). Finally, it should be noted that noise is added to the sensors (sensors may have 1024 different states ranging from 0.0 to 1.0 and noise has been accomplished by adding a random generated value ranging from –0.05 to +0.05 to each sensor value). The presence of noise and the fact that the state of the sensors and motors are updated each 100ms implies that the trajectory of the robot in different laps in the environment may differ slightly (see Figure 3).

The neural architecture is organized as a cascade of prediction and segmentation layers arranged hierarchically. Although the number of layers can be arbitrarily chosen in the experiment described in this paper we will refer to an architecture with only two layers (see Figure 6).

The first level prediction layer is a feedforward network with 10 sensory units (which encode the state of the eight infrared sensors of the robot and the speed of the two motors), 3 hidden units, and 8 output units (which encode the state of the infrared sensors at time t+1). Moreover, the network is provided with 3 additional input units which encode the activation state of the hidden units at time t-1. This recurrence gives the network dynamical properties which make it possible for the network to process

[6] The walls and the cylindrical objects were sampled by placing the real robot in front of one of them, and by letting it turn 360^{o}, recording, at the same time, the state of the infra-red sensors at different distances with respect to the objects. The activation level of each of the eight infra-red sensors was recorded for 180 different orientations and for 20 different distances. In this way two different matrices of activation were obtained for the two types of objects (walls and cylinders). These matrices were then used by the simulator to set the activation state of the simulated sensors depending on the relative position of Khepera and of the objects in the simulated environment. The effects of the two motors were sampled similarly by measuring how Khepera moved and turned for each of the 20x20 possible states of the two motors. At the end of this process a matrix was obtained that was then used by the simulator to compute the displacements of the robot in the simulated environment. This sampling procedure may prove to be time consuming in the case of highly unstructured environments because it requires to sample each different type of objects present in the environment. However, it has the advantage of taking into account the fact that different sensors, even if identical from the electronic point of view, do respond differently. Sampling the environment throughout the real sensors of the robot allowed us, by taking into account the characteristics of each individual sensor, to develop a simulator shaped by the actual physical characteristics of the individual robot we have (for more about methodological issues see Nolfi, Floreano, Miglino, and Mondada, 1994; Miglino, Lund, and Nolfi, 1995).

sequential inputs (Elman, 1993; see also Robinson & Fallside, 1987; Williams, 1989; Williams & Zipser 1992).

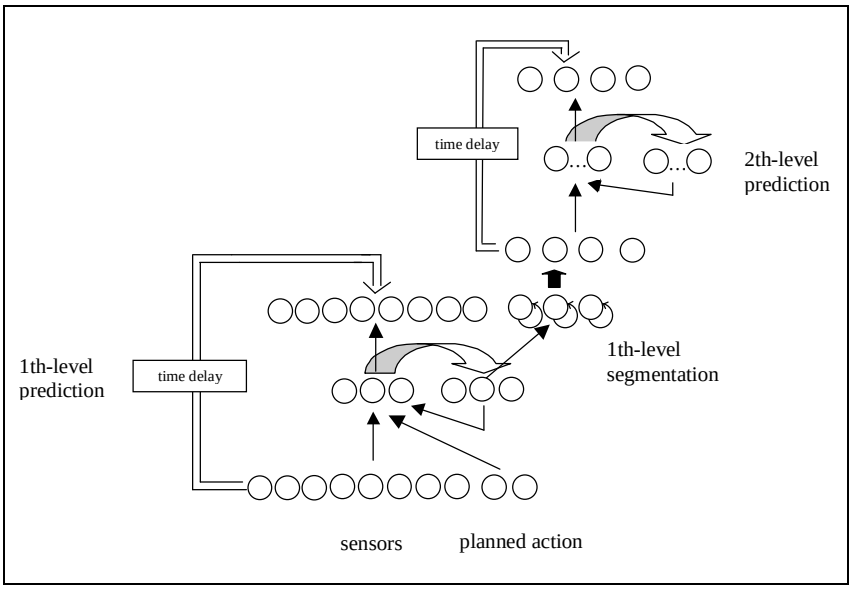

Figure 6. The architecture of the network. Single arrows indicate the weights of the first and of the second prediction layer which are taught by back-propagation and the weights of the segmentation layer which are taught through unsupervised learning. Empty double arrows indicate that the input state is used as teaching input at time t+1 both for the first and the second prediction layer (notice however that the first level prediction layer cycles each 100ms while the second level prediction layer cycles when a new segmentation occurs). The curved double arrows indicate that the state of the hidden units of the first and of the second prediction layer is copied after each cycle into a corresponding set of additional input units. Finally, the full double arrow indicates a 'winner take all' network.

The segmentation network is a fully connected two-layer network with 3 input units and 3 output units. The input layer is constituted by the hidden units of the first level prediction layer. The output layer is constituted by a set of output units with recurrent connections which locally encode the current segmented state. The activation state of each output unit is fed into a 'winner takes all' layer so that the output corresponding to the most activated unit is set to 1.0 and all other outputs are set to 0.0.

