Connectionist Learning for Control: An Overview[†]

Andrew G. Barto

Department of Computer and Information Science
University of Massachusetts, Amherst MA 01003

COINS Technical Report 89-89
September 1989

Abstract—This report is an introductory overview of learning by connectionist networks, also called artificial neural networks, with a focus on the ideas and methods most relevant to the control of dynamical systems. It is intended both to provide an overview of connectionist ideas for control theorists and to provide connectionist researchers with an introduction to certain issues in control. The perspective taken emphasizes the continuity of the current connectionist research with more traditional research in control, signal processing, and pattern classification. Control theory is a well–developed field with a large literature, and many of the learning methods being described by connectionists are closely related to methods that already have been intensively studied by adaptive control theorists. On the other hand, the directions that connectionists are taking these methods have characteristics that are absent in the traditional engineering approaches. This report describes these characteristics and discusses their positive and negative aspects. It is argued that connectionist approaches to control are special cases of memory–intensive approaches, provided a sufficiently generalized view of memory is adopted. Because adaptive connectionist networks can cover the range between structureless lookup tables and highly constrained model–based parameter estimation, they seem well–suited for the acquisition and storage of control information. Adaptive networks can strike a balance between the tradeoffs associated with the extremes of the memory/model continuum.

[†]The author acknowledges the support of the Air Force Office of Scientific Research, Bolling AFB, through grant AFOSR–87–0030, which made this chapter possible, and the support of the King's College Research Centre, King's College Cambridge, England, where much of it was written. Special appreciation is expressed to Chuck Anderson, Judy Franklin, Mike Jordan, Mitsuo Kawato, Rich Sutton, and Paul Werbos for useful discussions of the material presented in this chapter and many helpful suggestions on improving its presentation. This report is based on a talk given at the NSF Workshop on Neurocontrol, University of New Hampshire, October 1988. A version of this report will appear in *Neural Networks for Control*, T. Miller, R. S. Sutton, and P. J. Werbos (Eds.), The MIT Press, Cambridge, Massachusetts.

Contents

1 Introduction

This report is an introductory overview of learning by artificial neural networks with a focus on the ideas and methods most relevant to the control of dynamical systems. The perspective taken emphasizes the *continuity* of the current research on artificial neural networks, which I call connectionist research, with more traditional research in control, signal processing, and pattern classification. Tying connectionist learning methods to existing theory is important for those interested in connectionist approaches to adaptive and learning control because control theory is a well–developed field with a large literature, and many of the learning methods being described by connectionists are closely related to methods that already have been intensively studied by adaptive control theorists. On the other hand, the directions that connectionists are taking these methods have characteristics that are absent in the traditional engineering approaches. In this report I describe these characteristics and discuss their positive and negative aspects.

This report is not intended to be a review of all research on connectionist learning, and no attempt is made to provide exhaustive references to the connectionist literature. The review by Hinton [39] covers a wider range of learning methods (although it does not focus on control), Mars [63] reviews connectionist approaches to robot control, and there are other publications that carefully elaborate many of the observations I make. Here attention is restricted to basic issues concerning supervised and reinforcement learning, and I assume that readers already know about the most popular connectionist learning algorithms. Learning algorithms for recurrently–connected networks and unsupervised learning are not discussed even though these topics are relevant to applications in control. I also do not discuss the hardware side of connectionist research—taking for granted that the suitability of network methods for parallel implementation is one of the attractive aspects of this approach; and I do not discuss any of the important issues that arise when relating connectionist research to neuroscience.

2 History

Figure 1 is a sketch of my mental model of the recent history of connectionist research in relation to engineering and artificial intelligence (AI). Although it strongly reflects my own history and biases,[1] I think it accurately represents several important features of connectionist research. The first event shown in the sketch is the divergence of research into the lines labeled "engineering methods" and "sequential symbol manipulation methods," although these separate lines of research did not emerge as suddenly as suggested by Figure 1, and there were other offshoots marking the end of what Minsky and Papert [70] called the "romantic period" during which biologically–inspired computational schemes were first studied (also see Arbib [5]). I associate this divergence with Minsky's 1961 paper "Steps Toward Artificial Intelligence" [69], in which he argued that symbolic "articulated" representations have powerful advantages over the feature–vector representations to which connectionist methods were, and arguably still are, restricted. In fact, one might label these two main lines of research "feature vectors" on the left and "articulated representations" on the right.

A feature vector is a list of measurements, having numerical or logical values, made of some situation, state, object, etc. An articulated representation, on the other hand, can have a recur-

[1] In this report, when I refer to "engineering" and "AI" I am often using these terms in an admittedly stereotyped way that does not reflect the diversity of these fields. I ask the reader not to take this usage too literally and conclude that I am pigeon–holing researchers or their research.

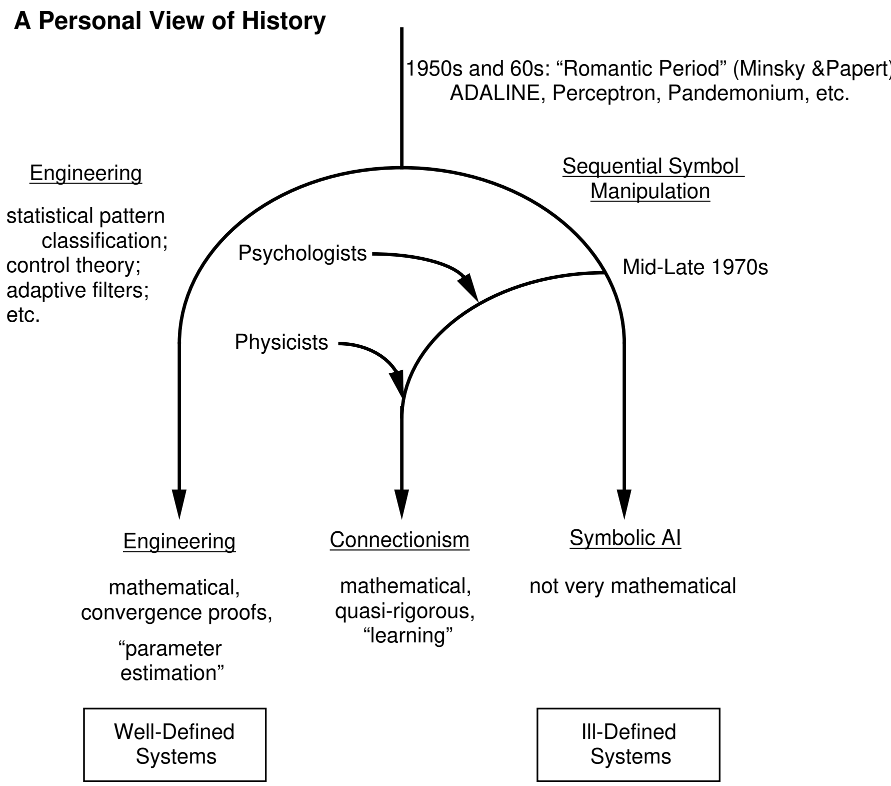

Figure 1: A personal view of the history of connectionist research.

sive structure allowing "wholes" to be represented in the same way that "parts" are represented. This allows descriptions of situations to be recursively embedded within the representations of more complex situations in a way that does not have an obvious counterpart for feature vectors. Despite considerable research effort directed toward bringing feature vectors and articulated, or symbolic, represention closer together (e.g., refs. [22, 40, 78, 97, 98]), there is still a fundamental gulf between these representational styles. The engineering tradition shown by the left line in Figure 1 has not addressed these representational issues because feature vector representations are natural for the kinds of prediction and control problems with which engineers are concerned. Consequently, discussions of the relevance of connectionist methods to control tend not to address the questions about connectionist analogs of symbolic representations that are occupying much of the attention of AI and cognitive science researchers. Although symbolic processing will undoubtedly play an important role in intelligent control systems—perhaps with connectionist approaches to these problems playing key roles—there seems to be ample opportunity for progress in control before one needs to consider the kinds of articulated representations that dominate symbolic AI. Perhaps it is also significant that at this workshop we will not be hearing a great deal about the profound problems in scaling network techniques to large problems (e.g., Judd [48, 49, 50]). This is certainly relevant for control, but considerable progress in control seems possible before these barriers are approached.

Another major feature of my sketch of history shown in Figure 1 is that the line representing current connectionist research branches off from the sequential symbol manipulation line—not the engineering line. The significance of this is that while connectionist techniques generally are closely related to engineering methods (numerical feature vectors, gradients, mean–square error, etc.), much connectionist research continues the tradition in AI of addressing problems that are not formulated with a high degree of mathematical structure (such as linearity). Partly because of this, connectionist research is more experimental and heuristic than my impression of modern engineering methodology. For example, many connectionist researchers are currently experimenting with the back–propagation method for training layered networks (independently developed by LeCun [60], Parker [77], Rumelhart, Hinton, and Williams [83], and Werbos [102]) without the aid of a theoretical framework that guides its application and provides guarantees about its results. Additionally, connectionist researchers routinely exercise a kind of representational freedom that is not commonplace in more orthodox engineering applications. For example, a feature vector may be a distributed representation of a kind of symbolic representation (e.g., Rosenfeld and Touretzky [82]) that is very far removed from the lists of measurements of "physical" quantities more natural in engineering applications (position, velocity, etc.). Connectionist methods often employ "expansive representations" (about which I say more in the section "Representation") having dimensionality much higher than the apparent intrinsic dimensionality of a problem.

I do not mean to imply that experiment and heuristics are absent from engineering, or that connectionist researchers can lay sole claim to all of the methods they use. Indeed, in the field of control engineering, there have been numerous efforts to develop more heuristic approaches to control. Examples are provided by Fu [29], Mendel and Fu [64], Saridis and Gilbert [86], and Waltz and Fu [100]. Much of this research was explicitly influenced by the methodology of AI and includes some of the features of the connectionist research discussed in this report. Studies such as these represent early efforts, not depicted in Figure 1, to explore the middle ground between engineering and AI for applications to control. There are undoubtedly many reasons that research of this kind did not lead to major development within control engineering. Although some of these reasons may involve real deficiencies of these methods for certain kinds of problems, I would like to suggest that two additional factors may have been important. First, the movement of AI away from these kinds of mathematically–based methods—the same developments in AI that isolated the connectionism of the 1960s—may have also isolated heuristically–minded research in control. Second, many of the control methods proposed by these researchers had memory requirements that were excessive in terms of the available technology. Now, however, not only is AI somewhat more hospitable to work of this kind, but technological advances have made memory–intensive approaches to control quite feasible. A perspective taken in this report is that connectionist approaches to control are instances of memory–instensive approaches, and that it is now technologically possible to explore the utility of these methods.

One of the strengths of connectionist research is that it lies in this middle ground between engineering and AI—utilizing engineering–like mathematical techniques while simultaneously incorporating expansive representations and an experimental methodology more like that of AI. It is obviously desirable to maintain high standards of mathematical rigor, but an experimental, heuristic approach seems essential both for developing applications involving complex nonlinear systems and for detecting the regularities over classes of problems that can guide rigorous mathematical development. However, as for any approach to complex control problems in which the hypotheses of convergence and stability theorems are not likely to be satisfied, it is important to recognize that some problems are not well suited for the application of connectionist learning methods. These are problems for which control failure is unacceptably costly or for which it is not possible to use proven control principles to safeguard against unacceptable performance. Although connectionist learning methods have the potential for extending the range of control applications, it may turn out that the applications most suitable for these methods will be those related to the control capabilities of animal nervous systems; although this is a very broad class of applications, it is not all–inclusive.

Connectionist research is providing a bridge between lines of research that diverged more than twenty years ago. The current high level of activity in connectionist research can be compared with the excitement that would result if scientists from different but similar planets suddenly found themselves able to communicate. There would be a sudden flow of information as concepts on each side were recast within the other sides' frameworks. Just as it would not be surprising that methods well–known for years on one side were being reformulated using a different language, it would not be surprising that some of the ideas on each side were new to the other. These sociological, or cultural, factors are as important as the technical factors in understanding what has happened in the last several years as interest in connectionist research has exploded.

In the rest of this report, I discuss connectionist learning methods using the framework of parameter estimation as it is understood by engineers, emphasizing the ideas and techniques that I think represent contributions from the connectionist research community.

3 Parameter Estimation

Most of the learning methods studied by connectionists are examples of *parameter estimation* methods. Parameter estimation is central to the fields of pattern classification (e.g., Duda and Hart [23]; Sklansky and Wassel [88]) adaptive signal processing (e.g., Widrow and Stearns [112]), and adaptive control (e.g., Goodwin and Sin [33]). For example, one way to construct a pattern classifier is to assume that the classification rule, or decision rule, is a member of a specific class of rules, and that each rule in the class is specified by selecting values for a set of parameters. Learning is the process of adjusting parameter values based on how well the decision rule performs on a set of training patterns whose correct classifications are supplied by a "teacher."

