Neural computing increases robot adaptivity

Carme Torras
Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Parc Tecnològic de Barcelona - Edifici U
Llorens i Artigas 4-6, 08028-Barcelona. Spain.
ctorras@iri.upc.es, http://www-iri.upc.es/people/torras

Abstract.
The limited adaptivity of current robots is preventing their widespread application. Since the biological world offers a full range of adaptive mechanisms working at different scales, researchers have turned to it for inspiration. Among the several disciplines trying to reproduce these mechanisms artificially, this paper concentrates on the field of Neural Networks and its contributions to attain sensorimotor adaptivity in robots. Essentially this type of adaptivity requires tuning nonlinear mappings on the basis of input-output information. After briefly reviewing the fundamentals of neural computing, the paper describes several experimental robotic systems relying on the following adaptive mappings: inverse kinematics, inverse dynamics, visuomotor and force-control mappings. Finally, the main trends in the evolution of neural computing are highlighted, followed by some remarks drawn from the surveyed robotic applications.

Keywords: Adaptivity, sensorimotor mappings, neural networks, robot control.

1. Introduction

Why are robots still confined to factory floors and research departments? Will they ever step out and be part of our everyday lives? Aside from ethical considerations and marketing strategies, there are technological reasons that explain why the use of robots is not as widespread as some envisaged they would be by now. At the risk of oversimplification, let me state that the Achilles heel of current robots is their lack of adaptivity, at all levels. This capability is dispensable in well-engineered environments, and thus we have very performant robots in manufacturing lines, but it is a *sine qua non* when tasks are to be carried out in non-predefined worlds.

In this sense, the biological world –where adaptivity is crucial for survival– constitutes a very good source of inspiration for robotics researchers, since it provides existence proofs of many adaptive mechanisms that do function. However, caution must be taken, because the best natural solution may not be the best artificial one (Simon, 1969). Wheels, wings and calculators have often been mentioned as examples of artificial solutions considerably different from their natural counterparts, and more performant according to certain criteria. The resources

available to engineering design depart a lot from those in nature, and not just when it comes to materials, but also in the number of instances and spendable time.

With this note of caution in mind, i.e., accepting that biological plausibility in itself adds no special value from an engineering viewpoint, it is safe to look into natural adaptivity to get seed ideas that can be instantiated in a different way by artificial means.

2. Natural and artificial adaptivity

What exactly do we mean by adaptivity? What does it encompass? What is its range? By adaptivity we mean the capability of self-modification that some agents have, which allows them to maintain a level of performance when facing environmental changes, or to improve it when confronted repeatedly with the same situation. The term 'agent' above stands for a single cell, an organ, an individual or even a whole society, because, in the biological world, adaptivity occurs at several levels, each having a possible counterpart in the design of autonomous robots (Steels, 1995; Higuchi et al., 1997; Ziemke and Sharkey, 1998).

At the cell level, several chemical and electrical mechanisms of **plasticity** have been discovered, some of which have been modelled and analysed within the Neural Modelling field, and later applied to adjust the parameters of robot sensors and actuators. See the chapters on 'neural plasticity' in (Arbib, 1995).

When referring to individuals, adaptation is usually called **learning** and it takes two rather different forms depending on whether it occurs at the sensorimotor or cognitive levels. Sensorimotor adaptation consists in building relevant associations between stimuli and responses, while cognitive learning entails constructing symbolic representations to guide decision-making. Two disciplines have tried to mimick these two capacities. Neural Networks, closer to Biology, has proven adequate to handle the massively-parallel tasks of perception and control of action, while Artificial Intelligence, steeming from Computer Science and Cognitive Psychology, has developed the necessary data structures and procedures to tackle symbolic learning (Torras, 1993; Torras, 2001). Results in both disciplines have been applied to Robotics, the former to attain adaptive robot sensorimotor mappings (Omidvar and van der Smagt, 1997), while the latter have led to so called learning robots (van de Velde, 1993; Dorigo, 1996; Mitchell et al., 1996; Morik et al., 1999).

Finally, at the species level, adaptation is attained through **evolution**. Genetic algorithms (Goldberg, 1989; Koza, 1992) and evolutionary computation (Bäck et al., 1997; Beyer and Schwefel, 2002)

Table I. Levels of adaptation and related disciplines

ADAPTATION LEVEL	TYPE OF ADAPTIVITY	"ARTIFICIAL" DISCIPLINE
Cell	Plasticity	Neural Modelling
Sensorimotor	Associative learning	Neural Networks
Cognitive	Symbolic learning	Artificial Intelligence
Species	Evolution	Evolutionary Algorithms

are starting to be used to tailor robot genotypes to given tasks and environments (Husbands and Meyer, 1998).

Table 1 summarizes the different adaptation levels and the involved disciplines.

In this paper we concentrate on adaptivity at the individual sensorimotor level, i.e. both the robot morphology and its components are assumed to be fixed and what may change with experience is the functional relationship between sensors and actuators.

3. Natural inspiration for artificial neural adaptivity

The approaches to adaptivity pursued within the Neural Networks field have their roots in the learning paradigms developed in the domain of Behavioural Psychology (refer to Figure 1). This is the reason why the rules to attain neural adaptivity are usually called learning rules. The role of the animal in the behavioural learning experiments is played here by the neuron. It is worth noting that, although inspiration comes from the biological world, the artificial techniques are here applied not only to a different physical substrate, but also at a different level (neuron instead of animal).

The most basic learning paradigm is **classical conditioning**, as introduced by Pavlov (1927), which consists of repeatedly presenting to an animal (e.g. a dog) an initially meaningless stimulus (e.g. the sound of a bell) together with an unconditioned stimulus (e.g. food) that triggers a reflex response (e.g. salivation). As a result of such paired presentations, the animal builds up an association so that, when presented with the conditioned stimulus (e.g. the sound of a bell) alone, it produces the same response as before (e.g. salivation). This type of learning is completely open-loop, in the sense that it entails no feedback. The neural learning rules mimicking this type of conditioning are called *correlational rules*.

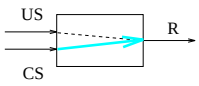
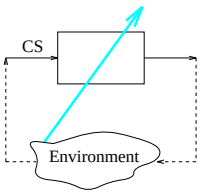
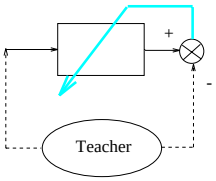
TYPE OF LEARNING	DEGREE OF FEEDBACK	DIAGRAM	KEYWORDS	NEURAL RULES
CLASSICAL CONDITIONING	none		Pavlov Unsupervised Open-loop	**CORRELATIONAL**
INSTRUMENTAL CONDITIONING	qualitative		Skinner Trial-and-error Optimal control	**REINFORCEMENT**
INPUT-OUTPUT TEACHING	quantitative		Supervised Reference-model control Closed-loop	**ERROR-MINIMIZATION**

Figure 1. Learning paradigms inspiring neural adaptivity. In the diagrams, US stands for unconditioned stimulus, CS for conditioned stimulus and R for response.

Instrumental conditioning was introduced by Skinner (1938) and requires that the animal under experimentation performs an arbitrary action (e.g. pressing a lever, walking around) in the presence of an initially meaningless stimulus (e.g. a flickering light). If the action is "appropriate" to the given stimular situation, the animal receives a reward (e.g. food). Otherwise, it receives nothing or punishment, depending on the particular experimental design being applied. Thus, this learning paradigm strongly relies on providing the animal with a reinforcement signal dependent on the action performed. This can be conceptualized as a qualitative feedback. The neural learning rules implementing this type of conditioning at the neuron level are known as *reinforcement-based rules*.

Note that the natural progression in the degree of feedback supplied suggests the use of a quantitative error signal to guide learning. This is represented in the third row of Figure 1 under the name of **input-output teaching**. Here, after presenting an input to the system and observing the emitted response, a teacher supplies the desired output whose difference with the emitted one provides the error signal which is fed back to the system. This is an entirely closed-loop learning process that requires perfect knowledge of the input-output pairs to be associated. The most widely used neural learning rules follow this scheme and we call them *error-minimisation rules*.

The mathematical formulation of the three types of learning rules above will be presented in Section 5. To set up the appropriate context, we provide a brief introduction to neural computing in the next section.

4. Fundamentals of neural computing

The basic elements of neural network models are the *neuron*, the *architecture* into which many neurons are arranged and interconnected, and an *adaptation rule* for modifying the connection strengths. These elements are presented below; for a more in depth treatment, the reader is referred to (Hertz et al., 1993).

4.1. Artificial neurons

The typical artificial neuron used in most neural network models is shown in Figure 2, where the correspondence of the different elements with their biological counterparts has been made explicit.

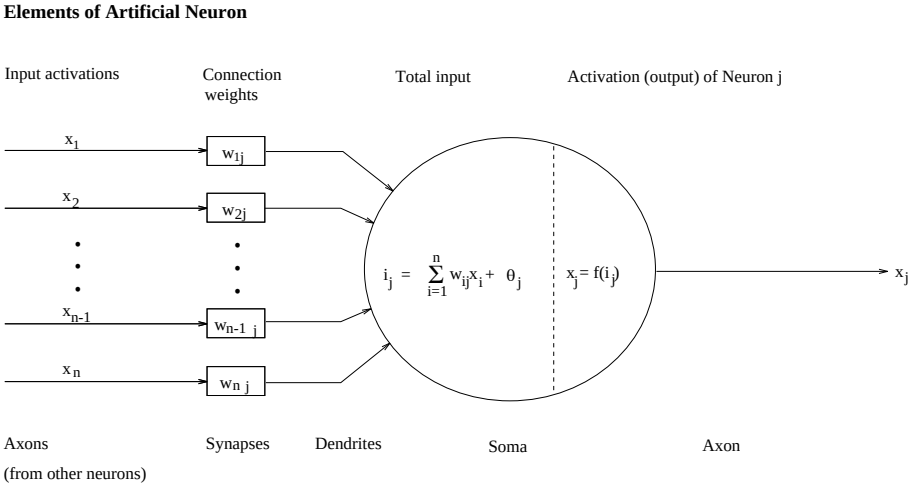


Figure 2. The neuron model.