The second level prediction layer is a feedforward network with 4 sensory units (which are constituted by the 3 outputs of the segmentation network and one input encoding how many cycles the previous segmented state lasted), 30 hidden units, and 4 output units (which encode the next segmented state and after how many cycles the segmentation will shift from the current to a new one). Moreover, the network is provided with 30 additional input units which encode the activation state of the hidden units at time t_{-1}.

In the case of the first and second level prediction layers the units are activated by using the logistic function and the weights are updated by using standard back-propagation (Rumelhart, Hinton, & Williams, 1986). A learning rate of 0.2 and no momentum were used. Weights were randomly initialized within -0.1 and +0.1 and were updated after each cycle. In the case of the segmentation network, the activation state of a unit (i) is set to a value proportionate to the inverse of the Euclidean distance (normalized between 0.0 and 1.0) of the corresponding weights (W_{ij}) from the current

input pattern (X_j) and to the previous activation state (due to the recurrent connections). The constant μ was set to 0.25.

$$i = \left(\mu \cdot \left(1.0 - \frac{\sum_i (X_j - W_{ij})^2}{\sum_j \sum_i (X_j - W_{ij})^2} \right) \right) + \left(i_{t-1} \ (1.0 - \mu) \right)$$

Weights were randomly initialized within -0.1 and +0.1. Learning is accomplished by reducing the distance between the weights of the winning unit (i^*) and the current input pattern (X_j) with a decreasing learning rate (η is initially set to 0.3 and reduced by 50% each 1000 cycles). The winning unit (i^*) is the unit which has the minimum Euclidean distance from the current input pattern (X_j). This means that, unlike Self-Organizing Maps (Kohonen, 1982), only the weights of one unit are updated each cycle[7].

$$\text{if } i = i^* \quad \rightarrow \quad \Delta W_{ij} = \left(\eta \cdot \left(X_j - W_{ij} \right) \right)$$
$$\text{if } i \neq i^* \quad \rightarrow \quad \Delta W_{ij} = 0$$

$$i^* = \min \left(\sum_i (X_j - W_{ij})^2 \right)$$

Finally, it should be noted that while the first level prediction layer is updated each 100ms (i.e. each step), the second level prediction layer is updated only when the segmentation layer produces a new segmented state (i.e. when, after the processing of the winner take all weights, the current activated unit is different from that which was active the previous cycle). Therefore, if the segmentation state changes on an average of each 10 steps, for example, the second level prediction layer will be updated each second, on the average.

The architecture is trained in three phases. First the first level prediction layer is trained, then the segmentation network, and finally the second level prediction layer.

In a recent paper we have proposed a different but related architecture (Tani & Nolfi, 1998). In this work we used a mixture of recurrent neural networks (MRE) which is an extension of the architecture proposed by Jacobs & Jordan (1991). Each module competes to become an expert at predicting different sub-sequences of the sensory-motor flow and the shift between experts is used to produce segmentations. Also in this case we have an architecture which is hierarchically organized. However, the second layer was updated each 10 steps (i.e. at a fixed rate) while in the case of the experiments described in this paper the update rate of the second level prediction layer is not fixed and is determined by the lower level segmentation network.

3. Experimental results

In this section we will report the results of the training of the first-level prediction layer, the segmentation layer, and the second level prediction layer. Moreover, we will

[7] Given that only the weights of the winning unit are updated it may happen that some units never become the winning unit and therefore are never updated. This problem however does not tend to arise when, as in the case of the experiment describe in this paper, the input patterns are affected by noise (see Szu, 1986).

provide an analysis which shows what type of regularities have been extracted through the interaction with the environment.

3.1 First-level prediction

We ran 10 experiments starting with different randomly assigned weights. Each network was trained for 100,000 steps (i.e. 100,000 cycles given that the first level prediction layer is update each step). As can be seen in Figure 7, in all cases the summed square error decreases until it reaches a stable value around 0.05. This means that the network produces almost perfect performance (i.e. the network output closely maps the state of the sensor at time t_{+1}).

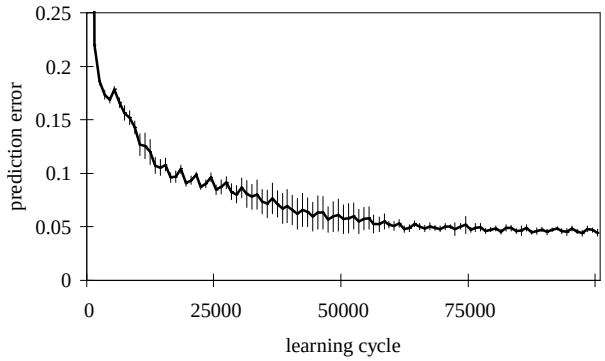

Figure 7. Sum of squared error over the 8 output units of the first level prediction layer. The average error and the standard deviation over 10 runs are shown. Before learning, the average error is around 1.6 (data not shown in the graph); however, during the first 1000 steps the error suddenly decreases to about 0.2.