In signal processing and control, one is often seeking a *model* capable of producing numerical values that match values measured from some physical system. This application of parameter estimation is known as *system identification*. One assumes that the model has a specific mathematical form expressed in terms of a set of unknown parameter values and then uses a parameter estimation method to find good parameter values. The estimation method requires data in the form of examples of how the system being modeled behaves for a collection of inputs. The role of the teacher is played by the system being modeled. In a similar manner, parameter estimation methods can be applied to the problem of identifying the *inverse* of a system (refs. [107, 110, 112]). I discuss system identification, inverse system identification, and other types of problems relevant to control in more detail below.

Another way to apply parameter estimation techniques is distinctively attributable to the connectionist approach: the formation of *associative memory networks*. Although the parameters (the connection weights) of associative memory networks are updated using parameter estimation techniques, connectionists think about associative memory networks differently from the way engineers think about parameterized classes of models. Instead of regarding the parameter estimation process as one of searching for the best–fitting model in a class of models, one thinks of it as storing information in a memory structure. Compared to the usual lookup–table form of memory, associative memory networks have the desirable properties of some degree of noise resistance and content addressability (see refs. [41, 57] for discussions of associative memory networks). Viewed as system identification procedures, on the other hand, associative memory networks can be criticised because they often implement transformations that are not specialized enough to provide good models of the process generating the data.

The idea of an associative memory network is a good example of an idea linking engineering methods with the style of thinking more like that of computer science and AI. It is not surprising that the modern wave of connectionist research began in the mid 1970s with the popularization of associative memory networks, although Widrow and Hoff [109] discussed the associative memory idea in 1960. Recognizing that associative memory and system, or rule, identification are closely related to one another and yet are evaluated by different criteria is important for understanding the relationship between connectionist and traditional applications of parameter estimation. These points are elaborated below.

The essential elements of a parameter estimation problem are as follows (after Goodwin and Sin [33]):

1. *Problem representation*—This is an aspect of a parameter estimation problem that is guided the least by useful theory. What aspects of objects, inputs, states etc. should be measured to provide data for classification or modeling? If basic physical variables are obvious candidates, should they be represented simply as real–valued measurements, or should they be coded in some other way? One characteristic of connectionist research is that representations are often chosen that are suggested by what we know, or think we know, about how nervous systems represent information. These representations often differ from those an engineer might choose. In using connectionist networks, choosing a representation corresponds to selecting how the signals reaching the network's input units are related to the problem under consideration, as well as how the network's output codes the relevant problem variables.

2. *Class of decision rules or models*—This is another aspect of parameter estimation that is guided little by theory. One must select an appropriate class of decision rules or model structures—a choice that includes deciding how many parameters to use and how they

define the decision rule or model. Generally, one must make a compromise between rule or model complexity and its adequacy for the application. Using a connectionist network, this corresponds to choosing the types of units, the structure of the network, and the constraints among connection weights to enforce during training.

3. *Performance criteria*—How does one compare the performance of the different decision rules or models in the delineated class? An important problem here is that one would like to measure the performance of a system without having to completely implement it. The best measure of performance might evaluate the overall, final peformance of the system that makes use of the decision rule or model (for example, in the commercial world, "How well does the device *sell*?"; in the biological world, "How well does the organism *reproduce*?"). However, in practice one has to devise simpler criteria that are closely correlated with overall performance and yet can be measured easily and quickly. For parameter estimation it is common to measure performance by the average of the squared error between the output of the decision rule or model and some target values. The problem of finding easily measured evaluation criteria that can act as surrogates for more basic but hard–to–measure criteria is a central issue in studies of learning and is discussed further in the section "Reinforcement Learning."

4. *Estimation methods*—Two major types of parameter estimation methods are distinguished: *off–line* and *on–line* methods. Off–line methods operate when the training data are available at one time. These data are collected before performing the analysis that determines the parameter values. On–line methods, on the other hand, process the data as it becomes available to the learning system. They must use efficient methods to be able to keep up with the temporal flow of events. Here, I restrict attention to on–line methods because connectionist networks generally implement on–line methods.

5. *Use of a priori knowledge*—The parameter estimation process can be improved by the use of *a priori* knowledge, not only as it is incorporated into the selection of the problem representation and the class of decision rules or models, but also in the form of initial parameter values and constraints among parameter values. Connectionist learning methods are often studied under the assumption that there is little prior information about the task at hand, but this assumption is not a defining characteristic of connectionist research. The use of parameter estimation methods, whether embodied in connectionist networks or not, by no means entails a *tabula rasa* view of learning and memory. On the contrary, the incorporation of prior knowledge can greatly accelerate, and otherwise improve, the learning process.

3.1 Feature Vectors, Decision Rules, and Models

The framework in which parameter estimation techniques are applied is essentially the same for classification and system identification tasks.[2] For a classification task, let x denote any one of the patterns one wishes to classify. For an identification task, let x denote any of the collections of data that will constitute input to the relationship being modeled. In what follows, I call

[2]I do not discuss identification methods using recursive, or auto–regressive, models that are widely studied in engineering. These are models whose past outputs can form part of their input. Identification using recursive models should not be confused with what is sometimes called recursive parameter estimation, or recursive system identification, in which the term recursive essentially means on–line.

all such collections of data *patterns*, even though this is not the usual terminology for system identification tasks. Let X denote the set of all patterns relevant to a given task and assume there are m real–valued functions, $\phi_i : X \to \Re$, $i = 1, \ldots, m$, each producing a measurement of each pattern. Writing $\phi_i(x)$ to denote the value of the i^{th} measurement of pattern x, the vector $\phi(x) = [\phi_1(x), \ldots, \phi_m(x)]^{\mathrm{T}}$ is the *feature vector* description of x. There are no restrictions on the forms of the functions ϕ_i—they can be linear or nonlinear functions of X (assuming X has enough mathematical structure for linearity to make sense). They are given concrete interpretation when a method is applied to a specific real–world problem, and this interpretation constitutes the basic problem representation discussed above.

Of particular importance are linear decision rules and linear models. For a two–class classification task, the set of linear decision rules is parameterized by a parameter vector $w = [w_1, \ldots, w_{m+1}]^{\mathrm{T}}$, and each rule has the form

$$x = \begin{cases} \text{class 1} & \text{if } w_1\phi_1(x) + \cdots + w_m\phi_m(x) > -w_{m+1} \\ \text{class 2} & \text{if } w_1\phi_1(x) + \cdots + w_m\phi_m(x) < -w_{m+1}. \end{cases} \tag{1}$$

This is the linear threshold rule often used in connectionist networks. In an identification task, linear models take the form

$$y(x) = w_1\phi_1(x) + \cdots + w_m\phi_m(x) + w_{m+1}, \tag{2}$$

where $y(x)$ is the model output for input x (this can be extended to models with vector output in the obvious way). For both pattern classification and system identification, one would estimate the value of the parameter vector, w, that produces the best match with the data supplied during training (where best is defined in terms of some specific criterion).

Two conventions are usually followed when decision rules or models in the form of Equations 1 and 2 are used. First, to the list of m functions ϕ_i, one adds an $(m+1)^{\text{st}}$ whose value is always 1; so that one can write the decision rule given by Equation 1 as

$$x = \begin{cases} \text{class 1} & \text{if } w^{\mathrm{T}}\phi(x) > 0 \\ \text{class 2} & \text{if } w^{\mathrm{T}}\phi(x) < 0, \end{cases} \tag{3}$$

where $w^{\mathrm{T}}\phi(x)$ is the inner product of w and $\phi(x)$. Similarly the model given by Equation 2 can be written

$$y(x) = w^{\mathrm{T}}\phi(x). \tag{4}$$

The second convention is to refer to the *time* at which a pattern appears for classification, or a pattern is input to the system being identified. Let ϕ_t, w_t, and y_t respectively denote the value of the feature vector, the parameter vector, and the classifier or model output at time t.

Notice that even in the case of a linear model, the model output is a nonlinear function of the underlying patterns when the ϕ functions are nonlinear functions of the patterns. However, because the ϕ functions are not parameterized, the model (or the decision rule if we ignore the threshold operation in Equation 3) is a linear function of the parameters, and the ϕ functions' structures do not directly enter into the derivation of a parameter estimation method.[3] Despite the flexibility that nonlinear ϕ functions add to linear models, models defined by nonlinear functions

[3]The ϕ functions do influence the parameter estimation process, however, because they determine important characteristics of the the training instances as seen by the linear model. Connectionist approaches make extensive use of these possibilities as discussed in the section "Representation."

of the parameters are of considerable interest because they provide wider latitude for adjusting behavior. For example, a layered network of the appropriate kinds of nonlinear units defines a class of nonlinear models whose parameters can be adjusted by the back–propagation method. I discuss nonlinear models in more detail in the section "Nonlinear Models."

3.2 Parameter Estimation Methods

A typical method for on–line parameter estimation operates as follows. At time step t, there is an estimate, w_t, for the unknown parameter vector that specifies the best model (according to some specified measure) within the class of models under consideration. There is also available a feature vector, ϕ_t, describing the current input pattern. The output of the decision rule or model specified by w_t is determined using the feature vector ϕ_t as input. This output is compared with the target output of the decision rule or model, which is assumed to be supplied by a "teacher." For example, assuming the linear model given by Equation 4, and letting y_t^* denote the target output at time t, a signed error, ϵ_t, is determined:

$$\epsilon_t = y_t^* - y_t = y_t^* - w_t^{\mathrm{T}} \phi_t. \tag{5}$$

In the most widely used methods, a new parameter estimate, w_{t+1}, is determined from the current estimate, w_t, as follows:

$$w_{t+1} = w_t + M_t \, \epsilon_t \, \phi_t, \tag{6}$$

where M_t is the algorithm gain at step t.

A number of algorithms follow the form of Equation 6 but differ in what M_t is and how it is computed. In some algorithms, M_t is a matrix computed in an iterative manner based on training data (see, for example, Goodwin and Sin [33]; Ljung and Söderstrom [62]) but connectionists tend to restrict attention to the simplest case in which M_t is a constant, c. Using a matrix M_t with non–zero terms off the diagonal requires non–local computation that is costly to implement when there is a large number of parameter as in many connectionist applications. In the simplest case of a constant scalar gain, one has the following update equation:

$$w_{t+1} = w_t + c \, \epsilon_t \, \phi_t. \tag{7}$$

If the parameterized linear model given by Equation 4 is assumed, then Equation 7 can be expanded via Equation 5 to yield:

$$w_{t+1} = w_t + c(y_{t+1} - w_t^{\mathrm{T}} \phi_t)\phi_t. \tag{8}$$

This is the estimation method presented by Widrow and Hoff [109], widely known as the LMS (Least Mean Square) rule. If one replaces the linear model of Equation 4 with the linear threshold decision rule given by Equation 3, one obtains the Perceptron learning rule of Rosenblatt [81], which is identical to Equation 8 except that the weighted sum $w_t^{\mathrm{T}} \phi_t$ is replaced by the threshold of the weighted sum as defined by Equation 3.

On–line estimation methods can be understood as incremental methods for attempting to minimize a measure of model error as a function of the parameters. According to this view, the task is to search for the parameter vector that minimizes a measure of classifier or model error over all the training data. The LMS rule (Equation 8), for example, produces a step in parameter space whose expected direction is down the surface giving this overall error, and hence the on–line application of the LMS rule tends to minimize the mean–squared error over the data. The back–propagation method is derived by computing this error gradient for the particular class of models specified by layered networks.

The theory of the LMS rule, and other parameter estimation methods, is very well developed. Under what conditions does a method converge to a fixed estimate? What criterion of best–fit is satisfied by the final estimate? In all cases, there is a requirement that the data set processed by the algorithm satisfies certain properties for good parameter estimates to result. For identification tasks, the system being identified has to be driven with a set of inputs that reveals all of its modes of activity to the parameter estimator. A sufficiently rich set of input signals is said to be *sufficiently stimulating*. Exactly what this means depends on the particular class of systems one is considering. Analogous conditions for classification require a sufficiently varied set of training instances and a training regime that repeats them all sufficiently often. For theoretical treatments of on–line parameter estimation methods see, for example, refs. [23, 33, 62, 88, 112].

Within the framework of parameter estimation just sketched, the two major factors influencing how the framework is applied are the choice of the basic problem representation embodied in the ϕ functions and the choice of the parameterized class of models. It is in terms of these factors that one can see how connectionist researchers tend to part company with more traditional users of parameter estimation techniques.