Like biological neurons, artificial neurons are simple processing units which may receive many inputs from other neurons, but produce a single output. A neuron's output is the weighted sum of all incoming activations from connected neurons, modified by a certain transfer function f:

$$x_j(t+\delta t) = f[\sum_{i=1}^{n} w_{ij}(t)x_i(t) + \theta_j] \qquad (1)$$

where θ_j is a bias or threshold term, which serves as a sort of zero adjustment for the overall input stimulus to the neuron, and w_{ij} is the synaptic weight of the connection from neuron i to neuron j. Excitatory and inhibitory connections are represented simply as positive or negative connection weights, respectively.

Four of the most common transfer functions used are shown in Figure 3.

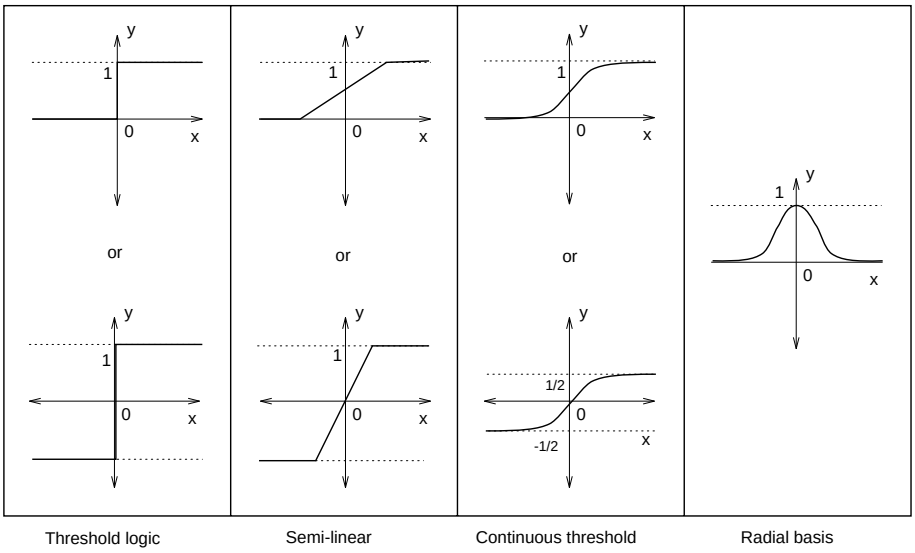


Figure 3. Most common neuron activation functions.

Threshold logic functions were used to implement binary neurons in early perceptron networks (Minsky and Papert, 1969). Although it was demonstrated by McCulloch and Pitts (1943) that networks of such neurons had the computational power of a Turing machine, they proved to be quite limited in their learning capabilities.

Semi-linear functions permit real-valued computation, but are still not differentiable at their transition points, which limits the type of learning rules that may be implemented on them.

The introduction of differentiable transfer functions, such as the sigmoid or the hyperbolic tangent, together with the backpropagation rule (see Section 5) was a major breakthrough in the neural network field. The sigmoid has the following equation:

$$x_j = f(i_j) = \frac{1}{1 + e^{-\alpha i_j}} \tag{2}$$

where i_j is the total input to neuron j, and α is a constant that determines the width of the sigmoid's central transition region.

Radial basis functions depart from the threshold behaviour in that they are not monotonously increasing, but attain high values in a bounded input domain. The Gaussian function is the most widely used example:

$$x_j = f(i_j) = e^{-\frac{1}{2\sigma_j^2}\sum_{i=1}^{n}(x_i - c_i^j)^2} \tag{3}$$

where c_i^j is the center of neuron j's transfer function for input i, and σ_j is the width of neuron j's transfer function.

While threshold logic and semi-linear transfer functions are adequate for discrete tasks, such as pattern association and classification, continuous threshold and radial basis functions are also useful for applications involving continuous mappings, such as those encountered in system identification and control.

The artificial neuron depicted so far has a static behaviour, since its output is a function of only the current inputs. This type of neuron is typically used for tasks consisting of essentially static input/output mappings. By making a neuron's activation dependent on its previous activation state, information with temporal (dynamic) content may be appropriately processed (Griñó et al., 2000). Dynamic neurons are useful for applications in which sequential or time-varying input patterns are encountered, such as speech processing and signal filtering. One method for implementing dynamic activation in neurons is to allow them to sum their activation value over a given time interval. Another is to use recurrent (feedback) connections, by including the neuron's own output as one of its inputs.

Most artificial neural networks to date are based on the simple neuron model depicted above, although more elaborate models which mimick biological reality more closely (Torras, 1985) have recently arousen interest in connection with a new model of computation consisting of networks of spiking neurons (Gerstner, 1999).

4.2. Network architectures

Topological structure is an important characteristic of neural network models, since their functional behaviour depends crucially on it. Consequently, networks are often classified according to their topological features (see Figure 4).

The two main elements which determine a network's architecture are the *layers* of neurons it contains and the *connectivity* between neurons in the same and different layers.

Two major classes of networks are feedforward and recurrent networks. Networks with forward connections pass information forward to successive layers, but not back to preceding ones. The most classical model of *feedforward network* is the multi-layer perceptron (MLP),

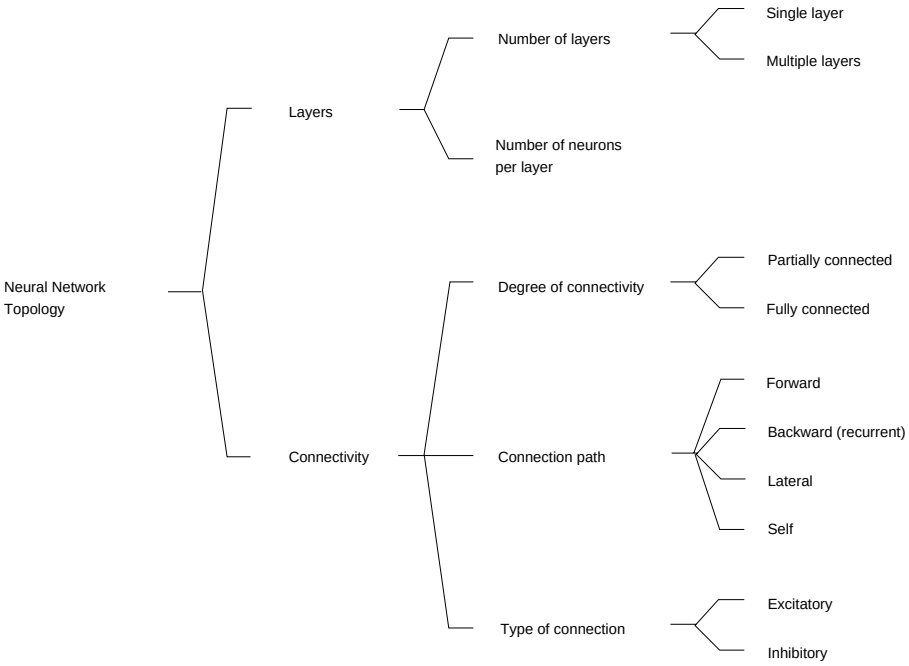


Figure 4. Characterization of neural network topology.

which connects inputs and outputs through one or more so called "hidden" layers. It has been shown that MLP are universal approximators when the hidden units have monotonically increasing differentiable transfer functions (Cybenko, 1989; Funahashi, 1989; Hornik et al., 1989) or Gaussian-shaped transfer functions (Girosi and Poggio, 1990).

If a network has backward connections as well, it is termed a *recurrent network*. Neurons can also have recurrent connections with themselves (self connections). The Hopfield network (Hopfield, 1982; Hopfield and Tank, 1985) and the Boltzmann machine (Aarts and Korst, 1988) are examples of single-layer fully-connected recurrent networks. Other networks have lateral connections between neurons of the same layer. One important class of networks with both excitatory and inhibitory connections are competitive-learning networks.

Several of these network models will be described later in relation to their adaptation capabilities (Section 5) or their use in a robotic setting (Section 7).

4.3. Processing levels

Neural networks process information at two different time scales. On the one hand, there is the short-term propagation of activity through

the network according to equation (1). This has computational interest only in the case of recurrent networks, some of which perform as optimization machines (Aarts and Korst, 1988; Peterson and Södeberg, 1989; Bofill et al., 1995; Bofill and Torras, 2001).

On the other hand, as in the nervous system, the effectiveness or strength w_{ij} of the connections between neurons varies along time, this being what endows neural networks with adaptive capabilities. This long-term level of processing is the focus of this paper, since it lies at the base of robot sensorimotor adaptation, and it will presented in the following section.

5. Neural adaptivity

Adaptivity in neural networks refers to the process of iteratively modifying the connection weights in order to attain a desired input-output behaviour. The procedures used to produce these weight changes are called "learning rules", many of which borrow their names and functionality from Behavioral Psychology, as explained in Section 3. There, we have distinguished between three types of rules: correlational, reinforcement and error-minimisation rules (refer to Figure 1), according to the amount of feedback they require.

5.1. Correlational rules

These rules adjust a connection weight according to the correlation between the activations of the two neurons connected. They can all be considered variants of the classical **Hebbian learning rule** (Hebb, 1949), whose expression in terms of the generic neuron model (1) is:

$$w_{ij}(t+1) = w_{ij}(t) + cx_i(t)x_j(t), \tag{4}$$

where c is a positive constant that determines the speed of learning. To prevent the unbounded growth of weights, different normalization procedures have been used, the most common one being that based on the euclidean metrics.

This rule can be viewed as a unineuronal analog of classical conditioning: one has only to pair repeatedly an unconditioned stimulus x_1 with a conditioned stimulus x_2, assigning initially to w_{1j} a value high enough to force the neuron to fire ($x_j = 1$) when activating x_1; after some pairings, the neuron will discharge when only x_2 is presented.

