NEURAL NETWORKS FOR TIME-VARYING DATA

Survey presentation for the workshop:

Neural Networks for Statistical and Economic Data

Dublin, 10-11 December 1990

Richard Rohwer
Centre for Speech Technology Research
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
rr@cstr.ed.ac.uk

Abstract

This paper reviews some basic issues and methods involved in using neural networks to respond in a desired fashion to a temporally-varying environment. Some popular network models and training methods are introduced. A speech recognition example is then used to illustrate the central difficulty of temporal data processing: learning to notice and remember relevant contextual information. Feedforward network methods are applicable to cases where this problem is not severe. The application of these methods are explained and applications are discussed in the areas of pure mathematics, chemical and physical systems, and economic systems. A more powerful but less practical algorithm for temporal problems, the moving targets algorithm, is sketched and discussed. For completeness, a few remarks are made on reinforcement learning.

1 Introduction

Neural network models can be used to process to time-varying data for various purposes using various methods. Research aimed at improving the methodology makes contact with a broad range of disciplines and raises a panoply of difficult, though enticing, questions. This happens largely because the brain not only lives in an environment filled with time-varying data, but also generates internal time-varying signals of its own. The internal signals vary from direct responses to environmental data to obscure mechanisms for internal processing. Therefore the subject of temporal data processing with neural networks draws one towards a study of the brain's inner mechanisms, about which very little is known. For engineering purposes, we hope to understand some of the basic computational principles involved without getting bogged down in the intricate biological details.

The first section reviews a simple recurrent network model, which can have highly complex dynamics, and two feedforward models which are inherently static. Next, the central problem in processing temporal data is illustrated with a speech-recognition example. This is the problem of temporal credit assignment, or learning relevant context. There are applications for which the temporal credit assignment problem is readily solvable, and others for which useful results can be obtained without solving it entirely. These include prediction problems for certain mathematical, chemical, and economic systems, and some speech recognition problems.

The moving targets algorithm is perhaps unique in it's ability to handle difficult temporal credit assignment problems. The basic ideas behind this algorithm are introduced and discussed. Unfortunately the algorithm is computationally impractical.

Reinforcement learning is briefly mentioned.

2 Basic Neural Network Models

Neural network models specify rules for changing the *output values* of model neurons, or *nodes*, with time. These output values are sometimes regarded as crude models of neural firing rates. Let y_{it} denote the value of node i at discrete time t. In a widely-used class of models, a subset I of the nodes are designated as *inputs*. These are assigned values Y_{it} taken from a model of the network's external environment. The values of the remaining nodes are computed for time $t+1$ from their values at time t according to the rule:

$$y_{i,t+1} = \begin{cases} f(\sum_j w_{ij} y_{j,t}) & i \notin I \\ Y_{i,t+1} & i \in I \end{cases} \tag{1}$$

where the *weight* w_{ij} models the strength of a synaptic connection *from* node j *to* node i, and f is a differentiable function with constant asymptotes, such as

$$f(x) = 1/(1+e^{-x}) \tag{2}$$

which varies smoothly from 0 at $-\infty$ to 1 at ∞. One input, say node 0, is traditionally assigned the constant value 1.0 so that w_{i0} provides a constant offset or *bias* for the weighted sum computed by node i. Through rule (1) the weights specify the network *dynamics*, the manner in which the state (the set of node values) changes with time.

A network model can be *trained* to produce a desired sequence of *target* values on a subset T of the non-input nodes. The set H of nodes which are neither input nor target nodes are called *hidden* nodes. Let Y_{it} (for $i \in T$) denote the target value for node i at time t. (There is no confusion with (1) because $I \cap T = \emptyset$.) A scalar measure of the network's performance is given by

$$E = \tfrac{1}{2} \sum_{i \in T} \sum_t (y_{it} - Y_{it})^2. \tag{3}$$

If the network operates perfectly, $E = 0$; otherwise $E > 0$. A popular procedure for training a network is *back propagation of error through time* [23], which proceeds by computing the derivatives dE/dw_{ij} and using these to incrementally adjust the weights to slightly better values. This procedure is often highly effective, but one is guaranteed neither that a perfect solution $E = 0$ exists, nor that the smallest value of E will be found by this procedure. That is, a *local* minimum can always be found, but not necessarily a *global* minimum. Even if a global minimum is found, there still may be other global solutions which be preferable for some reason not encoded in (3).