4 Representation

Because the ϕ functions determining the basic problem representation do not have to belong to a specific restricted class of functions in order for parameter estimation methods to apply, one can exercise wide latitude in selecting these functions. Although some choices are more satisfactory than others (depending on characteristics of the task and the class of models chosen), and "feature selection" has been widely studied as a problem in itself, it is considered largely an art to select suitable ϕ functions. So–called "unsupervised" learning methods[4] can be regarded as procedures for selecting good features, but these methods are beyond the scope of this overview (and, of course, one also has to select a basic representation for an unsupervised method). My aim here is remind the reader of some of the consequences of selecting different types of ϕ functions.

These consequences cover a spectrum of possibilities ranging from lookup tables, on one end, to purely computational schemes, on the other. For example, if the set, X, of patterns is finite, then one can define a set of ϕ functions, $\{\phi_x | x \in X\}$, where

$$\phi_x(y) = \begin{cases} 1 & \text{if } y = x \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Given this representation, a linear model (Equation 4) is just a lookup table, where the ϕ functions provide the indexing scheme and the parameters give the values stored in the table. Note that although information is stored in distinct "locations" when this representation is used—because there is a distinct parameter for each item stored—a sequential search of memory locations is not required to retrieve information if one assumes that each ϕ is implemented by hardware. The collection of ϕ functions acts as a hashing function; in this case, with no possibility for collision.

An infinite X can be quantized by a suitable selection of ϕ functions to achieve a lookup table based on the resulting finite set of disjoint regions, e.g. m–dimensional rectangles. There are

[4] I say "so–called" because I think it is less misleading to view a method for unsupervised learning as a supervised method with a fixed, built–in teacher. Unsupervised methods recode pattern information according to coding principles (such as principal component analysis) that are built into the learning rule.

numerous examples of systems that use this basic approach: Michie and Chambers' [66] "Boxes" system, on which Barto, Sutton, and Anderson [14] based their pole–balancing system; Samuel's [85] "signature tables"; and Feldman and Ballard's [26] "parameter networks." In connectionist terms, each of these ϕ functions represents the function computed by a "value unit" (Feldman and Ballard [26]). If these kinds of ϕ functions are used with nonlinear models, one obtains lookup tables where each entry is more complicated than just a value. For example, a hierarchical signature table of the kind used by Samuel [85], in which lookup tables contain parts of the addresses into other lookup tables, might be thought of as a combination of localized ϕ functions with a particular kind of multilinear model. Raibert's [80] lookup table in which each entry is itself a linear model can be regarded as the result of using this type of ϕ function with a class of bilinear models.

A more general type of lookup table results from using ϕ functions that aggregate the values of underlying variables into overlapping regions instead of disjoint regions. Albus' CMAC [2] is a good example of this approach. Information is stored by spreading each item to be stored over neighboring table entries according to weighting profiles given by the ϕ functions. Alternatively, one can locally store individual items and use a spread function on recall, for example, averaging over neighboring entries as discussed by Atkeson and Reinkensmeyer [6]. In either case, connectionists think in terms of neuron–like units, one for each ϕ function, with overlapping receptive fields and refer to this way of coding information as a type of "coarse coding" [37]. For example, referring to Figure 2 (after Hinton [37]; cf. Waltz and Fu [100]), each point on the x–y plane is encoded as a pattern of activity over many units (i.e., many ϕ functions take non–zero values), and each unit participates in representing many points.

A more traditional method that amounts to an interpolating lookup table is the Method of Potential Functions of Aizerman, Braverman, and Rozonoer [1] and discussed by Duda and Hart [23] and Niranjan and Fallside [76]. In connectionist terms, this method makes use of units each having a possibly global receptive field with a sensitivity curve that changes as a function of the distance from a particular point in the input space (cf. the decrease of electrical potential with distance from a charged particle). Methods relying on representations similar to this are gaining notice as researchers seek learning methods that are faster than back–propagation (e.g., Moody and Darken [71], Niranjan and Fallside [76]). The use of ϕ functions with overlapping receptive fields clearly provides a kind of interpolation and extrapolation that is absent from a simple lookup table.

Other choices of ϕ functions lead to more "algebraic" representations. For example, if $X = \Re^2$, one might select the following five ϕ functions: $\phi_1(x,y) = x$, $\phi_2(x,y) = y$, $\phi_3(x,y) = x^2$, $\phi_4(x,y) = y^2$, and $\phi_5(x,y) = xy$. Then with a linear model and a parameter estimation method suitable for a linear model (such as the LMS method), one can learn quadratic functions of X. Networks employing these kinds of nonlinear pre–processing functions have been called "higher–order networks" by Giles and Maxwell [32]. Although it is clear that extending this approach to high–order polynomials encounters a combinatorial explosion,[5] considerable improvement over linear models using linear ϕ functions may be possible with a relatively small investment in nonlinear pre–processing. For example, exclusive–or, and many other functions exhibiting the same kind of context sensitivity (where the signals on some input pathways radically alter how

[5]As David Eliot pointed out in the workshop discussion, Ivakhnenko's Group Method of Data Handling [45] is a method that helps avoid the combinatorial explosion by pruning each set of n^{th}–degree terms before using the remaining terms to generate the $(n+1)^{\text{st}}$–degree terms. Although this is an off–line method, Anderson [3] relates it to algorithms for learning in layered networks.

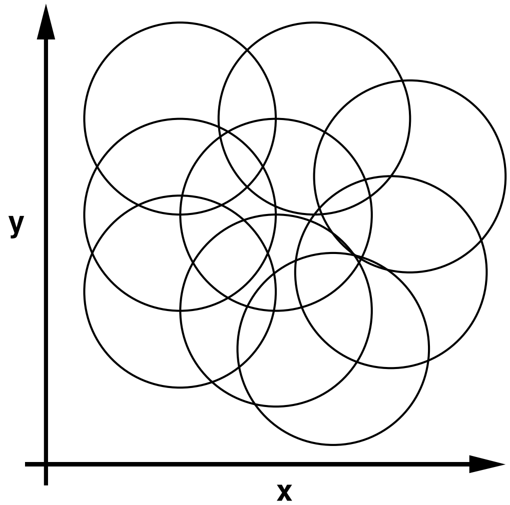

Figure 2: An example of coarse coding: Overlapping receptive fields of ϕ functions determining a representation for functions of $\Re^2$.

the information on other input pathways is processed), are essentially quadratic functions that can be learned quickly using second–order networks. Alternatively, the algebraic structure of ϕ functions can be determined from knowledge about the function to be learned, an approach illustrated by the use by Kawato, Furukawa, and Suzuki [52] of ϕ functions derived from the algebraic form of the dynamical equations of the robot arm to be controlled.

One characteristic of many connectionist applications of parameter estimation that differs from more orthodox applications is that more freedom is exercised in selecting ϕ functions. For example, in conventional system identification for adaptive control, strong assumptions are made about the nature of the system to be controlled, and ϕ functions are often identity functions of basic system variables and their delayed values. Connectionist researchers, on the other hand, often apply parameter estimation methods under weak assumptions about problems, and they often use expansive representations of very high dimension based on a multitude of nonlinear ϕ functions. Although sometimes representations of this kind are motivated by a desire to model the kinds of coding principles that might be employed in nervous systems (coding by the granular layer of the cerebellum being a good example, as described by Albus [2]), in other instances the ill–structured nature of the problem calls for a flexible representation. This kind of representational freedom is one of the most salient characteristics of connectionist research.

5 Nonlinear Models

Another salient characteristic of connectionist research is its use of novel classes of nonlinear models. The focus of the preceding section was the use of nonlinear ϕ functions with linear models, a situation in which one can apply well–behaved estimation methods for linear models. In this case, because the ϕ functions are not parameterized, they do not complicate the computation of the gradient of the model error as a function of the parameters. For example, with a linear model, the mean–square error is a quadratic function of the parameters with a unique minimum no matter what fixed representation is used. Nonlinear models, on the other hand, can generate error surfaces with many local minima so that the final parameter estimates strongly depend on initial estimates and the vagaries of training experiences. Despite these complexities, which make it impossible to prove global convergence results analogous to those existing for linear models, the on–line estimation of parameters for nonlinear models is being taken up by connectionists with great enthusiasm.

Largely responsible for this enthusiasm is the back–propagation algorithm independently developed by le Cun [60], Parker [77], Rumelhart, Hinton, and Williams [83], and Werbos [102].[6] This algorithm is based on the class of nonlinear models consisting of layered networks of units that apply a differentiable nonlinear "squashing function" to the weighted sum of their inputs. Due to the layered structure of these networks, this class of models consists of functions that can be constructed by recursively composing "squashed" weighted sums with other compositions of squashed weighted sums, etc. Extending gradient–descent analysis to this class of models yields the parameter estimation method known as error back–propagation.

Exactly what advantages this class of models has over others is not yet clear, but a number of attractive properties are evident. First, because such models correspond to networks, it is clearly feasible to construct fast, parallel devices to implement these models for real–time applications. Second, the gradient calculations required to update parameter estimates can be executed efficiently via the recursive back–propagation method, which can itself be implemented in parallel hardware. Third, the class of models is universal in the sense that essentially *any* function can be implemented to any desired degree of accuracy by a sufficiently large network (Hornik, Stinchcombe, and White [44]). However, on the negative side, there is no guarantee that the parameter estimates converge to globally–optimal parameters; convergence of any kind can take a considerable amount of training; and, as I discuss in more detail in the next section, there is no *a priori* reason to assume that the kind of generalization produced by this method is appropriate for a given problem.

Despite the current popularity of the back–propagation method, it is a mistake to identify the field of connectionism with the use of this method, as sometimes seems to be done. It is a technique whose capabilities are currently being explored, but it is neither the linchpin of connectionist research nor the only technique being studied for nonlinear parameter estimation. However, aspects of how back–propagation is being used are characteristic of connectionist research. Its utility is being explored in a highly experimental fashion: One tries several different network architectures to see which produces the best performance; one tinkers with the networks; etc. Less emphasis is placed on the detailed justification of a specific class of nonlinear models than seems characteristic of more traditional applications of nonlinear parameter estimation. It would be unorthodox, I think, for an engineer to use a parameter estimation method for a class of models that is rather arbitrary in the context of a specific problem, that has thousands of parameters, and for which there is no guarantee of convergence to a solution with known properties.

It is easy to find fault with this use of parameter estimation, which is undisciplined from the perspective of mathematical and engineering orthodoxy, and one might argue, in response, that this free–wheeling nature of current connectionist research will disappear as the field matures. But I think the experimental nature of this kind of research is desirable—as long as its limitations are properly understood. A field can mature in many ways, one of which is to focus its attention on more and more restricted classes of problems. One of the reasons that connectionist research

[6] Le Cun [61] writes that "Among all the supervised learning algorithms, back propagation is probably the most wi(l)dely used."

is generating so much activity is that it has made it acceptable, at least among its adherents, to think about applying certain kinds of methods to wider classes of problems than is acceptable according to engineering orthodoxy. However, the negative side of this is that there are, in fact, good reasons—theoretical and practical—for the more circumspect nature of orthodox engineering methodology, and connectionists need to be aware of these reasons.

6 Parametric and Structural Learning

The parameter estimation methods discussed above have been called *parametric learning* methods. They are based on assuming that the desired model is a member of a specific parameterized class of models. The term *structural learning* refers to the process of determining what class of models is appropriate for a particular set of applications. In applying error back–propagation, for example, structural learning involves how many layers, how many hidden units, and what kinds of constraints should be used for a particular problem. In practice, the researcher tends to be an essential part of the structural–learning loop as he or she experimentally searches for a network having enough, but not too many, hidden units and the right kinds of constraints among the weights. More generally, the structural–learning loop would address the question of whether or not to use a network at all.

It is tempting to regard structural learning as a special case of parametric learning. This is a view Anderson and I took in a 1985 paper [12]:

> Since one can always regard structures as being parameterized, so that adjusting structures amounts to adjusting more parameters, this distinction [between parametric and structural learning] is not completely straightforward. However, what we mean by structural learning generally involves a space of parameters that is so large, and a performance evaluation surface that is so complex, that the usual algorithms for parametric adaptation do not work....One can view the adjustment of a connection weight in a complex network as a structural adjustment since it affects the roles of other weights in generating network behavior (Barto and Anderson [12]).

It is indeed true that a class of models with a very large number of parameters can include within it many different kinds of structures. However, estimating such a large number of parameters does not automatically qualify as a structural learning technique because there is no guarantee that a structure that is best in any sense, or even good, will be selected by the parameter estimation process, nor that generalization, i.e., extrapolation to novel instances, will be effective. Adjusting parameters to maximize a performance criterion defined only in terms of matching the training data has no automatic implications for selecting suitable structures.