The most extensively studied application of the Hebbian rule is the implementation of *associative memories* (Hinton and Anderson, 1981), which are network learning models able to carry out pattern association

tasks. The three most interesting aspects of this kind of networks: resistance to noise, addressing by content, and generalisation capability, derive from the distributed way in which information is stored.

The Hebbian rule has also been incorporated into *competitive-learning networks* (Rumelhart and Zipser, 1985), which consist of a set of hierarchically layered neurons, each neuron receiving excitatory input from the layer immediately above. Futhermore, the neurons in each layer are grouped into disjoint clusters, each neuron in a cluster inhibiting all other neurons within the cluster. The name "competitive learning" comes from the fact that the neurons within a cluster "compete" with one another to respond to the pattern appearing on the layer above; the more strongly any particular neuron responds, the more it shuts down the other members of its cluster, which therefore becomes a winner-take-all network (Feldman and Ballard, 1982). A cluster containing n neurons can be considered an n-ary feature, every stimulus pattern being classified as having exactly one of the n possible values of this feature. It has been proved that, if the stimulus patterns naturally fall into classes, the system will find exactly these classes and the attained classification will be very stable. However, when presented with arbitrary input environments, competitive learning models can become very unstable and the need appears of stabilizing their response through the use of specialized mechanisms, leading to *adaptive resonance models* (Grossberg, 1987). These are recurrent modular networks able to form a new cluster whenever they are presented with an input pattern that is very different from the patterns previously seen.

Following the same line of competitive learning, Kohonen (1988) has proposed to use *self-organizing feature maps* (SOM) to learn mappings that preserve topography (i.e. neurons that are spatially close in the network learn to be maximally activated by input vectors close according to the euclidean metrics). Essentially, this is realized through two-layer networks with intralayer lateral inhibition and interlayer plastic excitatory connections. A self-organized map is thus a winner-take-all network, where neuron k wins if it satisfies:

$$\sum_i w_{ik} x_i \geq \sum_i w_{ij} x_i, \ \forall j. \tag{5}$$

The learning rule for this type of network is:

$$w_{ij}(t+1) = w_{ij}(t) + c h_k(j)(x_i(t) - w_{ik}(t)), \tag{6}$$

where $h_k(.)$ is a Gaussian function centered at k used to modulate the adaptation steps as a function of the distance to the winning neuron.

An updated state-of-the-art on SOMs can be found in (Kohonen, 2001) and (Oja and Kaski, 1999).

5.2. Error-minimisation rules

These rules work by comparing the response to a given input pattern with the desired response and then modifying the weights in the direction of decreasing error. Depending on whether the desired response is specified at the single neuron or overall network levels, the gradient of the error with respect to each synaptic weight will necessarily be or may only be locally computable; moreover, the repertoire of learning tasks that can be carried out in the latter case is wider than that accomplishable in the former case.

The most classic error-correction rule that requires specification of the desired response for each single neuron is the **perceptron learning rule** (Minsky and Papert, 1969). Its expression in terms of the generic neuron model that we use as reference (1) is:

$$w_{ij}(t+1) = w_{ij}(t) + c(x_j^*(t) - x_j(t))x_i(t), \tag{7}$$

where both the desired response $x_j^*(t)$ and the actual response $x_j(t)$ of the neuron j are binary, since the f in equation (1) is taken to be a threshold logic function. The same consideration about normalization made for the Hebbian rule applies also here.

The networks of neurons using the perceptron learning rule, called *perceptrons*, are well-suited to carry out linearly-separable pattern classification tasks. However, for non-linearly-separable tasks, a trade-off occurs between the discriminative abilities and learnability in these networks. Minsky and Papert (1969) proved that topological features such as connectedness and symmetry cannot be discriminated by two-layer perceptrons (see Figure 5), but if more layers are introduced then the learning rule can no longer be applied, since there is no way to compute the desired response for each single neuron from the desired network output. This led to the development of backpropagation (see below).

The **LMS learning rule** (Widrow and Hoff, 1960) is expressed by the same equation (7), but considering $x_j^*(t)$ and $x_j(t)$ to take real values and the f in equation (1) to be a semi-linear function. When the parameter c is made to approach zero with time, the weights of a two-layer network equipped with this rule –called *Madaline*– converge to a configuration that minimises the square error between the actual output and the desired one. This is why this rule is called the LMS rule, for "least mean squares".

An extension of the error-minimisation rules so far described to the case where the desired response is specified only for a subset of neurons (those whose outputs constitute the output of the network) is **backpropagation** (LeCun, 1985; Rumelhart et al., 1986). As its name

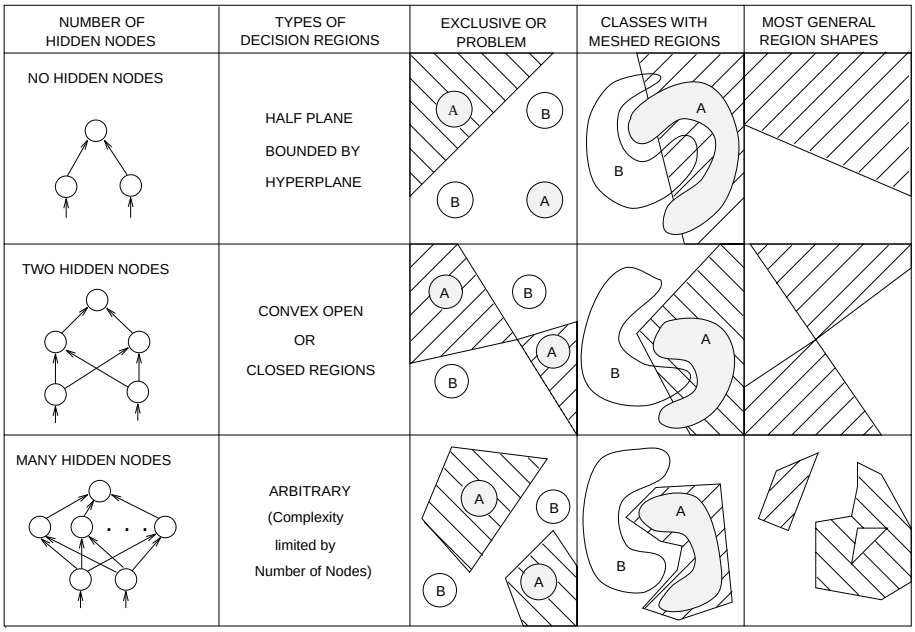


Figure 5. Discriminative abilities of two-layered and three-layered perceptrons.

indicates, it proceeds by propagating error signals from the output neurons back to the sensory neurons, through all intermediate layers, so that appropriate corrections can be applied to all connection weights. Note that this procedure works only for layered feedforward networks.

Backpropagation generalises the perceptron rule by minimising the mean square error E between the actual and the desired responses to all input patterns, through the repeated application of the rule:

$$w_{ij}(t+1) = w_{ij}(t) - c\frac{\partial E}{\partial w_{ij}}. \tag{8}$$

A crucial point is that the f in equation (1) must now be a continuously differentiable function, such as the sigmoid or the hyperbolic tangent, as described in Section 4.1.

Note also that the LMS learning rule is a particular instance of this generic rule, since for the former the transfer function f is semi-linear and thus $\partial x_j/\partial w_{ij} = x_i$, leading to:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}} = (x_j^* - x_j)x_i. \tag{9}$$

By incorporating this result into (8), equation (7) is obtained.

Rumelhart et al. (1986) have applied the generic rule (8) to multilayered feedforward networks of neurons with a sigmoidal transfer function. In this case, taking $\alpha = 1$ in equation 2 for simplicity, $\partial x_j/\partial w_{ij} =$

$x_i x_j(1 - x_j)$ and the factor $\partial E/\partial x_j$ has to be calculated from the activity levels x_k of the neurons in the next layer. Hence, starting with the neurons in the output layer, for which $\partial E/\partial x_j = (x_j^* - x_j)$, the computation proceeds backwards:

$$\frac{\partial E}{\partial x_j} = \sum_k \frac{\partial E}{\partial x_k} \cdot \frac{\partial x_k}{\partial x_j} = \sum_k \frac{\partial E}{\partial x_k} w_{jk} x_k (1 - x_k) \tag{10}$$

and therefore:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}} = x_i x_j (1 - x_j) \sum_k \frac{\partial E}{\partial x_k} w_{jk} x_k (1 - x_k). \tag{11}$$

Backpropagation is the most well-known neural learning algorithm and the most widely used in practice. It is especially well-suited for solving feature discovery tasks; features relevant for discrimination between input patterns get progressively encoded in the activity of the neurons belonging to intermediate layers. Of course, it has the usual drawbacks of all gradient descent techniques, namely the possibility of getting stuck in local minima and a slow convergence rate. Because of this, numerous acceleration procedures have been proposed.

Moreover, backpropagation suffers from *catastrophic forgetting* of the previously learned patterns when trained with a new pattern (Ruiz and Torras, 2000). Thus, techniques to prevent forgetting by explicitly minimising degradation while encoding a new pattern (Park et al., 1991; Ruiz and Torras, 1995) and by introducing noise (An, 1996; Ruiz and Torras, 2002b) have been devised.

Another problem that need to be coped with when applying backpropagation is *overfitting*: the neural network converges to an approximation tailored to the training samples, which does not reflect the underlying function, and thus yields a high generalisation error. This is usually addressed by using methods for model complexity control (Bishop, 1995; Cherkassky, 2002) and, in particular, regularisation. An interesting observation is that many such methods lead to functional invariance (Ruiz and Torras, 2001a; Ruiz and Torras, 2001b), i.e., they converge to the same function irrespective of network size for fixed regularisation parameters.

5.3. Reinforcement rules

These rules do not require being supplied with the desired responses, either at the single neuron or at the overall network levels, but instead a measure of the adequacy of the emitted responses suffices. This measure is reinforcement, which is used to guide a random search process to maximise reward. Hence, reinforcement rules can be considered neuronal analogs of instrumental conditioning in that the neuronal

spontaneous responses are favored or weakened through the application of certain reinforcement schemes.