However, if we test the trained network in the three variations of the environment shown in Figure 5 we can see how the prediction error remains quite low (see Figure 8). This implies that in order to predict most of the next sensory states the network does not need to extract knowledge about the topological structure of the environment. This can be explained by considering that the state of the sensors does not change dramatically during 100ms. Moreover, in most of the cases the current state of the sensors and the speed of the motors can be used to predict the next sensory state to a good extent without taking into account the previous states (indeed, by training a first level prediction network without memory units, we obtained almost identical performance)[8]. This implies that the network can produce outputs very close to the desired teaching inputs without taking into account high level features of the environment such as the length of the walls. Sometimes the error increases significantly (for example when the robot approaches a corner and as a consequence the frontal sensors start to be activated for the first time). However, such situations are infrequent. As a consequence, the network converges on a solution which does not take these cases into account[9].

[8] If we just copy the state of the first 8 input units which encode the current state of the infrared sensors into the output units we obtain an average error of 0.08 each 1000 cycles. The fact that the network produce a lower error imply that it is computing something more interesting. The network, in fact, is able to correctly predict if sensors will increase or decrease their activation level during 100ms in most of the cases.

[9] This also explains the variation of the prediction error in the three testing environments. For example, the fact that corners are encountered less frequently in environment (a) than in the other environments may explain why the prediction error is lower, on the average, in this environment.

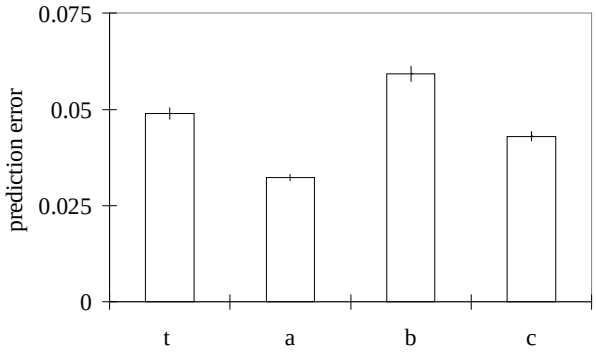

Figure 8. Prediction error for the first-level prediction layer in the case of the training environment (t) and in the case of three variations of the training environment described in Figure 5. The graph shows the average error and the standard deviation obtained by testing 10 different networks throughout 5000 steps. Networks have been tested after training without updating the weights.

3.2 First-level segmentation

After training the first level prediction layer we trained the segmentation layer which takes as input the state of the hidden units of the lower layer and should classify such patterns into three different classes on the basis of their similarities. The segmentation layer has been trained for 100,000 steps (i.e. 100,000 cycles). The training process has been replicated for each of the 10 networks starting with different randomly assigned weights. Figure 9 shows how the sensory-motor flow experienced by the robot is categorized in the case of the training environment. As can be seen, after the training process, the internal states of the first level prediction layer are classified into three classes: (1) patterns experienced while the robot is doing wall following (full circle); (2) patterns experienced while the robot is turning along a corner (full square); (3) patterns experienced while the robot is traveling along the corridor (empty circle). The same type of segmentation can be observed in the 10 replications of the experiment. In fact in all cases, the input patterns during one lap in the environment are segmented into 16 different sequences corresponding to the three classes described above.

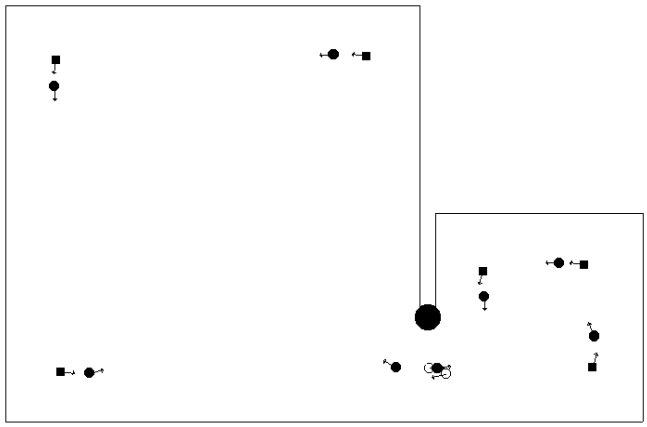

Figure 9. Segmentations performed during one lap in the environment. Full circle, empty circle, and full square indicate, respectively, which of the three segmentation units is active in a given cycle. The state of the segmentation units is displayed only when a new segmentation occurs (i.e. when the activation state of output units of the winner takes all layer changes with respect to the previous cycle). The arrows

indicate the direction of the robot when the corresponding segmentation occurs. Results for a typical network.