There are several approaches that address these shortcomings which I mention below, but for the moment consider using an unadorned parameter estimation method to estimate a large number of parameters from training examples, such as error back–propagation applied to a large network. The problem has recently been discussed by Denker et al. [21], Solla [89], and Baum and Haussler [17], and is a classical one in pattern classification. Duda and Hart [23] state it succinctly as follows: "In general, reliable interpolation or extrapolation cannot be obtained unless the solution is overdetermined." As the number of parameters increases, the amount of data necessary to reliably estimate them increases. Small mean–square error on the training set does not imply that generalization will be successful. This is exactly analogous to what one observes in some applications of back–propagation, as pointed out by Denker et al. [21]. We cannot say that the method necessarily discovers the "rule" by which the data set was generated. Reliable generalization requires that either the number of training examples is many times larger than the number of parameters,[7] or the class of parameterized models is already matched in some way to the generation rule.

There is a trade–off between the generality of a class of models and the adequacy of the kind of generalization it will produce. This has been analysed in terms of the entropy of a class of models by Denker et al. [21] and Solla [89]. I called this the generality/generalization tradeoff [9]: It is *because* of representational restrictions built into a class of models that correct generalization can occur if the class of models happens to be appropriate to a task. To the extent that a class of models can represent any structure, it cannot be expected to produce meaningful extrapolations beyond the data on which it is trained. Artificial intelligence researchers studying machine learning speak of the representational *bias* of an induction technique. One does not expect an unbiased form of representation to produce meaningful generalizations. This is the reason that using parameter estimation for very general classes of models, that is, classes of models lacking a high degree of bias, is more closely related to storing information into a memory structure than it is to identifying the structure of the process generating the data. That the class of layered networks trained by back–propagation is a "universally approximating" class is not, in itself, a particularly useful property.

Although even large connectionist networks are not without bias,[8] additional bias can be introduced into a connectionist learning method in several ways in order to facilitate the appropriate kind of extrapolation. In addition to introducing bias by the choice of the ϕ functions, one can enforce constraints among the parameter values. For example, one can restrict parameter changes so that each model estimate has certain symmetries, such as translational invariance. Numerous connectionist researchers have illustrated the utility of this method of biasing learning (see, for example, Denker et al. [21], Giles and Maxwell [32], Rumelhart, Hinton, and Williams [83], and Solla [89]). Another method is to alter the performance criterion by adding conditions other than a measure of model error on the training instances, a point made by Sanderson in the workshop discussion. Rumelhart (in preparation), for example, has experimented with back–propagation with additional conditions designed to force a network to favor representations utilizing few units. This is a method for introducing a form of bias while attempting to maintain representational universality. A related method was presented by Mozer and Smolensky [72] in which networks are pruned through the use of a relevance measure. Other methods for controlling the bias of network learning involve a "developmental" approach in which a small, and thereby highly biased, network is trained to some criterion, its weights are frozen, and it is then embedded into a larger network as training continues (e.g., Gallant [31], Honavar and Uhr [43]). This process continues until a desired level of performance is achieved. Incrementally expanding the network during learning in this manner may result in properties analogous to those obtained by representing a function as a linear combination of orthogonal functions as in Fourier analysis: The expansion can be continued as far as necessary, and the most accurate simple models form parts of more complex models (of course, there remains the question of how to bias each stage of the developmental process).

[7]Baum and Haussler [17] provide a theoretical analysis of the relationship between the number of weights, the number of training examples, and the accuracy of generalization for layered networks of linear threshold units.

[8]Back-propagation experts emphasize that the number of degrees of freedom in a network does not necessarily equal the number of adjustable weights. For example, a network with m input lines and n output lines, consisting of linear units (i.e., no squashing functions) has mn degrees of freedom no matter how many adjustable weights there are. This is true because any such network is equivalent to a one–layer network with mn weights. Something analogous to this can happen in nonlinear networks.

Finally, it is important not to overlook the fact that just representing models in the form of networks of interconnected components offers some possibilities for introducing bias that are absent in some other classes of models. Knowledge of the problem domain may dictate that certain variables should influence certain other variables, and that connections between other variables should be absent. In other words, selecting the architecture of a network can be a good way to specify a class of models if the selection is based on some understanding of the relationship to be learned. Hinton's [38] family tree network is a good example of an application of back–propagation to a judiciously designed layered network that genuinely extracts structure from the training process. The key point is that such rule extraction is not an automatic consequence of applying a connectionist network; it is rather a consequence of introducing the right kind of bias into the network on the basis of *a priori* knowledge. Consequently, one of the attractive features of the connectionist approach to nonlinear parameter estimation is that it facilitates the introduction of certain kinds of bias.

7 Memory versus Models

I suggested above that the idea of an associative memory network provides a perspective on parameter estimation that is a distinct contribution of the connectionist approach. Given the discussion above of representations, nonlinear models, and parametric–versus–structural learning, it is possible to comment more concretely on the relationship between parameter estimation for forming models, or rules, and for implementing memory structures. It was pointed out in the section "Representation" that a lookup table, which perhaps is the prototype memory structure, can be viewed as a special case of parameter estimation applied to a linear model with a particular expansive representation (Equation 9). The parameters are the contents of the memory locations, and the estimation process just amounts to storing information in specific locations. In the simplest case of noise–free training data, the learning rate, or gain, of the estimation method can be set so as to accomplish one–trial learning because there is never any interference between the stored items. Admittedly, this is a trivial example of parameter estimation, but it is parameter estimation nonetheless. However, in this case, it is not appropriate to regard the estimation process as identifying a system or rule. The parameterized class of models is so general–purpose that it does not bias the inductive process at all, and no generalization occurs. Moreover, assuming noise–free data, the one–to–one relationship between the data used to determine the "model" and the parameters is clear: each possible input pattern has to be seen once to store the entire function.

The point is not that the parameter estimation framework provides new insight about lookup tables, but rather that lookup tables represent one extreme of a continuum of possible applications of parameter estimation—the other endpoint being the estimation of the relatively few parameters specifying a model with a high degree of specialized structure. One extreme is clearly characterized as pure memory storage; the other as model–based rule identification. The criteria for evaluating the theoretical soundness of applications of parameter estimation are markedly different for the two ends of this continuum: In the case of pure memory, the relationship between data and parameters is relatively simple, and "learning" should be fast; in the case of model–based rule identification, on the other hand, slower learning is acceptable, but the utility of extrapolation and interpolation critically depends on the bias in the class of models and the degree to which parameter values are constrained by the training data. Because connectionist uses of parameter estimation range over this entire spectrum from memory–based to model–based applications—with the most controversial applications falling in the intermediate range—it is not clear whether

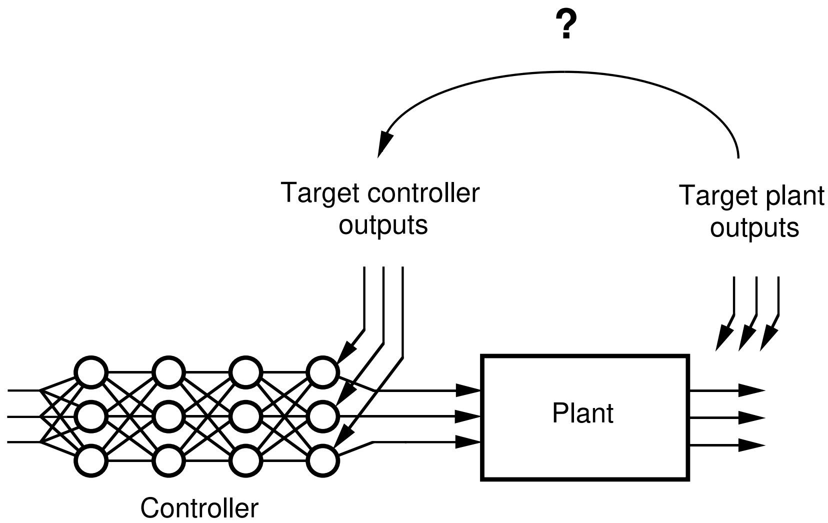

Figure 3: The problem of obtaining training information in a control problem: Target plant outputs may be known but not the control signals that produce them.

it is appropriate to evaluate network performance as memory storage or as model–based rule identification. This exposure of connectionist applications to both sets of criteria accounts to some degree for the controversial nature of connectionist methods. However, I believe that one of the strengths of connectionist applications of parameter estimation is that they do, in fact, freely range over this ambiguous region of the memory/model continuum. Suitably designed adaptive networks can achieve a balance between the relevant tradeoffs that is not possible for methods at either extreme of this continuum.

8 Where Does the Training Information Come From?

The parameter estimation methods that I have discussed all require a training procedure in which desired, or target, outputs of the decision rule, model, etc. are available for a sufficiently varied set of cases; that is, they are methods for supervised learning. Where does this training information come from? Figure 3 outlines the general problem.[9] Suppose one wants a connectionist network to learn to control a plant that is too complex, or about which too little is known, to permit the use of conventional techniques for designing controllers. In a typical control problem, one may have target plant outputs but not target network outputs, which in Figure 3 are the control signals. How can training signals be provided to the network unless some agency already knows a great deal about how to control the plant?

Notice that a difficulty analogous to this is present in the task of training the hidden units of a layered network when target responses are known only for the visible units (Figure 4). For any hidden unit, the portion of the network interposed between that hidden unit and the output units,

[9]Figure 3 and those that follow show layered networks, but these should be interpreted merely as icons representing any method for learning complex functions via supervised learning, including the storing of instances in a lookup table.

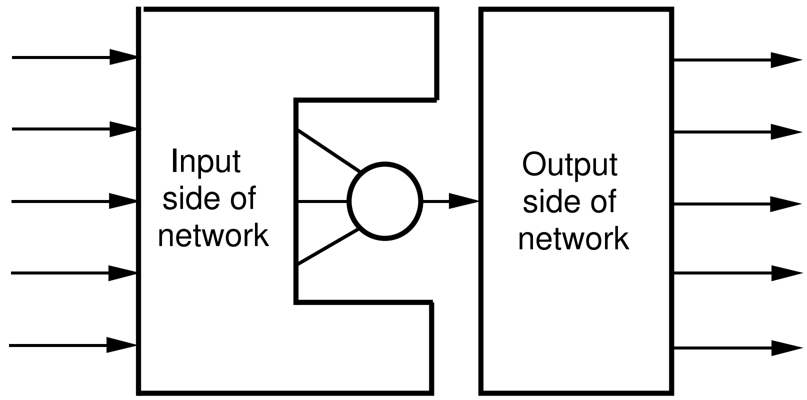

Figure 4: The problem of training hidden units in a network is similar to the problem of training a controller: Target outputs are not directly available for a hidden unit.

i.e., the portion of the network that determines how the activity of the given hidden unit influences the output units, is analogous to the plant in Figure 3. The problem of determining target responses for the hidden units is therefore analogous to the problem of training a controller on the basis of target plant behavior, and methods that apply in one case, such as back–propagation and reinforcement learning, also apply to the other (see refs. [7, 8, 12, 13]).

In the remainder of this section, I describe in highly schematic form the basic ways in which the kind of training information required for supervised learning can be obtained in tasks relevant to control. In the section "Reinforcement Learning," I discuss methods that do not require training signals as informative as those required by supervised learning methods. No attempt is made to describe in detail the types of plants and control problems with which control theorists are concerned, nor is justice done to the highly developed methods that exist for prediction, system identification, and adaptive control. Thinking of the plant simply as an input–output mapping and the controller as a feedforward controller suffices for most of the observations I make, but these observations, suitably modified, also apply to more complex types of plants and can be integrated with feedback control methods. It is not my intention to suggest that connectionist networks are suitable only for feedforward control, or that networks must be used in the absence of more conventional feedback control loops.