Here we will describe only the most widely used reinforcement rule, namely the **associative search learning rule** (Barto et al., 1981), which incorporates the required source of randomness in the transfer function of the neuron model used, i.e. a noise with Gaussian distribution is added to the weighted sum of inputs in equation (1). The simplest expression of this rule is:

$$w_{ij}(t+1) = w_{ij}(t) + cx_i(t)x_j(t)r(t) \tag{12}$$

where $r(t)$ is the reinforcement signal.

Note that this is the reinforcement-based counterpart of the correlational rules (equation 4).

A neuron model equipped with this rule learns to maximise $r(t)$ for each stimulus situation. If $r(t)$ is a random variable, its mathematical expectation is instead maximised. The neural networks that incorporate the above rule are called *associative search networks* and, if certain conditions are satisfied, they learn to respond to each stimulus situation $X_l = (x_{l1}, \ldots x_{ln})$ of a set $\{X_1, \ldots X_k\}$ repeatedly presented, with the vector $Y = (y_1, \ldots y_m)$ that maximises the reinforcement function r. The conditions that have to be satisfied are: (a) the function r has to be unimodal, and (b) for each neuron, the subset of stimulus situations in which the optimum response is 0 has to be linearly separable from the corresponding subset in which the optimum response is 1.

Depending on whether the reinforcement signal is provided at the overall network level or is particularised for each single neuron, the structural credit-assignment problem does or does not arise. This is the problem of correctly assigning credit or blame to the action of each neuron that contributed to the overall evaluation received. When dynamical situations need to be considered, because what is of interest is a temporal sequence of events, then a temporal credit-assignment problem also arises. Instead of assigning credit or blame to the action of each neuron in the network, credit or blame must here be assigned to each action in a sequence. In (Sutton, 1988) it has been proposed to use *temporal-difference methods* for this purpose. Methods of this type have been embodied, for example, in "critic modules" used in conjunction with reinforcement learning approaches. The goal of these modules in this setting is to produce an heuristic reinforcement signal which, by predicting future outcomes, is more informed than that directly supplied by the environment.

Sutton and Barto (1998) provide a very thorough and up-to-date introduction to reinforcement learning, and Kröse (1995) offer a view of the several ways in which it has been applied to robots.

6. Neurocontrol approaches

It follows from the preceding sections that neural networks can be viewed as general procedures for approximating nonlinear mappings given a set of inputs and some information on the corresponding outputs. Now, how can they be used in a robot control setting? This section presents the three approaches that have so far been proposed.

Generally, a system to be controlled (be it either an articulation, a wheel, or an entire robot) can be characterized by a transition function g_1 and an output function g_2. The control signal $\mathbf{u}(t)$ in conjunction with the current state of the system $\mathbf{x}(t)$ determines the next state $\mathbf{x}(t+1) = g_1(\mathbf{u}(t),\mathbf{x}(t))$. In each state $\mathbf{x}(t)$, the system produces an output $\mathbf{y}(t) = g_2(\mathbf{x}(t))$.

A controller can be thought of as an *inverse model* of the system in that, given a desired output $\mathbf{y^*}(t)$ and the current state, the controller has to generate the control signal that will produce that output.

The most straightforward neural control approach, named **direct inverse modelling**, uses the system itself to generate input-output pairs and trains the inverse model directly by reversing the roles of inputs and outputs (Figure 6). The applicability of this approach is restricted to systems characterized by one-to-one mappings (otherwise, the inverse is a one-to-many mapping) and its success depends on the quality of the sampling (the inputs have to be selected so that the induced outputs cover adequately the output space). Jordan and Rumelhart (1992) provide a detailed treatment of these issues.

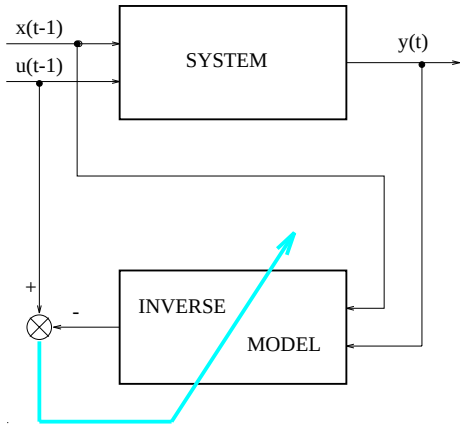


Figure 6. Direct inverse modelling approach.

The **forward modelling** approach proceeds in two stages. In the first stage a forward model of the system is learned from input-output pairs. The second stage consists of composing the obtained forward

model with another network and training the composition of the two to approximate the identity mapping (Figure 7). The weights of the forward model are held fixed in this second stage, while the weights of the controller network undergo adaptation. In the case that the forward mapping is many-to-one, this approach can be biased to find a particular inverse with certain desired properties.

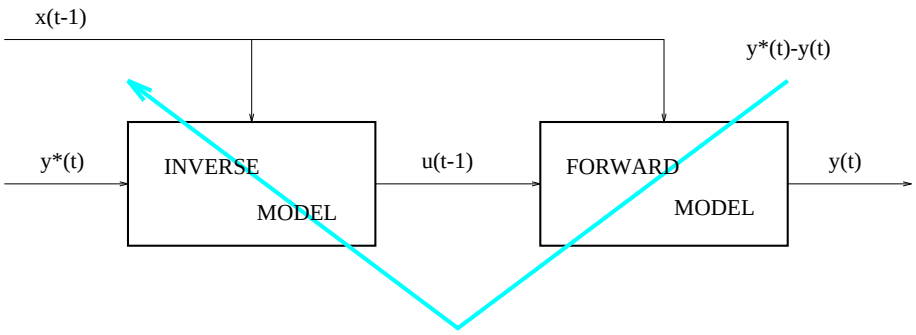


Figure 7. Forward modelling approach.

The first stage can be obviated if the Jacobian matrix of the system is known. This is the matrix of partial derivatives of outputs with respect to inputs. In the case of a robot arm, the Jacobian permits deriving the linear and angular velocity of the end-effector from the joint velocities (Fu et al., 1987). Different procedures to compute the Jacobian can be found in (Samson, 1990). By a straightforward application of the chain rule, the Jacobian can be used to derive the input errors as a function of the output errors (Kröse and van der Smagt, 1993). This is precisely the purpose of the forward model in the forward modelling approach and, therefore, its first stage is no longer needed.

Note also that, if instead of the desired outputs, only a reinforcement signal (i.e. a measure of how well the system is performing) is available, then this approach can still be applied. In this case, the forward model encompasses both the system and the reinforcement function, and it is used in the second stage to derive the suitable weight modifications to be performed in the controller network. Alternatively, the first modelling stage can be obviated and the reinforcement signal can be used directly to determine weight modifications in the controller network.

Finally, the **feedback-error learning** approach requires having a conventional feedback controller linked to the system and the role of the neural controller is to make the feedback error signal tend to zero (Figure 8). One interesting characteristic of this approach is that there is no need of a separate training phase, but instead the system is trained during operation.

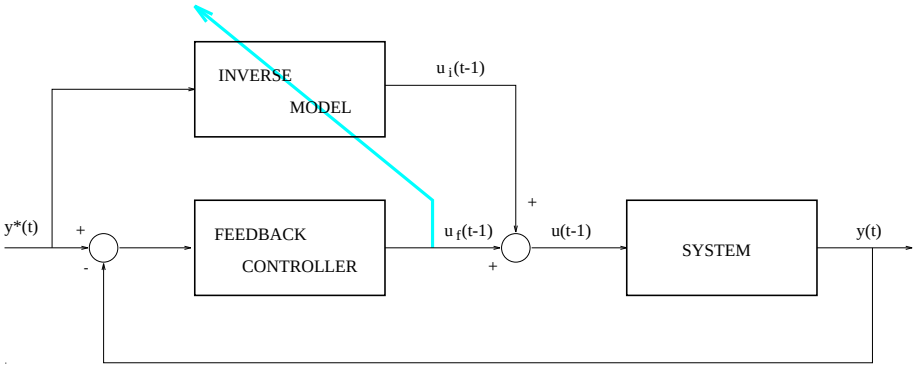


Figure 8. Feedback-error learning approach.

Jordan (1993) and the chapter on 'sensorimotor learning' in (Arbib, 1995) provide a detailed treatment of the concepts presented in this section. In Chapter 9 of (Miller et al., 1990b), Kawato carries out a comparison between the three approaches.

It is worth noting that the application of error-correction schemes does not necessarily require training with a teacher that tells the correct answer (which would be unfeasible in many robotic applications), neither does it imply the exclusive use of error-minimisation learning rules. To state it more explicitly, the above approaches can be applied under both supervised and unsupervised (or self-supervised) training modes and through the use of correlational, reinforcement or error-minimisation learning rules.

7. Robot sensorimotor mappings

Motion control, both in biological and technological systems, relies strongly on sensorimotor mappings. These mappings vary considerably (Torras, 1989), depending not only on the nature of the involved sensors and actuators, but also on the goal pursued.

Tasks to be carried out by robots are usually specified in world coordinates (or, alternatively, in terms of sensor readings), while robot moves are governed by their actuator's variables. For instance, a sealing task may be specified as a given curve in 3D space or as following a prespecified visual pattern, but it has ultimately to be translated into currents sent to the motors governing the different joints. Therefore, robot control critically depends on the availability of accurate mappings from physical space or sensor space to joint space or motor space. The discussion in what follows is centered on mappings required for robot

arms to work, but similar arguments apply to the case of mobile robots (Millán and Torras, 1992; Millán and Torras, 1999).

For a gripper to reach a desired position and orientation in space, the robot controller must access a mapping relating workspace coordinates to joint coordinates. This is called *inverse kinematics mapping*, because the natural (direct) map is that relating the values of the joint coordinates defining an arm configuration to the position and orientation of its end-effector (hand, gripper,...) in the workspace.