Of course, if we change the number of output units in the segmentation layer, partially different segmentations are obtained. The number of output units, in fact, determines the maximum number of classes in which the sensory-motor flow can be divided. By replicating the experiment with two output units we observed that in 8 cases the segmentation network converged in a solution in which one class corresponds to the patterns experienced while the robot was traveling along a corridor and the other class to both the patterns experienced while the robot was traveling along walls and corners. In the remaining two cases, the network converged in a solution in which one class corresponds to the patterns experienced while the robot was traveling along a corner and the other class to both the patterns experienced while the robot was traveling along walls and corridors. By replicating the experiment with four output units we observed that in all cases the segmentation layer converged in a solution in which sensory patterns were classified into patterns corresponding to: wall following, traveling along the corridor, negotiating the first part of a corner, and negotiating the second part of a corner. It should be noted, however, that by increasing the number of segmentation outputs the distance between different classes of patterns decreases. As a consequence, segmentations become progressively less stable throughout time (i.e. for example, in the case of the experiments with four segmentation units, patterns experienced by negotiating corners are usually segmented into two classes. Sometimes however, because of noise and because the trajectory of the robot may slightly differ in different laps in the environment or during the negotiation of different corners, they are segmented into only one class).

Notice that the segmentation layer takes as input the internal state of the first level prediction layer (i.e. a representation which may significantly differ from the sensory-motor flow). In particular, we expect that the first level prediction layer, by trying to predict the next sensory state, will produce a similar internal representation for sensory patterns which are different from the sensory point of view but are functionally equivalent (i.e. are followed by similar sensory patterns). On the contrary, sensory patterns which are similar but tend to be followed by different sensory-motor states will be represented by different internal states. Moreover, we may expect that the internal representation of the first level prediction layer will be organized so as to contain those characteristics of the sensory patterns which are useful in prediction and not necessarily other characteristics which are not useful in prediction. In other words, we may expect that the first level prediction network will partially filter out information that is unpredictable at the first level.

That the pre-processing of the sensory-motor flow is accomplished by the first level prediction layer affects the type of segmentations produced by the segmentation layer can be shown by replicating the experiment using the sensory-motor patterns (instead of the internal representation of the first level prediction network) as input for the segmentation network[10]. In doing so we observed that in 9 cases out of 10, despite the

[10] In this case we used an architecture without the first level prediction layer and with a segmentation network with 10 input units (which encode the state of the 8 infrared sensors and the speed of the two motors) and 3 output units. It should be noted that in the standard experiment the segmentation network receives input patterns that may also be influenced by the previous sensory states while in the replication of the experiment the input patterns consist in the current sensory-motor states only. However, as we said

segmentation networks had 3 output units, sensory-motor patterns were classified in only two classes: patterns experienced along a corner and patterns experienced during wall following. This can be explained by considering that these two classes of patterns are already significantly different at the sensory level given that they correspond to partially different state of the sensors and quite different state of the motors (the speed of the left motor is all in the back during corner negotiation and all in the front during wall following). On the contrary, the patterns experienced traveling along the corridor are not very different from the other patterns at the sensory-motor level (the only difference is that the infrared sensor positioned on the left side of the robot is activated while traveling along the corridor). The difference between the patterns experienced while traveling along walls and along the corridor is enhanced in the internal representation of the first level prediction layer because they are useful to predict the next sensory states. The left side infrared sensor, in fact, is only activated along the corridor; therefore its state is quite reliable and easy to predict after the sensor starts to be activated the first time.

3.3 Second level prediction

After training the first level prediction and segmentation layers we trained the second level prediction layer which takes as input the current segmented state (locally encoded) and the time length of the previous segmented state and should produce as output the next segmented state and the time length of the current segmented state (i.e. after how much time the new segmentation will occur). This layer has been trained for 10,000,000 steps. However, given that this layer is updated only when a new segmentation occurs (each 18.7 steps, on the average, given the type of segmentations produced by the segmentation layer described in the previous section), training lasted about 530,000 cycles. As can be seen in Figure 10, the summed square error decreases until it reaches a value of around 0.04 on the average (the error reaches 0.012 in 8 replications and 0.13 in the other 2 replications). This means that networks produce almost perfect predictions in 8 cases out of 10.

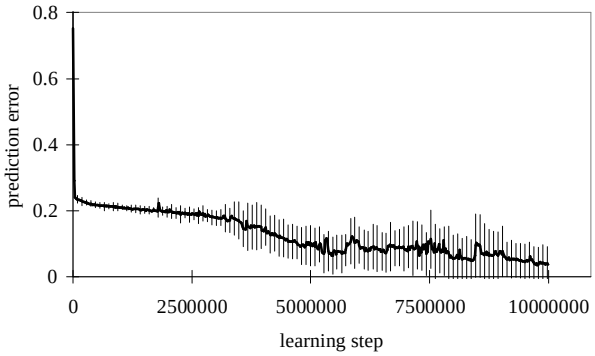

Figure 10. Sum of squared error over the 4 output units of the second level prediction layer. The average error and the standard deviation over 10 runs are shown.