Copying an Existing Controller—If there exists a controller capable of controlling the plant, then the information required to train a connectionist network can be obtained from this controller as shown in Figure 5. The target network output for a given input is the output of the existing controller for that input. The network learns to copy the existing controller. Widrow and Smith [111] applied this method to a version of the pole–balancing problem, and Widrow refers to this as a method for constructing an expert system by acquiring knowledge from an existing expert. One might question the utility of this method on the grounds that if there already exists an effective controller, why would it be useful to have another one in the form of a network? Two answers are apparent: first, the existing controller may be a device that is impractical to use (such as a person); second, the adaptive network may be able to form an effective control rule on the basis

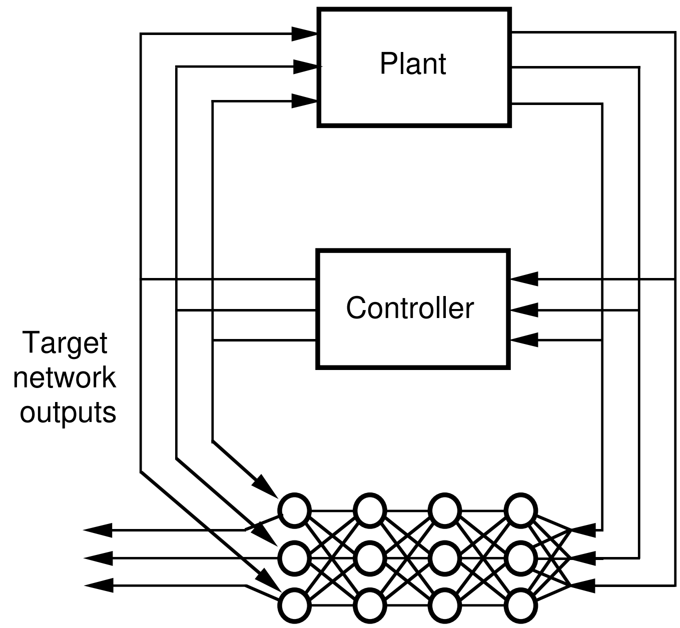

Figure 5: Copying an existing controller with a connectionist network.

of a representation of the system state that is easier to measure than the representation required by the existing controller (Widrow and Smith [111]).

Adaptive Prediction—Figure 6 illustrates the basic idea in training a network to be a predictor. Assume that the lines across the top of the figure carry signals whose values one wishes to predict. For simplicity, also assume that these same signals provide the information from which the predictions will be made. For training purposes, the input to the network consists of delayed values of the signals, and the target output consists of the current values of the signals. The network therefore tries to match the current signal values by adjusting a function of their past values. Then when the input to the network bypasses the delay units, the output of the network is a prediction of the values the signals will have in the future. Assuming each delay unit shown in Figure 6 delays its input for τ time steps, and that the network requires a negligible amount of time to compute its output from its input, then the trained network provides estimates for the values of the signals τ steps in the future. This approach to adaptive prediction rests on the assumption of a parameterized class of models for the functional relationship between the current and past values of the signals and their later values, or equivalently between earlier values of the signals and their current values. A parameter estimation method is applied with training information supplied by observations of the signal values over time.

The scheme illustrated in Figure 6 is only the simplest special case of more general prediction schemes where cascades of delay units (tapped delay lines) are used to provide input to the model, or where recursive, or auto–regressive, models are used in which past values of model outputs form part of the input (see, for example, Goodwin and Sin [33], Widrow and Stearns [112]). Lapedes and Farber [59] provide an example of nonlinear connectionist prediction.

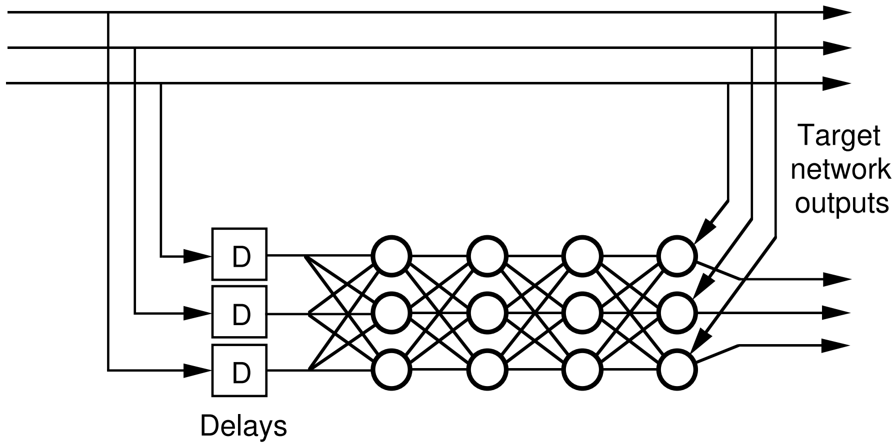

Figure 6: Using a connectionist network for adaptive prediction.

System Identification—Training information can be obtained by observing the input–output behavior of a plant, as suggested in Figure 7, where the network receives the same input as the plant, and the plant output is the target network output. The figure shows only the case where the model is a mapping from plant inputs to outputs, but more complex types of models are commonly employed. For example, the input to the model may consist of various delayed values of plant inputs (so that adaptive prediction as discussed above is a special case of system identification), and the model may be a recursive model (see, for example, Goodwin and Sin [33], Ljung and Söderstrom, [62], Widrow and Stearns [112]).

System identification is an essential part of some methods for adaptive control. In one approach to adaptive control, one selects a class of models for identification purposes such that each model in the class is a system for which one can design an effective controller using proven design methods for known plants. One can therefore express a control law in terms of the parameter estimates produced by the system identification procedure. This will be an effective control law if the plant can be adequately modeled within the assumed class of models. Complexities exist that have been studied extensively by adaptive control theorists, some arising when the control law is adjusted on–line at the same time the model parameters are being adjusted.

Using a nonlinear connectionist network instead of a more conventional system identification procedure does not immediately extend the applicability of adaptive control to the systems that nonlinear networks are able to identify. That a system can be identified does not imply that it can be controlled. It would be necessary to have controller design techniques that apply to the classes of nonlinear systems identifiable by networks. It may be possible to develop control theory in this direction, but because networks can represent essentially *any* system, there is no compelling reason to hope that the invocation of connectionist ideas is likely, by itself, to shed any light on the theory of nonlinear control. However, the method I describe below under the heading of "Differentiating a Model" does illustrate the utility of having a model of a nonlinear plant in the form of a network, but this model is used differently than models are used in the adaptive control methods just described.

Identification of System Inverse—Figure 8, Panel A, shows how an adaptive network can be used to identify the inverse of a plant. In contrast to the situation shown in Figure 7, here the input

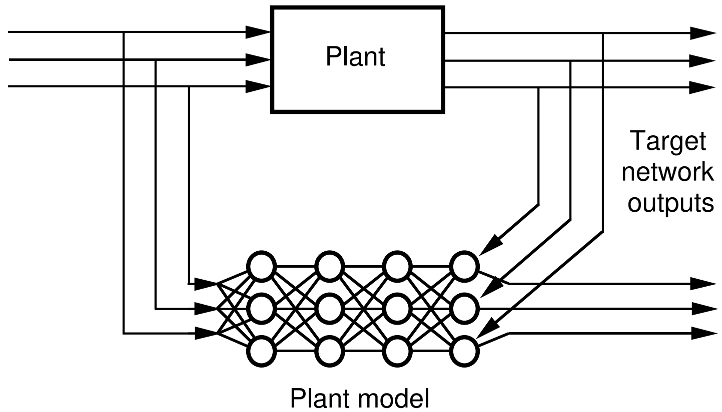

Figure 7: Using a connectionist network for system identification.

to the network is the output of the plant, and the target output of the network is the plant input. If the network can be trained to match these targets, it will implement a mapping that is a plant inverse. Once one has such an inverse, it can be used for control purposes as shown in Figure 8, Panel B. The desired plant output is provided as input to the network; the resulting network output is then used as input to the plant. If the network is a plant inverse, then this plant input causes this desired plant output (obviously, this simple description overlooks many subleties of a real control problem).

Widrow, McCool, and Medoff [110] proposed this method for the adaptive control of linear systems (also see Widrow [107] and Widrow and Stearns [112]), and it can be applied to the control of nonlinear systems as illustrated by Psaltis, Sideris, and Yamamura [79] and Srinivasan, Barto, and Ydstie [90], who use back–propagation to identify a nonlinear plant inverse. This basic method has also been used in models of motor control and robot control systems for learning inverse kinematics (e.g., Grossberg and Kuperstein [34], Kuperstein [58]) and inverse dynamics (e.g., Kawato, Furukawa, and Suzuki [52], Kawato, Uno, Isobe and Suzuki [53], and Miller [67]). The method of Kawato et al. [52], which they call feedback–error learning, differs from the scheme shown in Figure 8 in that the input to the network is the *desired* plant output instead of its actual output, and a feedback controller is used during training.

A major problem with inverse identification arises when many plant inputs produce the same output, i.e., when the plant's inverse is not well–defined. In this case, the network will attempt to map the same network input to many different target responses. Most parameter–estimation methods tend to average over the various targets and thereby produce a mapping that is not necessarily an inverse (as pointed out by Jordan [46]). Nevertheless, the use of nonlinear networks for identifying nonlinear plant inverses is a promising avenue for future research because an inverse model of a nonlinear plant, unlike a forward model, has immediate utility for control.

Differentiating a Model—This method for training a controller relies more on back–propagation than on general network methods, but it is a way of using networks that may have promise for adaptive control. This technique was described by Werbos [104, 105] and Jordan and Rumelhart [47], and variants of it have been applied to problems in motor control by Jordan [46], Kawato [51], and Widrow [106]. The method is illustrated in Figure 9. The back–propagation algorithm

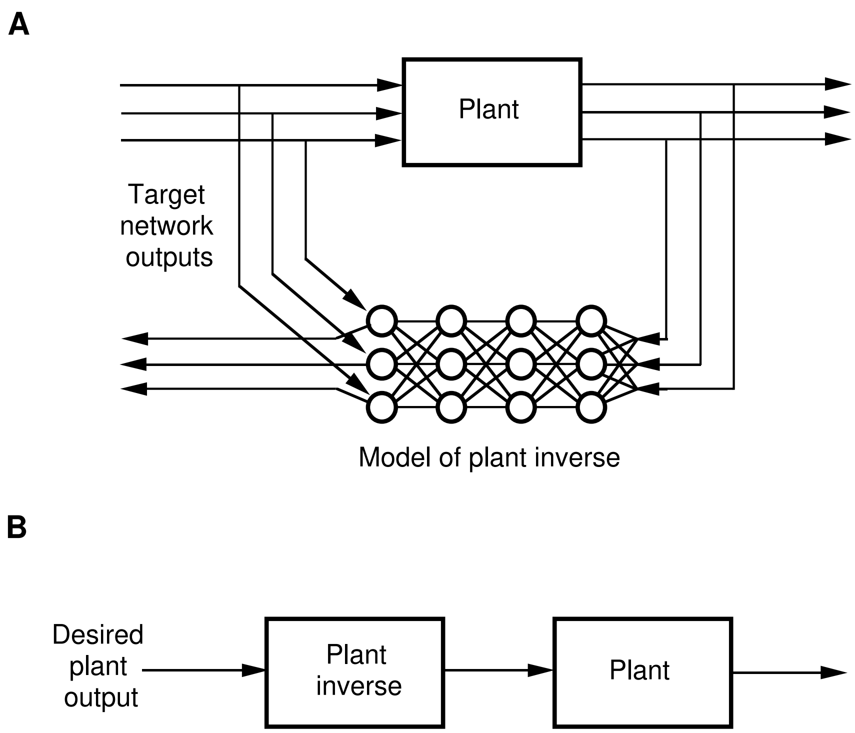

Figure 8: Using a connectionist network for identifying a system inverse.

is used to identify the plant as shown above in Figure 7 (the training signals for this identification stage are not shown in Figure 9), resulting in a forward model of the plant in the form of a layered network. The utility of a forward model having this form is that one can efficiently compute the derivative of the model's output with respect to its input by means of the back–propagation process, which as Jordan [46] points out, evaluates the transpose of the network Jacobian at the network's current input vector. Consequently, propagating errors between actual and desired plant outputs back through the forward model produces the error in the control signal, which can be used to train another network, or other kind of supervised learning system, to be a controller. In Figure 9 this back–propagation process is illustrated by the dashed line passing back through the forward model and continuing back through a second layered network that uses it to learn a control rule. As Jordan [46] points out, this method has advantages over the direct identification of a plant inverse when a plant inverse is not well–defined. Of course, to apply this basic idea one only needs a model in a form that can be differentiated; it need not be a layered network. However, using a layered network that can be trained by back–propagation is a nice idea because the back–propagation mechanism can be used both to adjust the network's parameters and to differentiate the resulting model.[10]

This method for obtaining training information for an adaptive controller illustrates a general

[10]Note, however, that the error which is back–propagated differs in the parameter–adjustment and model–differentiation stages. In the first case, it is the difference between the network output and the plant output; whereas in the second case, it is the difference between the plant output (or the network output if the network is well–trained) and the desired plant output.