If a desired end-effector trajectory is specified instead, then the controller should resort to an *inverse dynamics mapping* relating such trajectory to the forces and torques that need to be exerted at the different joints to realize it. Note that this mapping, which is again called inverse for the same reason above, cannot be characterized uniquely in terms of inputs and outputs, it being instead dependent on state variables (or the short-term history of inputs) as it is usually the case with dynamic systems.

For tasks entailing the achievement of a goal using sensory feedback, programming even in terms of the coordinates of the workspace can be very complex. An example of this is the insertion of components with small clearance, since devising a detailed force-control strategy that performs correctly in all possible situations, and subject to real-world conditions of uncertainty and noise, is extremely difficult. What is needed to accomplish this type of tasks is an appropriate *sensori-motor mapping* relating sensory patterns to motor commands. Such a mapping can be thought of as resulting from the composition of a map from sensor space to physical space, with another from physical space to motor space. This relay through physical space is necessary if an interpretation in terms of the coordinates of the workspace is required to program the robot, but it may appear to be superfluous from a behavioural viewpoint (see the chapter on 'limb geometry, neural control' in (Arbib, 1995) for a discussion about intermediate representations).

The diversity of the aforementioned mappings sometimes hides what they have in common: an underlying highly nonlinear relation between a continuous (often hard to interpret) input domain and a continuous motor domain; a relation that is very difficult (when not impossible) to derive analytically. Furthermore, because of environmental changes or robot tear-and-wear, the mappings may vary in time and then one would like the controller to adapt to these variations, without any human intervention if possible. Therefore, a way of learning (or tuning) these mappings automatically while robots move is highly desirable. Since, as we have mentioned, neural networks are essentially procedures for approximating nonlinear mappings, they constitute a good tool to attain the desired adaptivity.

7.1. Inverse kinematics

Making robots adaptive to changes in their own geometry (e.g., link bending, encoder shifting and other wear-and-tear deformations occasioned by regular functioning) would certainly widen their range of application. Since these geometric changes affect the robot inverse kinematics, the interest of using neural networks to approximate such mapping has been widely recognized. Especially when the operation conditions of the robot (in space, underwater, etc.) make it very hard to recalibrate it.

Along this line DASA (Daimler-Benz Aerospace S.A.), in the framework of the Advanced Servicing Robot project targeted at unmanned space stations, proposed an application of maintenance of electronic equipment that required the automatic recalibration of a 6-dof robot in-situ, since recalibration through teleoperation from earth is a very time-consuming task due to communication delays. After reviewing previous approaches to the learning of inverse kinematics, we will present the solution implemented in the REIS robot included in the space-station mock-up located at DASA's R&D laboratory, in Bremen, Germany (see Figure 9).

Figure 9. Space station mock-up at Daimler-Benz Aerospace, Bremen.

The most simple way to tackle the learning of inverse kinematics is to apply the *direct inverse modelling approach* to a *feedforward network*

using *backpropagation*. Many researchers have experimented with this approach by applying it to a variety of robot models (Kröse and van der Smagt, 1993; Torras, 1993). The one-to-many problem mentioned in Section 6 appears here, because most robots can access the same position and orientation under several joint configurations, but the problem is obviated by designing the training set in a way that the resulting function is one-to-one (by using robot configurations with the links always in the same half-spaces).

Jordan and Rumelhart (1992) showed that a minimum-norm constraint or temporal smoothness constraints could be easily incorporated into the learning procedure to bias the choice of the particular inverse function obtained, when the same type of network was used under a *forward modelling approach*.

The conclusion reached after extensive experimentation with feed-forward networks under both approaches is that a coarse mapping can be obtained quickly, but an accurate representation of the true mapping is often not feasible or extremely difficult. The reason for this seems to be the global character of the approximation obtained with this type of networks using sigmoid units: every connection weight has a *global* effect on the final approximation that is obtained (Kröse and van der Smagt, 1993).

An obvious way to avoid this global effect is of course using local representations, so that every part of the network is responsible for a small subspace of the total input space. Thus, Ritter et al. (1992) have used a *Kohonen self-organizing map* together with the *LMS rule* to learn the visuomotor mapping of a robot arm with three degrees of freedom (dof) in 3D space. A *direct inverse modelling approach* under a completely unsupervised training mode is used. The target position of the end-effector is defined as a spot registered by two cameras looking at the workspace from two different vantage points (refer to Figure 10, but disregarding the inner cube and the arrow pointing towards the gripper).

Neurons are arranged in a 3D lattice to match the dimensionality of physical space. It is expected that learning will make this lattice converge to a discrete representation of the workspace. Each neuron i has associated a four-dimensional vector $\mathbf{w}_i$ representing the retinal coordinates of a point of the workspace. The response of the network to a given input $\mathbf{u}$ is the vector of joint angles θ_k and the 3×4 Jacobian matrix $\mathbf{A}_k$ associated with the winning neuron k, i.e. that satisfying:

$$\mathbf{w}_k \mathbf{u} \geq \mathbf{w}_i \mathbf{u}, \ \forall i. \tag{13}$$

Note that this is just equation (5) rewritten in matrix form.

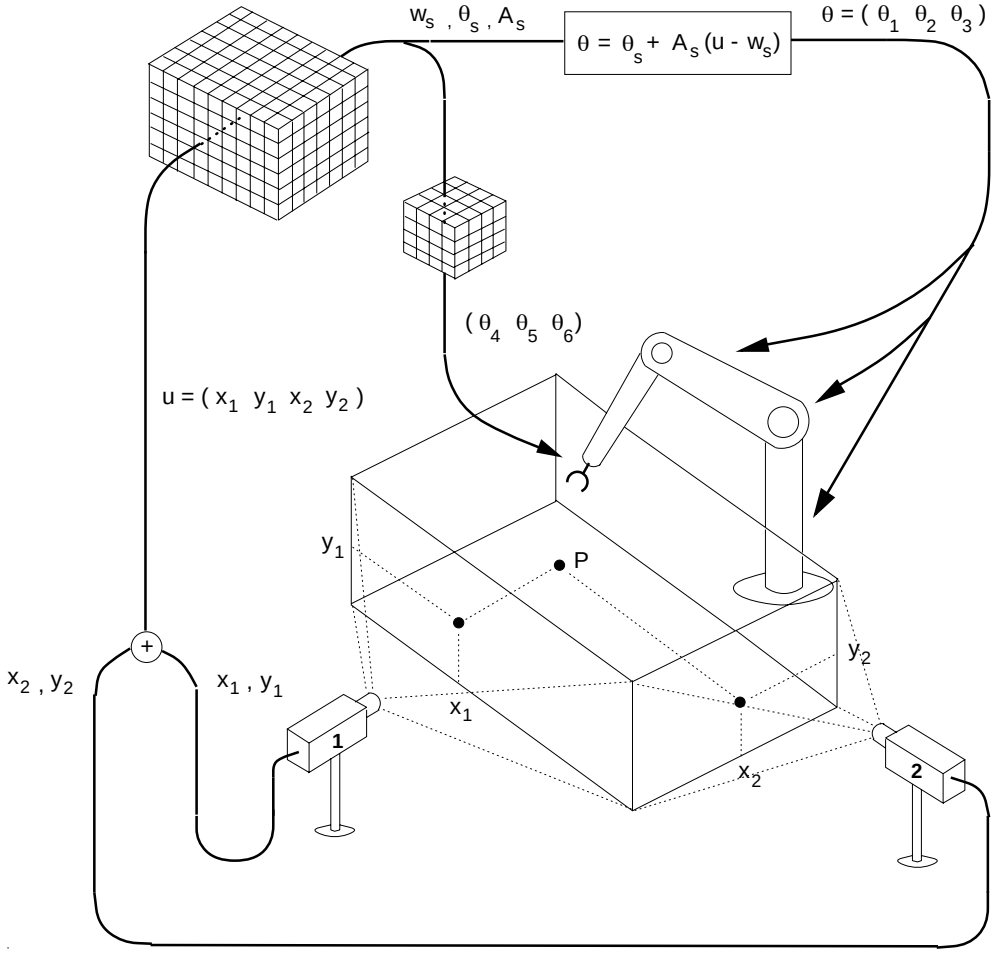


Figure 10. Set-up for inverse kinematics learning.

The joint angles produced for this particular input are then obtained with the expression:

$$\theta(\mathbf{u}) = \theta_k + \mathbf{A}_k(\mathbf{u} - \mathbf{w}_k). \tag{14}$$

A learning cycle consists of the following four steps:

1. First, the classical Kohonen rule (equation 6) is applied to the weights:

$$\mathbf{w}_i^{new} = \mathbf{w}_i^{old} + c\ h_k(i)\ (\mathbf{u}(t) - \mathbf{w}_k(t)), \tag{15}$$

where c is the learning rate and $h_k(.)$ is a Gaussian function centered at k used to modulate the adaptation steps as a function of the distance to the winning neuron.

2. By applying $\theta(\mathbf{u})$ to the real robot, the end-effector moves to position $\mathbf{u}'$ in camera coordinates. The difference between the desired position $\mathbf{u}$ and the attained one $\mathbf{u}'$ constitutes an error signal that permits applying an error-correction rule, in this case the LMS rule:

$$\theta^* = \theta_k + \Delta\theta = \theta_k + \mathbf{A}_k(\mathbf{u} - \mathbf{u}'). \tag{16}$$

3. By applying the correction increment $\mathbf{A}_k(\mathbf{u}-\mathbf{u}')$ to the joints of the real robot, a refined position $\mathbf{u}''$ in camera coordinates is obtained. Now, the LMS rule can be applied to the Jacobian matrix by using $\Delta\mathbf{u} = (\mathbf{u} - \mathbf{u}'')$ as the error signal:

$$\mathbf{A}^* = \mathbf{A}_k + (\Delta\theta - \mathbf{A}_k\Delta\mathbf{u})\frac{\Delta\mathbf{u}^T}{\|\Delta\mathbf{u}\|^2}. \tag{17}$$

4. Finally, the Kohonen rule is applied to the joint angles:

$$\theta_i^{new} = \theta_i^{old} + c'\ h'_k(i)\ (\theta *_i - \theta_k(t)), \tag{18}$$

and the Jacobian matrix:

$$\mathbf{A}_i^{new} = \mathbf{A}_i^{old} + c'\ h'_k(i)\ (\mathbf{A}_i^* - \mathbf{A}_k(t)), \tag{19}$$

where again c' is the learning rate and $h'_k(.)$ is a Gaussian function centered at k used to modulate the adaptation steps as a function of the distance to the winning neuron.