If we test the trained network in the three variations of the environment shown in Figure 5 we can see how the second level prediction error significantly increases when

in section 3.1, similar performance can be obtained by using a first level prediction network without memory units.

the robot experiences different environmental conditions (see Figure 11). This implies that the second level prediction layer extracted internal representations which include the topological structure of the environment. These representations allow the robot to make the correct predictions in the training environment and, conversely, to produce incorrect predictions when the environment changes.

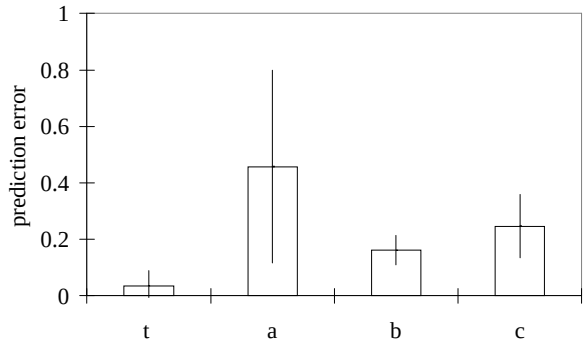

Figure 11. Prediction error for the second level prediction layer in the case of the training environment (t) and in the case of three variations of the training environment described in Figure 5. The graph shows the average and the standard deviation of the error obtained by testing 10 different networks throughout 100,000 steps. Networks have been tested after training without updating the weights.

The knowledge extracted through prediction learning allows the robot not only to predict the next states but also to localize its own position in the environment. In Figure 12 we can see the prediction error of the second level prediction layer in a test made by moving the robot from one to another different position each 1000 steps. As can be expected, after each replacement, there is a sudden increase in the prediction error because the robot is disoriented. It is expecting to perceive a sequence of sensory-motor state corresponding to the part of the environment where it was located while it is perceiving a sequence corresponding to another portion of the environment where it was replaced. However, after relatively few cycles the prediction error decreases again (the robot is able to recover after one lap in the environment, on the average). This shows how the robot has developed a dynamical representation which includes the relative position of the robot itself in the environment. Moreover, this shows how such a representation is robust enough to allow the robot itself to easily recover the right representation of its relative position after a replacement.

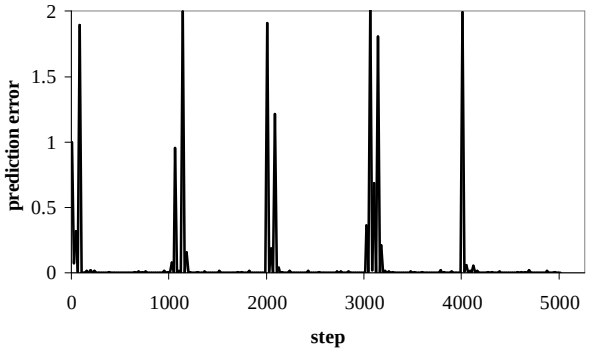

Figure 12. Prediction error for the second level prediction layer in the case of a test in which the robot is replaced each 1000 steps. For each replacement, the robot is moved to the position in which it was 150

steps before (given that the robots takes about 320 steps to perform a complete lap in the environment this means that most of the times it is replaced from one to the other room). Result in the case of the best replication (the replication in which the second level prediction layer reaches the lowest error). The network has been tested after training without updating the weights.

That the robot extracts a representation of the topological structure of the environment can be further shown by replicating the experiment with a second level prediction layer which should not only predict the next segmented state but also should produce as output one or more labels which indicate where the robot is currently located. This can be accomplished, for example, by adding to the second level prediction layer two output units which should be turned on when the robot is traveling along the left side wall of the large room and the right side wall of the small room. The teaching input for these two additional units will be provided by the experimenter and not from the environment as in the case of the other units trained by back-propagation. In particular, we provide as teaching input the patterns [1 0] when the robot is at a distance lower than 75mm from the left side wall of the large room, [0 1] when the robot is at a distance lower than 75mm from the right side wall of the small room, and [0 0] in all other cases.

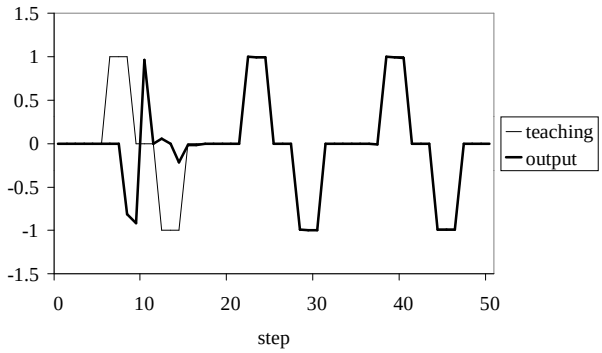

Figure 13. Outputs and teaching inputs of the two additional units which locally encode: (1) when the robot is located close to the left side wall of the large room; (-1) when the robot is located close to the right side wall of the small room; (0) when the robot is located elsewhere. The graph shows the difference between the activation of the two units (i.e. 1.0 represents the vector [1.0 0.0], 0.0 represents the pattern [0.0 0.0], and -1.0 the vector [0.0 1.0]). Data obtained by testing one typical individual for 911 steps corresponding to 50 cycles. The robot is initially placed in a randomly selected position and it is tested without updating the weights.