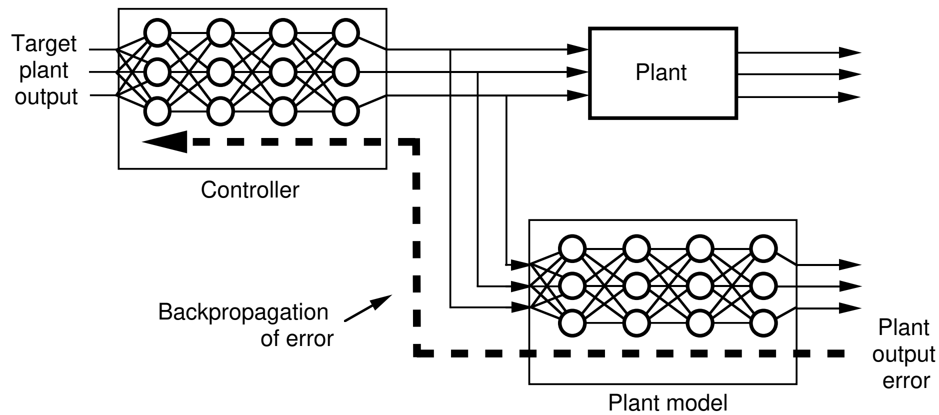

Figure 9: Backpropagating through a forward model of the plant to determine controller errors.

principle that can be exploited in other ways in control problems. This principle is to choose the class of models for system identification partly on the basis of the existence of specialized techniques applicable to the models in the class. For example, in conventional approaches to adaptive control, a class of models is selected such that techniques for controller synthesis can be applied to any system representable within the class (i.e., time–invariant linear systems). When a model is specified by the parameter estimation process, one can design a controller under the assumption that this model is an accurate model of the plant. In the same way, determining a plant model in the form of a layered network with differentiable squashing functions allows one to apply the back–propagation process to estimate how inputs to the plant should be altered to change the plant's output in desired ways.

Le Cun [61] has pointed out that the back–propagation method can be derived using a Lagrangian formulation like the one used in optimal control. It turns out that algorithms similar to back–propagation are used in optimal control—although not for adjusting the parameters of a system model. It is interesting that the method described here for using back–propagation to differentiate a model for control purposes is more closely related to optimal control methods than is its application to parameter estimation for system identification (see, for example, Bryson and Ho [19]).

9 Reinforcement Learning

The preceding section focuses on how the training information required for supervised learning can be obtained in tasks relevant to control. It is clear that there are many possibilities. However, some learning tasks arising in control require methods that are not accurately characterized as supervised learning methods. Suppose one wishes to adjust a control rule in order to improve the performance of a plant as measured by a performance measure that in some way evaluates the overall behavior of the plant; for example, one might wish to maximize a measure of the plant's energy efficiency over time. Optimal control methods apply if there are models of the plant and performance measure that are sufficiently accurate and tractable. However, in less structured situations it may be possible to improve plant performance over time by means of on–line learning methods performing what we call *reinforcement learning*, after Mendel and McLaren [65], who

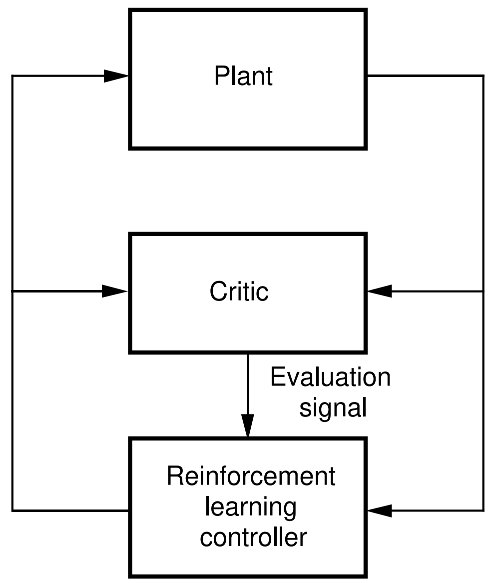

Figure 10: A reinforcement learning control system.

discuss this approach to control. Whereas the performance measure for a supervised system is defined in terms of a set of targets by means of a known error criterion (e.g., mean–square error), reinforcement learning addresses the problem of improving performance as evaluated by *any* measure whose values can be supplied to the learning system. Consequently, in tasks of this kind there exist desired control signals—namely those that lead to optimal plant performance—but the learning system is not told what they are because there is no agency knowledgeable enough to act as this kind of teacher. The problem is to find these optimal control signals, not simply to remember and generalize from them.

Viewed this way, reinforcement learning involves many of the issues discussed above involved in obtaining the training information required for supervised learning when it is not directly available (Figure 3). In the case of reinforcement learning, however, the situation is more general than that shown in Figure 3 in that instead of trying to determine target controller outputs from target plant responses, one trys to determine target controller outputs, or desired changes in controller outputs, that would lead to increases in a measure of plant performance—a measure not necessarily defined in terms of target plant responses. This is illustrated in Figure 10 which shows a reinforcement learning controller interacting with a plant and receiving an "evaluation signal" generated by a "critic" capable of evaluating plant performance.

Reinforcement learning essentially involves two problems. The first problem is to construct a critic capable of evaluating plant performance in a way that is both appropriate to the actual control objective and informative enough to allow learning. The second problem is to determine how to alter controller outputs to improve performance as measured by the critic. Assuming the first problem is solved—a highly nontrivial assumption—the second problem, discussed next, is relatively straightforward given what has already been said about obtaining training information for supervised learning.

9.1 Improving Performance

If one assumes that the critic is capable of providing an immediate evaluation of plant performance that is appropriate to the actual control objective, then it is necessary to determine the gradient of the critic's evaluation as a function of control signals. One approach is to adapt the method "Differentiating a Model" described above (Figure 9). The idea is to learn a model—in a form that can be differentiated easily, such as a layered network—of the process by which control signals lead to evaluations. Referring to Figure 10, this would be a model of the plant together with the critic. Then, the gradient of evaluation as a function of the controller's actions can be estimated by differentiating the output of this model with respect to its input, for example, by backpropagating through the network representing it, thus producing the information required to adjust the control rule. This approach to reinforcement learning has been suggested by Werbos [104, 105], and the case where there is a critic but no explicit plant model has been studied by Munro [73] and Williams [115].

A more direct approach to adjusting control actions in order to improve plant performance is to actively explore the space of controller outputs by modifying control actions and observing how the performance evaluation changes as a result. Beneficial changes in control actions are incorporated into the control rule. Learning in this way is related to the trial–and–error learning studied by psychologists in which behavior is selected according to its consequences in producing reinforcement—hence the term reinforcement learning. Active exploration is necessary when the evaluation signal is available but not its gradient as a function of control actions. In supervised learning, the appropriate gradient is directly available (at least for each training example) because the performance measure has a known form (e.g., mean–square error) whose gradient can be precomputed and expressed in terms of target responses. Similarly, the method for reinforcement learning described above, where a model of the evaluation process is constructed, provides a means for directly providing gradient estimates without the need for active exploration, except for the exploration needed to construct the model.

Direct methods for reinforcement learning, on other hand, perform gradient ascent (or descent) on an evaluation surface whose values (usually scalars) are directly available to the learning system as evaluation signals, but whose gradient as a function of control actions is not. This gradient has to be estimated by comparing the evaluations obtained during exploration and relating these changes to the actions taken, or to the action changes made, during exploration. One way of understanding the distinction between gradient descent on the basis of receiving evaluation signals instead of gradient vectors is to contemplate the difference between the sign of a scalar evaluation signal and the sign of an error signal. A negative error means that the scalar action was too high (assuming error = target − actual), whereas the sign of a negative evaluation intrinsically means nothing (perhaps -10 is the best evaluation possible). One signal is a derivative with respect to an action; the other is not. Confusing this view, however, is the situation that arises when performance is evaluated with respect to some performance standard. In this case, a negative evaluation may mean that system performance is not as good as could be expected based on past experience with similar situations. In this case, the signal is a kind of derivative, but it is not a derivative with respect to actions, and its being negative does not in itself indicate how the action should be altered (indeed, the action will generally be a vector). A single evaluation of an action does not contain information about how to beneficially change that action. There are, however, many ways to estimate the required gradient information on the basis of evaluation signals, each of which produces a different type of reinforcement learning algorithm.

A direct approach to reinforcement learning that is highly developed is the theory of learning automata. This theory originated in the Soviet Union with the work of Tsetlin [99] and the independent research in the West of mathematical psychologists (for example, Estes [24] and Bush and Mosteller [20]). It has been extensively developed since then within engineering (Narendra and Thathachar [74, 75]). A class of reinforcement learning algorithms known as *stochastic learning automata* has been widely studied. These algorithms probabilistically select actions from a countable set of possible actions and update action probabilities on the basis of evaluative feedback. They are applicable when the evaluation process is stochastic and the task is to maximize the expected value of the evaluation received. Although stochastic learning automata have been studied mostly in *nonassociative* form, that is, where they search for a single optimal action, they can be extended to learn mappings, such as control rules, that associate input patterns with actions. It is straightforward to combine nonassociative learning automata with lookup–table representations of mappings. Each table entry effectively has its own nonassociative learning automaton whose job is to discover—without being told—what is best to store there, that is, what control action is best when that entry is accessed. This approach is illustrated by the pole–balancing system of Barto, Sutton, and Anderson [14] and Franklin's [27, 28] method for learning the feedforward component of a control signal to cancel unmodeled disturbances. Another way to use learning automata to learn mappings is illustrated by the SNARC ("Stochastic Neural–Analog Reinforcement Calculator") system of Minsky [68], which used a nonassociative learning automaton to control the transmission probability of each link in a network.

It is also possible to combine stochastic learning automata with the framework of parameter estimation by parameterizing mappings from pattern input to action probabilities. As these parameters are adjusted under the influence of evaluative feedback, action probabilities are adjusted to increase expected evaluation. Farley and Clark [25] experimented with such a method taking the form of a neuron–like unit, and Barto, Sutton, and Brouwer [15] studied a related method in the context of associative memory networks, following the neural hypothesis of Klopf [54, 55] that neurons are attempting to extremize a physiological quantity. Barto, Sutton, and Brouwer [15] called their network an "Associative Search Network" because it actively searched for the optimal output pattern to associate with each input pattern. Sutton [92], Anderson [3, 4], and Gullapalli [35] have studied related methods and have combined stochastic reinforcement learning with the back–propagation method. Barto and Anandan [11] presented an algorithm of this type, called the *associative reward–penalty*, or A_{R-P} , algorithm and proved a convergence theorem. The A_{R-P} algorithm has been used as the basis for learning in layered networks (see refs. [3, 7, 8, 10, 12, 13]), an approach analysed and extended by Williams [113, 114]. The A_{R-P} algorithm is closely related to an earlier method for parametric reinforcement learning described by Widrow, Gupta, and Maitra [108] which they called the "selective bootstrap algorithm."

9.2 Critics

I now turn to the problem of constructing a critic capable of evaluating plant performance in a way that is both appropriate to the actual control objective and informative enough to allow efficient learning. If the objective is relatively straightforward, such as maintaining plant output near a reference level, then it is not difficult to provide useful performance feedback continuously over time in the form of error signals, and it would not be necessary to resort to reinforcement learning. On the other hand, if the objective is to achieve a terminal state with desired properties, or a trajectory that maximizes a measure of process yield, efficiency, etc., then the problem of providing useful performance information continuously over time is much more difficult. How is *current* performance to be evaluated if the objective concerns properties of *future* trajectories? Although the theory of optimal control concerns performance criteria of this kind, optimal control techniques require an accurate plant model and either an enormous amount of computation or sufficient mathematical tractability of the plant model and objective function to permit analytical solution. An on–line adaptive approach calls for a *subgoal performance measure* (e.g., Mendel and McLaren [65], Saridis and Gilbert [86], and Waltz and Fu [100]) that is consistent with the actual control objective but more easily accessible, where consistency means that the control objective can be achieved by the moment–to–moment maximization of the subgoal performance measure. The idea of a critic, specifically, the idea of an *adaptive critic*, represents an adaptive approach to optimal control by means of an on–line procedure for learning a subgoal performance measure that is consistent with control objectives.

The perspective most familiar to me comes from the early AI literature. Minsky [69] discusses the basic problem as one of *credit assignment* for reinforcement learning systems. How does one correctly assign credit or blame to an action when its consequences unfold over time and interact with the consequences of other actions? Samuel [84, 85] described a program capable of learning how to play a respectable game of checkers that addressed this problem. Among the many learning methods used in the program was one that adjusted a parameterized evaluation function so that the value it gave to a current board position came to reflect the utility of board positions that were likely to arise later in the game. Using this method, it was possible to assign credit to moves that were instrumental in setting the stage for later moves that directly captured opponent pieces.