Extensive experimentation by Ritter et al. (1992) and other groups has shown that the network self-organizes into a reasonable representation of the workspace in about 30.000 learning cycles. This should be taken as an experimental demonstration of the powerful learning capabilities of this approach, because the conditions in which it is made to operate are the worst possible ones: no a priori knowledge of the robot model, random weight initialization, and random sampling of the workspace during training.

This basic model has been extended in three directions to cope with higher-dof robots. First, a hierarchical version, consisting of a 3D SOM whose nodes have associated a 2D SOM each, was applied to a 5-dof robot. The 3D net encodes the workspace as before, while each 2D subnet approximates the end-effector orientation space at the corresponding position (Ritter et al., 1992).

Ruiz de Angulo and Torras (1997) adapted this hierarchical model to suit a practical setting (refer now to the complete model in Figure 10). Thus, instead of learning the kinematics from scratch, only the deviations from the nominal kinematics embedded in the original robot controller are learnt. This, together with informed initialization and sampling, as well as several modifications in the learning algorithm aimed at improving the cooperation between neurons, led to a speed-up of two orders of magnitude with respect to the original model.

The resulting algorithm constitutes the core of the recalibration system that was installed in the REIS robot included in the space-station mock-up located at DASA, as mentioned above. Figure 9 is a photograph of such a set-up, where the different racks containing the electronic cards that the robot should maintain are shown. The robot must reach the handles of the racks with enough precision to be able to pull them out and, afterwards, extract a faulty card in order to replace it by another one. Although testing in this set-up has been constrained by the need to preserve robot integrity, the system has proven able to correct large miscalibrations of the robot: 95% of the decalibration was corrected with the first 25 movements, this percentage raising to 98% after 100 movements (Ruiz de Angulo and Torras, 1997). Moreover, other desirable features in stand-alone applications, such as parameter stability, are guaranteed.

The third extension of the basic model in equations (13)-(19) relies on the generalisation of SOMs to parameterized SOMs (called PSOMs). The idea is to turn the discrete representation into a continuous one by associating a basis function to each neuron, so that a parameterized mapping manifold is obtained. Moreover, PSOMs make no distinction between inputs and outputs, thus encoding bidirectional mappings. The PSOM reduces considerably the number of training samples required to attain a given precision as compared to the SOM (Walter and Ritter, 1996), allowing the learning of the full inverse kinematics of a 6-dof robot with less than 800 movements.

The main drawback of using neural networks to approximate the inverse kinematics of robot arms is precisely the high number of training samples (i.e., robot movements) required to attain an acceptable precision. A trick has recently been proposed (Ruiz de Angulo and Torras, 2002a), valid for most industrial robots, that greatly reduces the number of movements needed to learn or relearn the mapping to a given accuracy. This trick consists in expressing inverse kinematics as a composition of learnable functions, each having half the dimensionality of the original mapping. A training scheme to learn these component functions has also been proposed. Experimental results obtained by using PSOMs, with and without the decomposition, show that the time

savings granted by the proposed scheme grow polynomically with the precision required.

Recently, the development of humanoid robots has raised the interest in learning inverse kinematics. Due to the many dof's involved, the aim is no longer learning the mapping for the whole workspace, but focussed on a specific trajectory. Following the trend of using localized representations, D'Souza et al. (2001) have applied a supervised algorithm –locally weighted projection regression– in this context, with promising results.

7.2. Inverse dynamics

When the robot dynamics needs to be taken into account, as in trajectory following, the control learning problem becomes more involved. An inverse dynamics mapping relating end-effector accelerations to the required joint forces and torques needs now to be considered. Only very rarely this mapping can be computed analytically and, if so, it is of course highly specific to a particular robot and payload.

Neural networks have been applied to learning either the inverse dynamics mapping directly or through the error provided by a fixed-gain controller, as described below. Observe that the dynamics case differs from the kinematics one in that, to generate the input-output pairs, one needs some sort of fixed controller driving the robot from the very beginning. As learning of the inverse dynamics mapping proceeds, the output of the neural network becomes more accurate and the effect of the fixed controller tends progressively to zero.

Since the cerebellum is known to be involved in the production and learning of smooth movements, several cerebellar models have been proposed and applied to control robot arms. The pioneer such model was the Cerebellar Model Articulation Controller (CMAC) developed by Albus (1975), but today the debate is still open as to what model best captures the functionality of the cerebellum and whether any such model can constitute a practical option to control robots (van der Smagt and Bullock, 1997). A point of agreement is that the cerebellum constructs an inverse dynamics model as it learns. Thus, cerebellar models have been used for this purpose inside robot controllers.

Miller et al. (1990a) have combined the table look-up facilities provided by CMAC with an error-correction scheme similar to the LMS rule to accomplish the dynamic control of a 5-dof robot. The idea underlying this combination is similar to that of enlarging SOMs with the LMS rule, as described in the preceding section. Here, CMAC is used to represent the state space in a compact and localized manner, as there SOMs were used to cover the robot workspace. To teach the robot to follow a given trajectory, successive points along it are supplied

to both the neural network and a fixed-gain controller and then their responses are added up to command the robot. Therefore, the neural network acts as a feedforward component. After each cycle, the actual command given to the robot together with its current state are used as an input-output pair to train the neural network following a *direct inverse modelling* approach. As learning progresses, the CMAC network approximates the inverse dynamics mapping so that the difference between the current and desired states tends to zero, and consequently the neural network takes over control from the fixed-gain controller. The network converges to a low error (between 1 and 2 position encoder units) within 10 trials, provided enough weight vectors are used.

The same trajectory learning task above has been tackled by Miyamoto et al. (1988) by using the *feedback-error learning* approach. They used directly as error signal the output of the feedback controller, which can be interpreted as a local linearization of the inverse dynamics mapping if the learning rate is sufficiently small. This error measure is less accurate than that used by Miller et al. (1990a), but has the advantage of being directly available in the control loop, thus avoiding the computation of the current state of the robot required in the direct inverse modelling approach. The authors report that, after training the robot to follow a trajectory lasting 6 seconds for 300 trials, the average feedback torque decreased from a few hundreds to just a few units, demonstrating that the neural network had taken over control from the fixed-gain controller. Moreover, the mean square error in the joint angles decreased steadily 1.5 orders of magnitude.

7.3. Force-motor mappings

As mentioned at the beginning of Section 7, in tasks requiring the insertion of components with small clearance, devising a detailed force-control strategy is extremely difficult. Thus, the possibility of using neural networks to learn the action to apply in response to each force pattern (i.e., the appropriate sensorimotor mapping) looks very attractive.

Gullapalli et al. (1994) have used an associative reinforcement learning system to learn active compliant control for peg-in-hole insertion using a 6-dof robot. The system takes the sensed peg positions and forces, as well as the previous position command, as inputs, and produces a new position command as output. Thus, eighteen real values are entered into a network with two hidden layers of backpropagation units, and six real values are produced by its output layer of stochastic reinforcement-learning units. The reinforcement signal depends on the discrepancy between the sensed and the desired position of the peg, with a penalty term being activated whenever the sensed forces on the

peg exceed a preset maximum. The training runs start with the peg at a random position and orientation with respect to the hole, and end when either the peg is successfully inserted or 100 time steps have elapsed. Experimental results show that, after 150 trials, the robot is consistently able to complete the insertion. Moreover, the time to insertion decreases continuously from 100 to 45 time steps over the subsequent 500 training runs.

7.4. Visuomotor mappings

Depending on the task to be performed and the camera-robot arrangement, visuomotor mappings take different forms. Thus, in eye-hand coordination, where cameras external to the robot are used to monitor the pose (position and orientation) of its end-effector, a mapping from the camera coordinates of a desired end-effector pose to the joint angles that permit attaining that pose is sought. This mapping is closely related to the inverse kinematics one, especially if the camera coordinates of selected points in the end-effector uniquely characterize its pose. Therefore, the same models used to learn inverse kinematics have been applied to the learning of the visuomotor mapping underlying eye-hand cooordination (Ritter et al., 1992).

A camera mounted on a robot arm is used in tasks such as visual positioning and object tracking. The goal of these tasks is to move the camera so that the image captured matches a given reference pattern. The target is thus no longer a position of the robot in space but a desired image pattern, and the desired visuomotor mapping needs to relate offsets w.r.t. that pattern with appropriate movements to cancel them. In visual positioning, the scene is assumed to be static and the main issue is to attain high precision. Applications include inspection and grasping of parts that cannot be precisely placed (e.g., in underwater or space settings). The aim of object tracking is to maintain a moving object within the field of view, speed being here the critical parameter instead of precision.

The classical way of tackling these tasks consists of defining a set of image features (corners, holes, etc.) and then deriving an interaction matrix relating 2D shifts of these features in the image to 3D movements of the camera (Espiau et al., 1992). The advantages of applying neural networks to this task are the direct learning of the interaction matrix, as well as avoiding the costly matching of features in the current and reference images.

Note that the visuomotor mapping can be implemented with or without a relay through physical space, depending on how the movements of the camera are commanded. Figure 11 shows the visual positioning

scheme in the case of using such a relay, which therefore requires having an inverse kinematics mapping in place.