In doing so, we can see that after a few cycles the robot, in addition to correctly predicting, it is able to correctly label when it is traveling along the two chosen locations (see Figure 13). Given that the robot starts from a random selected position, it takes some time to localize itself in the environment (about 200 steps in this case). In this phase both the predictions and the labeling produced by the second level prediction layer are incorrect (see Figure 13). After such a phase, however, the robot starts to produce correct predictions and to label the two rooms correctly. In other words the robot is able to indicate when it is located in a certain selected location of the environment. Notice that if we add the two additional output units to the first level prediction layer instead of to the second level the network is unable to learn to provide the correct labeling even if we increase the number of internal units (result not shown).

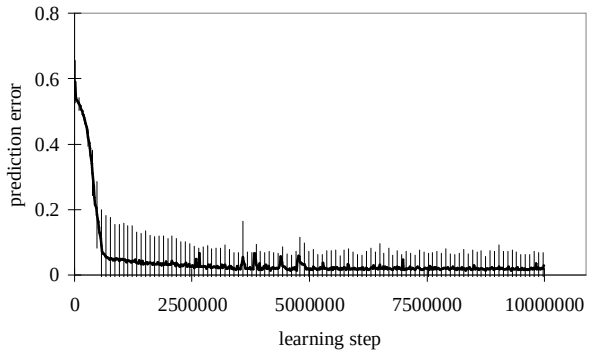

Figure 14. Sum of squared error over the 6 output units of the second level prediction layer (the two additional units encode the current location of the robot). The average error and the standard deviation over 10 runs are shown. The error is initially higher than 0.8. However, we keep the same scale of Figure 10 to allow an easier comparison of the two graphs.

Interestingly, in this last experiment in which we added the two additional output units and provided as teaching input the correct label, the second level prediction layer converges much faster (see Figure 14) than in the case in which the layer is only asked to predict the next segmentation (Figure 10). This can be explained by considering that the addition of the labeling units, can be seen as a way of channeling the learning process in the right direction[11]. In fact, it is reasonable to think that, to produce the correct predictions, the robot should be able to 'know', among other things, when it is located in the two chosen locations. By asking the network to label these locations and by providing the right teaching input we force the robot to develop an internal representation which includes such information. This internal representation is a first step toward the development of a richer internal representation (which also includes information concerning other locations) that is needed to correctly predict the next segmented states.

4. Discussion

We proposed that the ability to extract regularities from time series through prediction learning can be enhanced if we use a hierarchical architecture in which higher layers are trained to predict the internal state of lower layers when such states change significantly. This hierarchical organization has two main functions: (a) it forces the system to progressively re-code sensory information so as to enhance useful regularities and filter out useless information; (b) it progressively reduces the length of the sequences which should be predicted going from lower to higher layers. As a consequence, while lower layers extract low level regularities, higher layers can extract higher level regularities which are hidden at the sensory level.

We investigated this issue in the context of a mobile robot which navigates in a structured environment. In this context, as in many other realistic cases, the problem is particularly hard because sensory information comes at high frequency and regularities

[11] A similar technique has been used by Dorigo and Colombetti (1994) with the goal of facilitating the learning of complex tasks and has been named *behavior shaping*. The term shaping indicates that the knowledge of the experimenter is used to channel the learning process in the right direction.

at different levels of organization should be taken into account to predict the next sensory states.

By training a neural network organized in two layers to predict the next sensory state in a robot navigating in a two room environment we showed how the first level prediction layer was able to extract low level regularities such as 'walls', 'corners', and 'corridors' while the second level prediction layer was able to extract higher level regularities such as 'the left side wall of the large room'. The ability to extract regularities at different levels allows the robot to detect if something changes in the environment (i.e. the fact that a door has been closed, the presence of a new obstacle, or an alteration of the shape of a room) or to learn to label a particular location of the environment such as the left side wall of the large room.

We showed how these tasks, by requiring higher level regularities, can only be accomplished by the second level prediction layer which has available high level regularities. On the other hand we showed how the regularities extracted by the second level prediction layer depends on the re-coding of the sensory-motor information performed by the first level prediction layer. This first level layer, in fact, transforms the representation at the sensory-motor level by decreasing the difference between functionally similar patterns and increasing the difference between functionally different patterns.

Finally, we showed how the internal representations obtained through prediction learning do not only include information about the topological structure of the environment in which the robot has been trained but also an indication of the relative position of the robot in the environment. The fact that these two types of information are obtained by the dynamical interaction of the robot with the environment and are represented over the same pattern of units allows the robot to easily recover its ability to localize itself after being replaced in a completely different location of the environment.