In his PhD dissertation, Sutton [92] developed an algorithm, calling it the "Adaptive Heuristic Critic" algorithm, that is closely related to Samuel's method but extended, improved, and abstracted away from the domain of game playing. This work, which began with Sutton's earlier interest in classical conditioning and our exploration of Klopf's [54, 55] idea of "generalized reinforcement," led to the use of the algorithm in the reinforcement learning pole balancer of Barto, Sutton, and Anderson [14], where it was incorporated into a connectionist unit called the "Adaptive Critic Element," or ACE. This system was studied further by Selfridge, Sutton, and Barto [87]. Anderson [3, 4] continued to explore the method, combining it with error back–propagation in order to learn nonlinear evaluation functions. Since then, Sutton [93] has extended the theory and has proved a number of results for a general class of algorithms he calls "Temporal Difference," or TD, algorithms. Related credit assignment methods have been studied by Gaines and Andreae [30], Holland [42] (the "bucket–brigade" algorithm), Booker [18], and Hampson [36]. Witten [116] discussed a similar method in the context of Markov decision problems. Werbos [103] independently proposed a class of methods that includes TD–like algorithms and related these algorithms to the framework of dynamic programming, calling them "heuristic dynamic programming" methods. A similar connection was made recently by Watkins [101], who uses the term "incremental dynamic programming." The relationship between TD algorithms and dynamic programming is extensively discussed by Barto, Sutton, and Watkins [16].

This connection to dynamic programming provides an essential step in understanding TD algorithms and their relevance to control. Computational methods for optimal control rely on dynamic programming to determine an optimal control rule by determining an ideal subgoal performance measure that encompasses all future consequences of control decisions. However, conventional dynamic programming requires an accurate model of the plant and is computationally intensive. A TD method is an on–line parameter estimation method that can be used as a component of methods for estimating an ideal subgoal performance measure while interacting with the plant. This estimation process does not require a model of the plant, although there are natural ways to use TD methods in conjunction with a plant model (as is the case in Samuel's checker player).

Finally, I want to mention another aspect of TD methods that I find especially interesting. In addition to providing methods for addressing credit assignment problems with strong ties to a large body of mathematical theory, TD methods provide good models of a wide range of animal behavior in classical, or Pavlovian, conditioning experiments. There are a number of conditioning models using TD–type algorithms, including those of Sutton and Barto [95] and Klopf [56]. The more recent conditioning model of Sutton and Barto [94, 96], called the TD model, most closely fits into the framework of dynamic programming. That this simple model accounts for a wide range of animal conditioning behavior suggests that the theoretical principles it embodies are relevant to natural as well as synthetic learning systems.

9.3 The Role of Reinforcement Learning in Control

Reinforcement learning is a very general approach to learning that can be applied when the knowledge required to apply supervised learning is not available. The generality of reinforcement learning is obtained at the cost of efficiency when compared to the more specialized supervised methods *when these methods are applicable.* Consequently, although *any* control task requiring learning can be formulated as a problem of reinforcement learning (where the objective is to maximize a performance measure), it is better to use more specialized methods whenever possible in order to take full advantage of the available knowledge. For example, if the performance measure is an error based on known targets, then a reinforcement learning method can solve the same problem a supervised method would solve, but it would do so without making use of the fact that the performance measure has a specific form; consequently, it would take longer than a more specialized supervised method. For even small problems, these differences in efficiency can be very significant, as illustrated by the simulations by Barto and Jordan [13] in which comparisons are made between using reinforcement learning and error back–propagation to train the hidden units of layered networks (which is a kind of control problem as pointed out above and illustrated in Figure 4).

It is clear, then, that the use of reinforcement learning is most easily justified when there is a genuine lack of the knowledge required for applying more specialized learning methods. However, it is always possible to approach these problems by applying methods designed to acquire the requisite knowledge, either before or during the application of the methods that make use of it. This approach is familiar from adaptive control, where some methods implement a control law whose design is based on a model under the assumption that the model provides a complete and accurate characterization of the plant being controlled. This approach also can be applied to tasks requiring the attainment of a future goal while optimizing plant performance over time: a plant model can be formed and then used as the basis for a conventional dynamic programming approach to optimal control; or perhaps a kind of space/time model in the form of a layered network can be obtained so that the appropriate control signals can be computed by "back–propagating through time" (see Kawato [51]; Widrow's "truck backer–uper" [106] also uses this approach). Although these latter approaches might still be viewed as instances of reinforcement learning because they do not begin with knowledge of moment–to–moment target outputs for either the plant or controller, they do not employ the kind of direct on–line search in control space that is usually associated with reinforcement learning. Is there a role for these more direct reinforcement learning methods in solving control problems?

I think these methods have a definite role because model–based methods are not necessarily effective with inaccurate models, and exploratory behavior is necessary *anyway* to achieve accurate models. It makes sense to take advantage of this exploration to directly modify control rules via reinforcement learning in order to obtain improvements in performance while model parameters are being adjusted. This strategy is likely to be especially useful in tasks involving the optimization over time of a nonlinear plant's performance. Considerable time may be required to form sufficiently accurate models, and there can be sensitive dependencies between a model and the control actions it implies. Moreover, the range of assumptions made in the model–based approach may not be justified to the degree necessary to obtain the desired refinements in performance. Adjusting the control rule by direct appeal to the performance measure, or a consistent subgoal measure, can shorten the chain of assumptions on which a method is dependent. Additionally, in some problems there may not be enough time to take full advantage of an elaborate model, especially if dynamic programming methods are required to generate control actions from it. Reasons such as these, which all involve aspects of the the usual tradeoffs between the acquisition of knowledge and its use for control, suggest that direct adjustment of a control law via reinforcement learning can play a useful role in control, and that when combined with adaptive critic methods, it may be most useful for perfecting plant performance with respect to complex performance measures.

10 Conclusion

By emphasizing the continuity of connectionist learning methods with those from more established traditions, I have highlighted some of the significant features of connectionist research while minimizing discussion of features that I think will prove more trainsient. I have not discussed all the learning methods that connectionists have devised, focusing instead on those that seem immediately relevant to control. I have argued that the most distinctive character of connectionist learning systems lies not so much in their technical specifications as in the methodology with which they are applied. Connectionists have adopted an experimental approach more like that of AI than of the engineering disciplines to which, technically, connectionist methods have the closest ties. This experimental, heuristic approach is characterized by what I termed representational freedom and a willingness to plunge ahead when theoretical guarantees are lacking. It is easy to criticise this free–wheeling nature of much connectionist research, but theoretical guarantees are often obtained at the cost of extremely restrictive assumptions about tasks (e.g., the assumption that a plant is linear and time–invariant), which are almost always violated in practice. Certainly rigorous theory is important, and a valid criticism of any research is that it proceeds in ignorance of relevant theoretical frameworks and previous research, but an experimental methodology seems necessary for developing control applications involving complex nonlinear systems.

Despite the association of connectionist methods with applications involving weak assumptions about system structure, it is not appropriate to characterize these methods as knowledge–free, or *tabula rasa*, approaches to learning. Networks can be applied with little regard for prior knowledge (and often are), but they need not be. Indeed, as I pointed out in this report, networks provide numerous avenues for introducing bias into the learning process on the basis of prior knowledge. I think it is more accurate to characterize connectionist methods as special kinds of memory–intensive approaches to computation sharing many of the characteristics of memory–based reasoning as discussed by Stanfill and Waltz [91] and illustrated in a control application by Atkeson and Reinkensmeyer [6]. Networks generally have a large number of parameters compared with more traditional uses of parameter estimation techniques, and some networks are basically lookup tables that automatically provide interpolation between, and extrapolation beyond, the data stored. As in the case of conventional computer memory, how this storage capacity is initialized depends on how it is to be used.

However, the connectionist approach provides a generalized concept of memory that I think will turn out to be its most important contribution to control engineering. Storing information in connection weights generalizes the idea of inserting information into memory locations, and on–line parameter estimation, whether used in supervised or reinforcement learning modes, generalizes the process of memory storage. Because adaptive connectionist networks fit in the range between structureless lookup tables and highly constrained model–based parameter estimation, they seem well–suited for the acquisition and storage of control information. These methods suggest how new techniques for adaptive control can be developed which take full advantage of the possibilities for fabricating associative memory systems having high capacity, high speed, and the ability to usefully interpolate and extrapolate in real time.

References

[1] M. A. Aizerman, E. M. Braverman, and L. I Rozonoer. On the method of potential functions. *Automatika i Telemekhanika*, 26, 1964.

[2] J. S. Albus. Mechanisms of planning and problem solving in the brain. *Mathematical Biosciences*, 45:247–293, 1979.

[3] C. W. Anderson. *Learning and Problem Solving with Multilayer Connectionist Systems.* PhD thesis, University of Massachusetts, Amherst, MA, 1986.

[4] C. W. Anderson. Strategy learning with multilayer connectionist representations. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 103–114, Irvine, CA, 1987. Morgan Kaufmann.

[5] M. A. Arbib. *Brains, Machines, and Mathematics, Second Edition.* Springer-Verlag, New York, 1987.

[6] C. G. Atkeson and D. J. Reinkensmeyer. Using associative content-addressable memories to control robots. In *IEEE Conference on Decision and Control*, 1988.

[7] A. G. Barto. Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology*, 4:229–256, 1985.

[8] A. G. Barto. Game-theoretic cooperativity in networks of self-interested units. In J. S. Denker, editor, *Neural Networks for Computing*, pages 41–46. American Institute of Physics, New York, 1986.

[9] A. G. Barto. An approach to learning control surfaces by connectionist systems. In M. A. Arbib and A. R. Hanson, editors, *Vision, Brain and Cooperative Computation.* MIT Press/Bradford Books, Cambridge, MA, 1987.

[10] A. G. Barto. From chemotaxis to cooperativity: Abstract exercises in neuronal learning strategies. In R. Durbin, R. Maill, and G. Mitchison, editors, *The Computing Neuron.* Addison-Wesley, 1989.

[11] A. G. Barto and P. Anandan. Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:360–375, 1985.

[12] A. G. Barto and C. W. Anderson. Structural learning in connectionist systems. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Irvine, CA, August 1985.

[13] A. G. Barto and M. I. Jordan. Gradient following without back-propagation in layered networks. In *Proceedings of the IEEE First Annual Conference on Neural Networks*, pages II629–II636, San Diego, CA, 1987.

[14] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846, 1983. Reprinted in J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, The MIT Press, Cambridge, MA, 1988.

[15] A. G. Barto, R. S. Sutton, and P. S. Brouwer. Associative search network: A reinforcement learning associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, 40:201–211, 1981.

[16] A. G. Barto, R. S. Sutton, and C. Watkins. Prediction, control, and learning. In M. Gabriel and J. W. Moore, editors, *Learning and Computational Neuroscience*. The MIT Press, Cambridge, MA. To appear.

[17] E. B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[18] L. B. Booker. *Intelligent Behavior as an Adaptation to the Task Environment*. PhD thesis, University of Michigan, Ann Arbor, MI, 1982.

[19] A. E. Bryson, Jr. and Y. C. Ho. *Applied Optimal Control*. Blaisdel Publishing Co., 1969.

[20] R. R. Bush and F. Mosteller. *Stochastic Models for Learning*. Wiley, New York, 1955.

[21] J. S. Denker, D. B. Schwartz, B. S. Wittner, S. A. Solla, R. E. Howard, L. D. Jackel, and J. J. Hopfield. Automatic learning, rule extraction, and generalization. *Complex Systems*, 1:877–922, 1987.

[22] M. Derthick. Mundane reasoning by parallel constraint satisfaction. Technical Report CMU-CS-88-182, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, 1988.

[23] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[24] W. K. Estes. Toward a statistical theory of learning. *Psychololgical Review*, 57:94–107, 1950.

[25] B. G. Farley and W. A. Clark. Simulation of slef-organizing systems by digital computer. *IRE Transactions on Information Theory*, PGIT-4, 1954.

[26] J. A. Feldman and D. H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6:205–254, 1982.

[27] J. A. Franklin. Learning control in a robotic system. In *Proceedings of the 1987 International Conference on Systems, Man, and Cybernetics*, pages 466–470, October 1987.

[28] J. A. Franklin. Refinement of robot motor skills using reinforcement learning. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 1096–1101, December 1988.

[29] K. S. Fu. Learning control systems—Review and outlook. *IEEE Transactions on Automatic Control*, pages 210–221, 1970.

[30] B. R. Gaines and J. H. Andreae. A learning machine in the context of the general control problem. In *Proceedings of the Third IFAC Congress*, pages 14B.1–14B.8, London, 1966. Institute of Mechanical Engineers.