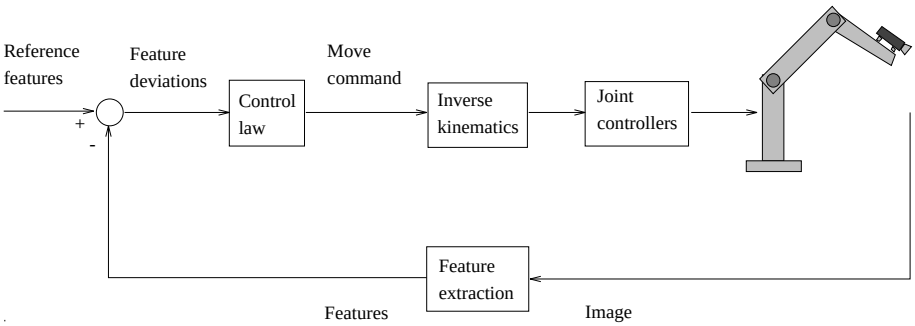


Figure 11. Visual positioning scheme.

This scheme has been used in an application developed for Thomson Broadcast Systems (Wells et al., 1996) for the inspection of large objects (e.g., ship hulls, airplane wings, etc.). Since these objects are difficult to move, it is the camera that has to travel to attain a pre-specified position and orientation with respect to the object. The developed camera control system consists of a *feedforward network* trained with *backpropagation* under a *direct inverse modelling approach*. The training procedure consists of moving the camera from the reference position to random positions and then using the displacement in image features together with the motion performed as input-output pairs. In operation, the robot is commanded to execute the inverse of the motion that the network has associated to the given input.

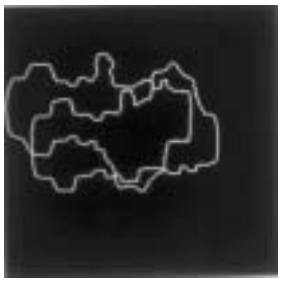
The key option in this work is the use of global image descriptors, which permits avoiding the costly matching of local geometric features in the current and reference images. By using a statistical measure of variable interdependence (the mutual information criterion), sets of global descriptors as variant as possible with each robot dof are selected from a battery of features, including geometric moments, eigenvectors, pose-image covariance vectors and local feature analysis vectors (Wells and Torras, 2001). The results obtained with a 6-dof show that, after 10.000 learning cycles, translation and rotation errors are lower than $2mm$ and 0.1 degrees, respectively. Figure 12 shows the robot-mounted camera and the reference image of an object to be inspected (a water valve), together with several snapshots along the visual positioning process. In this case, the silhouette of the object could be readily extracted and 32 Fourier descriptors coding it were used as image features. It can be observed that, after 7 movements, the captured image is practically registered with the reference one.

a) Robot and camera

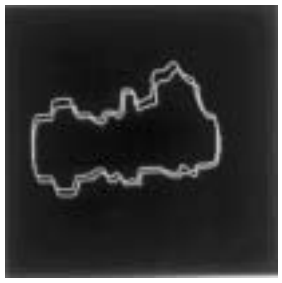
b) Reference image

c) Initial image

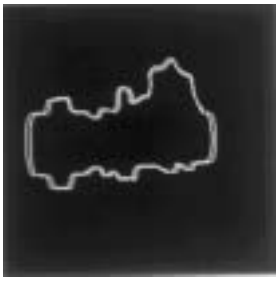
d) Initial contours



e) After 1 movement



f) After 5 movements



g) After 7 movements

h) Final image

Figure 12. Visual positioning system developed in collaboration with Thomson Broadcast Systems.

Concerning object tracking, Schram et al. (1996) have used a feed-forward network together with a conjugate gradient learning algorithm to make a camera track a cart moving arbitrarily on a table. A visuo-motor mapping relating the current and past visual coordinates of the cart with joint displacements is built on-line as the robot moves. Only two robot dofs need to be controlled, and thus the network has two outputs, while several numbers of inputs have been tried. The tracking performance is shown to improve as more previous positions of both the cart and the robot are used, attaining an average lag of only $8mm$ in the case of seven inputs.

8. Summary and concluding remarks

This paper has reviewed the ways in which neural computing may help to increase sensorimotor adaptivity in robots. First, the most common neuron models and network topologies have been presented, to next describe in more detail the mechanisms of neural adaptivity. These mechanisms have been inspired in the learning paradigms of Behavioural Psychology (classical and instrumental conditioning), and fall into three classes that require progressively more feedback: correlational, reinforcement and error-minimisation rules.

Some trends in the development of these learning rules deserve notice, since they have parallels in other disciplines dealing with adaptivity at diferent scales, such as Evolutionary Computation and Artificial Intelligence. The first trend is that of progressing from binary variables to continuous ones. This entails moving from discrete search spaces and classification tasks to manifold representations and function approximation. Then, issues such as model complexity control and functional invariance become very important.

A second trend is that of progressively replacing local feedback for more global one, this globalisation taking place both spatially and temporally. The first learning rules proposed required feedback to be supplied to each single neuron. Backpropagation made a big step forward in allowing feedback to be supplied at the overall network level (spatial globalisation). Reinforcement learning has greatly contributed to dealing with deferred feedback (temporal globalisation).

The dichotomy between locality and globality appears also in the state space representation. Correlational rules are often incorporated into network models that build localised representations (such as SOM, ART and CMAC), while the strength of most models based on error-minimisation and reinforcement rules lies precisely in the distributed (global) way in which information is represented across all the net-

work weights. In the localised representations, appropriately tuning the neighbourhood size is a key issue.

Moving from off-line to on-line learning is another trend observed in neural computing. Initial learning procedures were designed to work in a batch mode (with all training samples supplied at the same time), while a later challenge was to incorporate new samples into an already trained network. Sequential learning addresses this challenge by explicitly seeking to avoid catastrophic forgetting.

Finally, let us mention the important role that randomness plays in learning. This has been widely acknowledged in many contexts, but specifically in neural computing noisy inputs and weights have proven useful for regularisation (a complexity control method), and randomness is of course a key ingredient of reinforcement learning.

After the overview of neural adaptivity, the paper has focused on its application to robot control. This basically entails the learning of nonlinear mappings relating stimuli to responses. Three neurocontrol approaches have been presented: direct inverse modelling, forward modelling and feedback-error learning. Then several robotic applications have been surveyed, classified according to the underlying mapping that needs to be approximated: inverse kinematics, inverse dynamics, force-control mapping and visuomotor mapping. Each application has been presented in terms of the neurocontrol approach used and the network model and learning strategy applied.

The learning of inverse kinematics makes robot arms adaptive to changes in their own geometry (e.g., link bendings, encoder shiftings, etc.), while learning of the remaining three mappings allows robots to cope with changing environments (e.g., different loads, moving objects, etc.).

A first remark stemming from the survey of robotic applications is that in the case of mappings that can be easily sampled, it seems sufficient to apply a direct inverse modelling approach combined with a plain error-minimisation procedure. Some simple inverse kinematics mappings and visuomotor mappings used for visual positioning have been learned in this way. If the input space is complex, then many researchers have resorted to a combination of correlational rules for the efficient coding of that space, with error-minimisation rules to build the appropriate association with the outputs. The use of SOMs to encode the robot workspace or the sensor space, as well as the application of CMAC to the coding of the robot dynamics state space, fall into this category. In both cases, the LMS rule is used to build the appropriate input-output mapping: inverse kinematics in the former case and inverse dynamics in the latter one. In the case that a measure of the error is directly available in the control loop, as it happens in

some inverse dynamics applications, then it seems natural to apply a feedback-error learning approach. The very nature of this approach points towards the use of error-minimisation rules, although if the error measure is thought of as a qualitative rather than a quantitative one, then it becomes a sort of reinforcement and the learning scheme can be considered as a reinforcement-based one. Finally, when the task is specified as a goal to be reached using sensory feedback, without making explicit the movements necessary to reach it, then the only possibility is to resort to reinforcement learning schemes, which depend just on the availability of a measure of success rather than an error measure.

The number of learning cycles required ranges widely in the applications described, depending on the complexity of the mapping to be learned as well as on the accuracy required. Note that learning time is directly related to the number of training samples, each of which entails at least one robot movement. And robots are slow as compared to computers. Therefore, minimising the number of training samples is of paramount importance in the application of neural networks to robotics, and many efforts are currently oriented in this direction (e.g., adaptive sampling, function decomposition).

Acknowledgments

This work has been partially supported by the Catalan Research Commission, through the "Robotics and Control" group, and the Spanish Science and Technology Commission (CICYT) under contract TAP99-1086-C03-01. The author thanks Bernd Maediger, from Daimler-Benz Aerospace, and Christophe Venaille, from Thomson Broadcast Systems, for providing the specifications and the robot set-ups for the applications described in Sections 7.1 and 7.4 in the framework of the former Esprit project CONNY, and also for supplying the photographs in Figures 9 and 12, respectively. Thanks also to Gordon Wells, from Telecom Italia, for supplying Figures 2 to 5.