An aspect of our model which we want to stress is the fact that it involves a developmental process (the network is trained in three successive phases). The fact that the training process of different layers is conducted in successive phases going from lower to higher levels is important because higher layers operate on the results of lower levels. In other words, what is learned by the first layers constrains what can be learned by the higher layers[12]. From this point of view the architecture proposed in this paper has some similarities with other models which involve a developmental process. Shrager and Johnson (in press) and Elman et al (1996) investigated the effects of a developmental wave of plasticity in which different regions of a neural network were plastic in different points in time (weights were changed according to a Hebbian learning rule). By comparing the results obtained by training the network with and without the wave of plasticity they found that in the first case some neurons learned to perform complex functions such as the exclusive OR (XOR). These neurons tended to be on regions which were late in reaching maturity. On the contrary, in the case in which all regions of the network were trained from the beginning, neurons only learned to compute simple functions such as the logical AND and OR. The reason why the

[12] In principle, the second level prediction layer will receive in input the same information after the first phase whether it is trained from the beginning or is trained after the first phase. However this does not imply that the second level layer will converge in the same solution in the two cases. The plasticity of a network trained by back-propagation, in fact, progressively decreases during the training. As a consequence, if the second prediction layer is trained by the beginning, it may fail to re-adapt to the changes that occur in the first level prediction layer.

developmental process allowed some neurons to compute more complex functions is that they did not learn until later, after early neurons had learned simpler functions. These early learning neurons, in fact, became additional inputs to the units which learned later. Since the XOR function can be decomposed into the AND and OR functions, this made it possible to learn a function which could not otherwise have been learned. In another study on grammar acquisition which we already mentioned in the introduction, Elman (1993) trained a simple recurrent neural network with a corpus of sentences of a pseudo-natural language. Words were presented one at a time and the network was asked to predict the next word which would occur. The author showed how the network was unable to solve the task through ordinary backpropagation learning. However, success could be achieved in either of two ways: (a) by dividing the training data into graded batches beginning with simple sentences and progressing to more complex ones; (b) by starting the learning process with a limited memory window (i.e. the activation state of the recurrent neurons was reset every third or fourth word) and then increasing it as training progressed.

Both these two examples and our hierarchical architecture show how some complex functions that are not normally learned in a static mature system in which learning occurs everywhere simultaneously can be learned in systems in which different regions are plastic in different periods of time. As claimed by Elman et al. (1996) and Clark & Thornton (1997) this can be explained by considering that the re-coding of the sensory representation achieved in the first learning phase reduces the complexity of the subsequent search (see also Schmidhuber, 1992a, 1992b). In our experiment and in the experiment with the wave of plasticity described in Elman et al. (1996) the representation acquired by the first layers of the network which initially undergoes the learning process constitutes the input for the successive layers which are subjected to the learning process later on. Similarly, in Elman (1993) the internal representation which is initially learned on the basis of a short time window and which is copied back into additional input units (memory units), constitutes a re-coded input for the successive learning in which the time window is increased.

The difference, in the case of the model presented in this paper with respect to the developmental models described above, is that representations are not only re-coded in space but also in time (i.e. sequence of functionally similar patterns are transformed into a single pattern going from lower to higher layers). In the architecture proposed in this paper, the first and second prediction layers operate a transformation in space while the first level segmentation layer operates a transformation in time. In particular, a set of sensory patterns is transformed into a single pattern on the basis of their similarity at a given level. This is only one possible way to accomplish a transformation in time and may not be the best one. In particular, this type of transformation in time implies a classification process in which each pattern should be necessarily grouped into one or another category. This way to proceed has advantages and drawbacks. On one hand, the change from one to another category can be used to easily compress patterns in time (all patterns corresponding to a single category are treated as a single pattern at higher levels). On the other hand, it is clear that patterns that lie between two categories tend to be randomly categorized and that information concerning differences between a single category (i.e. the differences between the sensory states which are grouped together by the segmentation layers) is lost. It would be interesting to find another way to transform information in time which does not necessarily imply a categorization process and at the same time allows a compression of the information in time.

Future directions of research may include, in addition to the study of different ways to re-code information in time, the introduction of feedback signals. The predictions elaborated by the higher levels of the architecture, in fact, may be usefully put back in lower predicting layers. This may be accomplished introducing additional input units at each level which encode the state of the hidden units of higher levels or by asking lower levels to predict also the hidden states of higher levels (see Schmidhuber, 1992a). In this case, the prediction at each level except for the last would be performed both on the basis of sensory data and of top down information (expectations). In this case, the developmental process cannot be divided into three separate phases. Changes from one to the next phase should be less abrupt so as to allow lower levels to learn how to use the information coming from higher levels. Finally, another important direction to explore is allowing the system to decide how to behave. This is both in order to use the extracted information in order to perform some other task and to enhance the system's ability to predict (for an example of how a mobile robot may shape its behavior in order to learn to perform a task see Nolfi & Parisi, 1997).

Acknowledgment
Stefano Nolfi thanks SONY for generously supporting his stay at SONY-CSL which provided an excellent environment for conducting this research. The authors thank the anonymous referees for their valuable suggestions.