[31] S. I. Gallant. Three constructive algorithms for network learning. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, August 1986.

[32] C. L. Giles and T. Maxwell. Learning, invariance, and generalization in higher-order networks. *Applied Optics*, 26, December 1987.

[33] G. C. Goodwin and K. S. Sin. *Adaptive Filtering Prediction and Control.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1984.

[34] S. Grossberg and M. Kuperstein. *Neural Dynamic of Adaptive Sensory-motor Control: Ballistic Eye Movements.* Elsevier, Amsterdam, 1986.

[35] V. Gullapalli. A stochastic algorithm for learning real-valued functions via reinforcement feedback. Technical Report 88-91, University of Massachusetts, Amherst, MA, 1988.

[36] S. E. Hampson. *A Neural Model of Adaptive Behavior.* PhD thesis, University of California, Irvine, CA, 1983.

[37] G. E. Hinton. Distributed representations. Technical Report CMU-CS-84-157, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa., 1984.

[38] G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, 1986.

[39] G. E. Hinton. Connectionist learning procedures. Technical Report CMU-CS-87-115, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, 1987.

[40] G. E. Hinton. Representing part-whole hierarchies in connectionist networks. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, 1988.

[41] G. E. Hinton and J. A. Anderson, editors. *Parallel Models of Associative Memory.* Erlbaum, Hillsdale, NJ, 1981.

[42] J. H. Holland. Escaping brittleness: The possibility of general-purpose learning algorithms applied to rule-based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach, Volume II.* Morgan Kaufmann, Los Altos, CA, 1986.

[43] V. Honavar and L. Uhr. A network of neuron-like units that learns to perceive by generation as well as reweighting of its links. In D. Touretsky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School.* Morgan Kaufmann Publishers, 2929 Campus Drive, Suite 260, San Mateo, CA 94403, 1988.

[44] K. Hornik, M. Stinchcombe, and H. White. Multi-layer feedforward networks are universal approximators. Technical report, Department of Economics, University of California, San Diego, CA, June 1988.

[45] A. G. Ivakhnenko. Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1:364–378, 1971.

[46] M. I. Jordan. Sequential dependencies and systems with excess degrees of freedom. Technical Report 88-27, University of Massachusetts, Amherst, MA, 1988.

[47] M. I. Jordan and D. E. Rumelhart. Supervised learning with a distal teacher. 1989. Submitted to: *Cognitive Science.*

[48] J. S. Judd. Complexity of connectionist learning with various node functions. Technical Report 87-60, University of Massachusetts, Amherst, MA, 1987.

[49] J. S. Judd. Learning in networks is hard. In *Proceedings of the IEEE First Annual Conference on Neural Networks*, volume 2, pages 685–692, San Diego, CA, 1987.

[50] J. S. Judd. On the complexity of loading shallow neural networks. *Journal of Complexity*, September 1988. Special issue on Neural Computation, in press.

[51] M. Kawato. Computational schemes and neural network models for formation and control of multijoint arm trajectory. In T. Miller, R. S. Sutton, and P. J. Werbos, editors, *Neural Networks for Control.* The MIT Press, Cambridge, MA. To appear.

[52] M. Kawato, K. Furukawa, and R. Suzuki. A hierarchical neural-network model for control and learning of voluntary movement. *Biological Cybernetics*, 57:169–185, 1987.

[53] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki. Hierarchical neural-network model for voluntary movement with application to robotics. *IEEE Control Systems Magazine*, 8:8–16, April 1988.

[54] A. H. Klopf. Brain function and adaptive systems—A heterostatic theory. Technical Report AFCRL-72-0164, Air Force Cambridge Research Laboratories, Bedford, MA, 1972. A summary appears in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, 1974, IEEE Systems, Man, and Cybernetics Society, Dallas, TX.

[55] A. H. Klopf. *The Hedonistic Neuron: A Theory of Memory, Learining, and Intelligence.* Hemishere, Washington, D.C., 1982.

[56] A. H. Klopf. A neuronal model of classical conditioning. *Psychobiology*, 16:85–125, 1988.

[57] T. Kohonen. *Associative Memory: A System Theoretic Approach.* Springer, Berlin, 1977.

[58] M. Kuperstein. Adaptive visual-motor coordination in multijoint robots using parallel architecture. In *IEEE International Conference on Robotics and Automation*, pages 1595–1602, 1987.

[59] A. Lapedes and R. Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical Report LA-UR-87-2662, The Los Alamos National Laboratory, Los Alamos, NM, 1987.

[60] Y. le Cun. Une procedure d'apprentissage pour reseau a sequil assymetrique [A learning procedure for asymmetric threshold network]. *Proceedings of Cognitiva*, 85:599–604, 1985.

[61] Y. le Cun. A theoretical framework for back-propagation. In D. Touretsky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School.* Morgan Kaufmann Publishers, 2929 Campus Drive, Suite 260, San Mateo, CA 94403, 1988.

[62] L. Ljung and T. Söderstrom. *Theory and Practice of Recursive Identification.* The MIT Press, Cambridge, MA, 1983.

[63] P. Mars. Neural nets and robotic control. Technical Report RIPRREP/1000/33/88, School of Engineering and Applied Science, University of Durham, Durham DH1 3LE, UK, 1988.

[64] J. M. Mendel and K. S. Fu. *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications.* Academic Press, New York, 1970.

[65] J. M. Mendel and R. W. McLaren. Reinforcement learning control and pattern recognition systems. In J. M. Mendel and K. S. Fu, editors, *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*, pages 287–318. Academic Press, New York, 1970.

[66] D. Michie and R. A. Chambers. BOXES: An experiment in adaptive control. In E. Dale and D. Michie, editors, *Machine Intelligence 2*, pages 137–152. Oliver and Boyd, 1968.

[67] W. T. Miller. Sensor based control of robotic manipulators using a general learning algorithm. *IEEE Journal of Robotics and Automation*, 3:157–165, 1987.

[68] M. L. Minsky. *Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem.* PhD thesis, Princeton University, 1954.

[69] M. L. Minsky. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49:8–30, 1961. Reprinted in E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought.* McGraw-Hill, New York, 406–450, 1963.

[70] M. L. Minsky and S. A. Papert. *Perceptrons: an Introduction to Computational Geometry.* The MIT Press, Cambridge, MA, 1969. An expanded edition was published by The MIT Press in 1988.

[71] J. Moody and C. Darken. Learning with localized receptive fields. In D. Touretsky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School.* Morgan Kaufmann Publishers, 2929 Campus Drive, Suite 260, San Mateo, CA 94403, 1988.

[72] M. C. Mozer and P. Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. Touretzky, editor, *Proceedings of the IEEE Conference on Neural Information Processing Systems*, Denver,CO, 1989.

[73] P. Munro. A dual back-propagation scheme for scalar reward learning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pages 165–176, Seattle, WA, 1987.

[74] K. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[75] K. S. Narendra and M. A. L. Thathachar. Learning automata—A survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 4:323–334, 1974.

[76] M. Niranjan and F. Fallside. Neural networks and radial basis functions in classifying static speech patterns. Technical Report CUED/F-INFENG/TR 22, University Engineering Department, Cambridge, Cambridge, CB2 1PZ, England, 1988.

[77] D. B. Parker. Learning logic. Technical Report TR-47, Massachusetts Institute of Technology, 1985.

[78] J. Pollack. Recursive auto-associative memeory: Devising compositional distributed representations. Technical Report MCCS-88-124, Computing Research Laboratory, New Mexico State University, Las Cruces, NM 88003, 1988.

[79] D. Psaltis, A. Sideris, and A. A. Yamamura. A multilayered neural network controller. *IEEE Control Systems Magazine*, 8:17–21, April 1988.

[80] M. H. Raibert. A model for sensorimotor control and learning. *Biological Cybernetics*, 29:29–36, 1978.

[81] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, 6411 Chillum Place N.W., Washington, D.C., 1961.

[82] R. Rosenfled and D. S. Touretzky. A survey of coarse-coded symbol memories. In D. Touretsky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School.* Morgan Kaufmann Publishers, 2929 Campus Drive, Suite 260, San Mateo, CA 94403, 1988.

[83] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol.1: Foundations.* Bradford Books/MIT Press, Cambridge, MA, 1986.

[84] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, pages 210–229, 1959. Reprinted in E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought*, McGraw-Hill, New York, 1963.

[85] A. L. Samuel. Some studies in machine learning using the game of checkers. II—Recent progress. *IBM Journal on Research and Development*, pages 601–617, November 1967.

[86] G. N. Saridis and H. D. Gilbert. Self-Organizing approach to the stochastic fuel regulator problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:186–191, 1970.

[87] O. Selfridge, R. S. Sutton, and A. G. Barto. Training and tracking in robotics. In *Proceedings of the Ninth International Joint Conference of Artificial Intelligence*, Los Angeles, CA, August 1985.

[88] J. Sklansky and G. N. Wassel. *Pattern Classifiers and Trainable Machines.* Springer-Verlag, New York, 1981.

[89] S. A. Solla. Learning and generalization in layered neural networks: The contiguity problem. In G. Dreyfus and L. Personnoz, editors, *Proceedings of the nEuro'88 Conference.* To appear.

[90] V. Srinivasan, A. G. Barto, and B. E. Ydstie. Pattern recognition and feedback via parallel distributed processing. In *Annual Meeting of the AIChE*, Washington D. C., November 1988. Abstract to appear.

[91] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228, December 1986.

[92] R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts, Amherst, MA, 1984.

[93] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[94] R. S. Sutton and A. G. Barto. Time-derivative models of pavlovian conditioning. In M. Gabriel and J. W. Moore, editors, *Learning and Computational Neuroscience.* The MIT Press, Cambridge, MA. To appear.

[95] R. S. Sutton and A. G. Barto. Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 88:135–171, 1981.

[96] R. S. Sutton and A. G. Barto. A temporal-difference model of classical conditioning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, 1987.

[97] D. S. Touretzky. BoltzCONS: Reconciling connectionism with the recursive nature of stacks and trees. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, 1986.

[98] D. S. Touretzky and G. E. Hinton. Symbols among the neurons: Details of a connectionist inference architecture. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1985.

[99] M. L. Tsetlin. *Automaton Theory and Modeling of Biological Systems.* Academic Press, New York, 1973.

[100] M. D. Waltz and K. S. Fu. A heuristic approach to reinforcment learning control systems. *IEEE Transactions on Automatic Control*, 10:390–398, 1965.

[101] C. J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, Cambridge University, Cambridge, England, 1989. Pending.

[102] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[103] P. J. Werbos. Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems Yearbook*, 22:25–38, 1977.

[104] P. J. Werbos. Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 1987.

[105] P. J. Werbos. Generalization of back propagation with applications to a recurrent gas market model. *Neural Networks*, 1, 1988.

[106] B. Widrow. In T. Miller, R. S. Sutton, and P. J. Werbos, editors, *Neural Networks for Control.* The MIT Press, Cambridge, MA. To appear.

[107] B. Widrow. Adaptive inverse control. In *Second IFAC Workshop on Adaptive Systems in Control and Signal Processing*, pages 1–5, Lund, Sweden, 1986.

[108] B. Widrow, N. K. Gupta, and S. Maitra. Punish/reward: Learning with a critic in adaptive threshold systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 5:455–465, 1973.

[109] B. Widrow and M. E. Hoff. Adaptive switching circuits. In *1960 WESCON Convention Record Part IV*, pages 96–104, 1960. Reprinted in J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, The MIT Press, Cambridge, MA, 1988.

[110] B. Widrow, J. McCool, and B. Medoff. Adaptive control by inverse modeling. In *Twelfth Asilomar Conference on Circuits, Systems, and Computers*, 1978.

[111] B. Widrow and F. W. Smith. Pattern-recognizing control systems. In *Computer and Information Sciences (COINS) Proceedings*, Washington, D.C., 1964. Spartan.

[112] B. Widrow and S. D. Stearns. *Adaptive Signal Processing.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.

[113] R. J. Williams. Reinforcement learning in connectionist networks: A mathematical analysis. Technical Report ICS 8605, Institute for Cognitive Science, University of California at San Diego, La Jolla, CA, 1986.

[114] R. J. Williams. Reinforcement-learning connectionist systems. Technical Report NU-CCS-87-3, College of Computer Science, Northeastern University, 360 Huntington Avenue, Boston, MA, 1987.

[115] R. J. Williams. On the use of backpropagation in associative reinforcement learning. In *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, 1988.

[116] I. H. Witten. An adaptive optimal controller for discrete-time Markov environments. *Information and Control*, pages 286–295, 1977.