References

Aarts EHL and Korst JHM (1988) Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing. John Wiley and Sons

Albus JS (1975) A new approach to manipulator control: The cerebellar model articulation controller (CMAC). Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control 97: 220-227

An G (1996) The effects of adding noise during backpropagation training on generalization performance. Neural Computation 8: 643-674

Arbib MA (1995) Handbook of Brain Theory and Neural Networks. MIT Press, Cambridge, MA (An updated second edition will appear in November 2002)

Bäck T, Fogel DB and Michalewicz Z (eds) (1997) Handbook of Evolutionary Computation. Oxford University Press, New York, and Institute of Physics Publishing, Bristol

Beyer H-G and Schwefel H-P (2002) Evolution strategies - A comprehensive introduction. Natural Computing 1(1): 3-52

Barto AG, Sutton RS and Brouwer PS (1981) Associative Search Network: A reinforcement learning associative memory. Biological Cybernetics 40: 201-211

Bishop C (1995) Neural Networks for Pattern Recognition. Oxford University Press

Bofill P, Fontdecaba E and Torras C (1995) Optimization networks for the generation of Block Designs. Journal of Artificial Neural Networks 2(4): 303-312

Bofill P and Torras C (2001) Neural cost functions and search strategies for the generation of Block Designs: An experimental evaluation. Intl. Journal of Neural Systems 11(2): 187-202

Cherkassky V (2002) Model complexity control and statistical learning theory. Natural Computing 1(1): 109-133

Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems 2(4): 303-314

D'Souza A, Vijayakumar S and Schaal S (2001) Learning inverse kinematics. Proc. IEEE/RSJ Conf. on Intel. Robots and Systems, Maui, Hawaii, USA, pp. 298-303

Dorigo M (ed) (1996) Special issue on 'Learning Autonomous Robots'. IEEE Trans. on Systems, Man and Cybernetics, Part B: Cybernetics 26(3)

Espiau B, Chaumette F and Rives P (1992) A new approach to visual servoing in robotics. IEEE Trans. on Robotics and Automation 8(3): 313-326

Feldman JA and Ballard DH (1982) Connectionist models and their properties. Cognitive Science 6: 205-254

Fu KS, González RC and Lee CSG (1987) Robotics: Control, Sensing, Vision, and Intelligence. McGraw-Hill Book Company, New York

Funahashi K-I (1989) On the approximate realization of continuous mappings by neural networks. Neural Networks 2(3): 183-192

Gerstner W (1999) Spiking neurons. In: Maass W and Bishop CM (eds) Pulsed Neural Networks. MIT Press, Cambridge, MA

Girosi F and Poggio T (1990) Networks and the best approximation property. Biological Cybernetics 63: 169-176

Goldberg DE (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA

Griñó R, Cembrano G and Torras C (2000) Nonlinear system identification using additive dynamic neural networks. Two on-line approaches. IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications 47(4): 150-165

Grossberg S (1987) Competitive learning: from interactive activation to adaptive resonance. Cognitive Science 11: 23-63

Gullapalli V, Barto AG and Grupen R. (1994) Learning admittance mappings for force-guided assembly. Proc. IEEE Intl. Conf. on Robotics and Automation, IEEE Computer Society Press, Los Alamitos, CA, pp. 2633-2638

Hebb DO (1949) The Organization of Behavior. Wiley, New York

Hertz J, Krogh A and Palmer RG (1993) Introduction to the Theory of Neural Computation. Addison-Wesley

Higuchi T, Iwata M and Liu W (eds) (1997) Evolvable Systems: From Biology to Hardware. Springer-Verlag, Berlin Heidelberg New York

Hinton GE and Anderson A (1981) Parallel Models of Associative Memory. Erlbaum, Hillsdale, NJ

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proceedings National Academy of Sciences USA 79: 2554-2558

Hopfield JJ and Tank DW (1985) 'Neural' computation of decisions for optimization problems. Biological Cybernetics 52: 141-152

Hornik K, Stinchcombe M and White H (1989) Multilayer feedforward networks are universal approximators. Neural Networs 2(5): 359-366

Husbands P and Meyer JA (1998) Proc. 1st European Workshop on Evolutionary Robotics (EvoRobot'98). Springer-Verlag, Berlin Heidelberg New York

Jordan MI (1993) Computational aspects of motor control and motor learning. In: Heuer H. and Keele S (eds) Handbook of Perception and Action: Motor Skills, Academic Press, New York

Jordan MI and Rumelhart DE (1992) Forward models: Supervised learning with a distal teacher. Cognitive Science 16, 307-354

Kohonen T (1988) Self-Organization and Associative Memory (second edition). Springer-Verlag, Berlin Heidelberg New-York Tokyo

Kohonen T (2001) Self-Organizing Maps (third edition). Series in Information Sciences 30, Springer-Verlag, Berlin Heidelberg New York Tokyo

Koza JR (1992) Genetic Programming: On Programming Computers by means of Natural Selection. MIT Press, Cambridge, MA

Kröse BJA (ed) (1995) Special issue on 'Reinforcement Learning and Robotics'. Robotics and Autonomous Systems 15(4)

Kröse BJA and van der Smagt PP (1993) An Introduction to Neural Networks (5th edition). University of Amsterdam

LeCun Y (1985) Une procedure d'aprentissage pour reseau au seuil assymetrique. Proc. of COGNITIVA, pp. 599-604

McCulloch WS and Pitts WH (1943) A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. 5: 115-133

Millán JR and Torras C (1992) A reinforcement learning connectionist approach to robot path finding in non-maze-like environments. Machine Learning 8(3/4): 363-395

Millán JR and Torras C (1999) Learning sensor-based navigation. In: Morik K, Kaiser M and Klingspor V (eds) Making Robots Smarter: Combining Sensing and Action through Robot Learning. Kluwer Academic Publishers, Boston, MA

Miller WT, Hewes RP, Glanz FH and Kraft LG (1990a) Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller. IEEE Trans. on Robotics and Automation 6(1): 1-9

Miller WT, Sutton RS and Werbos PJ (1990b) Neural Networks for Control. MIT Press, Cambridge

Minsky M and Papert S (1969) Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge, MA (An expanded second edition appeared in 1988)

Mitchell T, Franklin J and Thrun S (1996) Recent Advances in Robot Learning. Kluwer Academic Publishers, Boston, MA

Miyamoto H, Kawato M, Setoyama T and Suzuki R (1988) Feedback-error-learning neural network for trajectory control of a robotic manipulator. Neural Networks 1: 251-265

Morik K, Kaiser M and Klingspor V (1999) Making Robots Smarter: Combining Sensing and Action through Robot Learning. Kluwer Academic Publishers, Boston, MA

Oja E and Kaski S (eds) (1999) Kohonen Maps. Elsevier Science, Amsterdam, Holland

Omidvar O and van der Smagt P (1997) Neural Systems for Robotics. Academic Press, San Diego, CA

Park DC, El-Sharkawi MA and Marks II RJ (1991) An adaptively trained neural network. IEEE Trans. on Neural Networks 2(3): 334-345

Pavlov IP (1927) Conditioned Reflexes. Oxford University Press

Peterson C and Södeberg B (1989) A new method for mapping optimization problems onto neural networks. Intl. Journal of Neural Systems 1(1): 3-22

Ritter H, Martinetz T and Schulten K (1992) Neural Computation and Self-Organizing Maps. Addison Wesley, New York

Ruiz de Angulo V and Torras C (1995) On-line learning with minimum degradation in feedforward networks. IEEE Trans. on Neural Networks 6(3): 657-668

Ruiz de Angulo V and Torras C (1997) Self-calibration of a space robot. IEEE Trans. on Neural Networks 8(4): 951-963

Ruiz de Angulo V and Torras C (2000) A framework to deal with interference in connectionist systems. AI Communications 13(4): 259-274
Ruiz de Angulo V and Torras C (2001a) Architecture-independent approximation of functions. Neural Computation 13(5): 1119-1135
Ruiz de Angulo V and Torras C (2001b) Neural learning invariant to network size changes. Proc. Intl. Conf. on Artificial Neural Networks (ICANN'01), Vienna, Austria, Lecture Notes in Computer Science 2130: 33-40
Ruiz de Angulo V and Torras C (2002a) Learning inverse kinematics via cross-point function decomposition. Proc. Intl. Conf. on Artificial Neural Networks (ICANN-2002), Madrid, Spain, Lecture Notes in Computer Science 2415: 856-861
Ruiz de Angulo V and Torras C (2002b) A deterministic algorithm that emulates learning with random weights. Neurocomputing 48(1-4): 975-1002
Rumelhart DE, Hinton GE and Williams RJ (1986) Learning representations by back-propagating errors. Letters to Nature 323: 533-535
Rumelhart DE and Zipser D (1985) Feature discovery by competitive learning. Cognitive Science 9: 75-112
Samson C, LeBorgne M and Espiau B (1990) Robot Control: The Task Function Approach. Oxford Engineering Science Series 22, Oxford Science Publications
Schram G, van der Linden FX, Kröse BJA and Groen FCA (1996) Visual tracking of moving objects using a neural network controller. Robotics and Autonomous Systems 18: 293-299
Simon HA (1969) The Sciences of the Artificial. MIT Press, Cambridge, MA
Skinner BF (1938) The Behavior of Organisms: An Experimental Analysis. Appleton Century
Steels L (1995) The Biology and Technology of Intelligent Autonomous Agents. NATO ASI Series F, Springer-Verlag, Berlin Heidelberg New York
Sutton RS (1988) Learning to predict by the methods of temporal differences. Machine Learning 3: 9-44
Sutton RS and Barto AG (1998) Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA
Torras C (1985) Temporal-Pattern Learning in Neural Models. Lecture Notes in Biomathematics 63, Springer-Verlag, Berlin Heidelberg New York
Torras C (1989) Sensorimotor integration in robots. In: Ewert P and Arbib MA (eds) Visuomotor Coordination: Experiments, Comparisons, Models and Robots, Plenum Press, pp. 673-689
Torras C (1993) Symbolic planning versus neural control in robots. In: Rudomín P, Arbib MA, Cervantes-Pérez F and Romo R (eds.) Neuroscience: From Neural Networks to Artificial Intelligence, Research Notes in Neural Computing 4: 509-523, Springer-Verlag: Berlin Heidelberg New-York
Torras C (ed) (2001) Special issue on 'Neural Networks at IJCAI'01'. Intl. Journal of Computational Intelligence and Applications 1(4)
van del Smagt P and Bullock D (eds) (1997) Can Artificial Cerebellar Models Compete to Control Robots? DLR Technical Report #515-97-28, German Aerospace Center
van de Velde W (ed) (1993) Special issue on 'Towards Learning Robots'. Robotics and Autonomous Systems 8(1-2)
Walter J and Ritter H (1996) Rapid learning with parametrized self-organizing maps. Neurocomputing 12: 131-153
Wells G and Torras C (2001) Assessing image features for vision-based robot positioning. Journal of Intelligent and Robotic Systems 30(1): 95-118
Wells G, Venaille Ch and Torras C (1996) Vision-based robot positioning using neural networks. Image and Vision Computing 14: 715-732
Widrow B and Hoff ME (1960) Adaptative switching capatibility and its relation to the mechanisms of association. Kybernetik 12: 204-215
Ziemke T and Sharkey N (eds) (1998) Special issue on 'Biorobotics'. Connection Science 10(3-4)