References

Clark, A. & Thornton, C. (1997). Trading spaces. *Behavioral and Brain Sciences*, 20, 57-90.

Dorigo, M. & Colombetti, M. (1994). Robot shaping: developing autonomous agents through learning, *Artificial Intelligence*, 71, 321-370.

Elman, J.L. (1990). Finding structure in time. *Cognitive Science*, 14:179-211.

Elman, J.L. (1993). Learning and development in neural networks: The importance of starting small. Cognition, 48: 71-99.

Elman, J.L., Bates, E.A., Johnson, M.H., Karmiloff-Smith, A., Parisi, D. & Plunket, K. (1996). *Rethinking innateness*. *A Connectionist perspective on development*. Cambridge, Mass: MIT Press.

Floreano, D. & Mondada, F. (1996). Evolution of Homing Navigation in a Real Mobile Robot. *IEEE Transactions on Systems, Man, and Cybernetics--Part B: Cybernetics*, 26(3), 396-407.

Ito, M. & Tani J. (1998). Dynamical adaptation of a neural-net based agent. In L. Niklasson, M. Bodén, & T. Ziemke (Eds.), *ICANN 98 - Proceedings of the 8th International Conference on Artificial Neural Networks*. London, UK: Springer Verlag.

Jacobs, R.A. & Jordan, M.I. (1991). Adaptive mixtures of local experts. *Neural Computation*, (3) 1:79-87.

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

Mataric, M. (1992). Integration of representation into goal-driven behavior-based robot. *IEEE Transaction on Robotics and Automation*, 8(3):304-212.

Miglino, O., Lund, H. H. & Nolfi, S. (1995). Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life*, (2), 4, pp.417-434, MIT Press.

Mondada, F., Franzi, E. & Ienne, P. (1993). Mobile Robot miniaturisation: A tool for investigation in control algorithms, In: *Proceedings of the Third International Symposium on Experimental Robotics*, Kyoto, Japan.

Nehmzow, U. (1992). Experiments in Competence Acquisition for Autonomous Mobile Robots, *Ph.D. Thesis*, University of Edinburgh, U.K.

Nolfi S. (1997). Evolving non-trivial behaviors on real robots: a garbage collecting robot, *Robotics and Autonomous Systems*, 22, 187-198

Nolfi, S., Florano D., Miglino, O. & Mondada, F. (1994). How to evolve autonomous robots: different approaches in evolutionary robotics. R.A. Brooks and P. Maes (Eds.), *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, Mass: MIT Press

Nolfi, S. & Parisi, D. (1997). Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 1, 99-105

Parisi, D. & Cecconi, F. (1995). Learning in the active mode. In F. Moran, A. Moreno, J.J. Merelo & P.Chacon (Eds.), *Advances in artificial life. Proceedings of the third European Conference on Artificial Life*. Berlin: Springer-Verlag, 439-462.

Robinson, A.J. & Fallside, F. (1987). The utility driven dynamic error propagation network. *Technical Report* CUED/F-INFENG/TR.1, Cambridge, University Engineering Department.

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

Schmidhuber, J. (1991). Adaptive decomposition of time. In Kohonen, Maekisara, Simula, Kangas (Eds.), *Artificial Neural Networks*, North-Holland: Elsevier.

Schmidhuber, J. (1992a). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4 (2):234-242.

Schmidhuber, J. (1992b). Learning unambiguous reduced sequence descriptions. NIPS 4, San Mateo, CA: Morgan Kaufmann

Schmidhuber, J. & Prelinger, D. (1993). Discovering predictable classifications. *Neural Computation*, 5(4):625-635.

Shrager, J. & Johnson, M.H. (in press). Modelling the development of cortical function, In B. Julesz & I. Kovacs (Eds.), *Maturational windows and cortical plasticity in human development: Is there a reason for an optimistic view?* Reading, MA: Addison Wesley.

Szu H. (1986). Fast simulated annealing, In J. S. Denker (Ed.) *Neural Networks for Computing*, New York: American Institute of Physics.

Tani, J. (1996). Model-based Learning for Mobile Robot Navigation from the Dynamical Systems Perspective, *IEEE Transactions on System, Man and Cybernetics Part B (Special Issue on Robot Learning)*, (26) 3: 421-436.

Tani, J. & Nolfi, S. (1998). Learn to perceive world as articulated: An approach for Hierarchical learning. In R. Pfeifer, B. Blumberg, J-A. Meyer, S.W. Wilson (Eds.), *From Animals to Animats VI*, Cambridge, MA: MIT Press.

Williams, R.J. (1989). Complexity of exact gradient computation algorithms for recurrent neural networks. Technical Report Technical Report NU-CCS-89-27, Boston: Northeastern University, College of Computer Science.

Williams, R.J. & Zipser, D. (1992). Gradient-based learning algorithms for recurrent networks and their computational complexity. In Back-propagation: Theory, Architectures and Applications. Hillsdale, NJ: Erlbaum.