A useful subclass of network models are the *feedforward* networks. Nodes of these networks have no connections to themselves ($w_{ii} = 0$) and no feedback paths ($|w_{ij}| > 0$ implies $w_{ji} = 0$). Typically the nodes of feedforward networks are arranged in numbered layers, with nonzero weights from each layer going to higher-numbered layers, but none going in the reverse direction and none going within a layer. The lowest layer is assigned input from the environment. If the input remains fixed in an L-layer feedforward net, then every node value in the network remains fixed after $L-1$ time steps. By specifying target values for the highest layer, and using its derivatives to minimize a function similar to (3), these networks can be trained to implement a mapping from input vectors to target vectors which agrees with a set of examples. The hidden nodes in such networks are those in the middle layers between the inputs and targets. It has been proven that a feedforward network with a single hidden layer can approximate any mapping to arbitrary accuracy (although a large number of hidden units may be required) [7].

A popular class of feedforward networks is the *radial basis function networks* [17, 18, 16, 4]. These networks have one layer of hidden nodes, each of which implements a radial basis function. Hidden node a is assigned a *centre* in the input space having input coordinates c_{ia} and a *radius* r_a, and computes an output which is large only for inputs near its centre (on a scale set by its radius). The output of this network is feed through a linear transformation. Specifically, the output y_{ip} produced by example p is

$$y_{ip} = \sum_a w_{ia} g \left(\frac{\sqrt{\sum_i (Y_{ip} - c_{ia})^2}}{r_a} \right) \tag{4}$$

where Y_{ip} is the input for example p and g is typically a Gaussian $g(x) = e^{-x^2}$. Non-Euclidian distance measures are sometimes used. The loosley-defined region for which a radial basis function has a significant output is its *receptive field.* Usually the centres and radii of a radial basis functions network are assigned using one of a great variety of simple algorithms which ensure that most input data points fall within a few receptive fields. Then the weights are adapted to minimize an error measure similar to (3). This problem amounts to solving a large linear system of equations, which can be accomplished using standard methods [15] much more quickly than a minimization algorithm can be applied to a standard feedforward back propagation network. This feature is a major attraction of radial basis functions.

The name "radial basis functions" derives from the spherical symmetry of the receptive fields. However this is not an important property of the method and generalizations to less symmetric basis functions are commonly used.

A network which is not feedforward is *recurrent.*

3 Distant relevant context: a speech example

Let us introduce some temporal processing issues by looking at some sample data: a short segment of speech. A spectrogram of the phrase "quite quiet at church" is shown in figure

1. This shows the energy in the acoustic signal as a function of frequency (vertical axis) and time (horizontal axis). Dark areas on the spectrogram signify high energy. The raw time-domain waveform is shown at the top of the figure. A phonetician has assigned phoneme labels to the time segments indicated along the bottom of the spectrogram. The neural network model (1) might be trained to recognize speech by feeding the energy at each monitored frequency into a corresponding input node and assigning a specific target node to 1.0 when a corresponding phoneme is present, and 0.0 otherwise.

There are two /k/ phonemes in the figure. Their characteristics include a moderately long period of silence, followed by a rapid burst of energy spread widely over the spectrum. After another moderate time interval, the onset of voicing produces three strong resonances (*formants*) in the low frequencies. The divergence of the second and third formants as the burst recedes helps to discriminate /k/'s from /p/'s and /t/'s. This variation occurs during the course of the following two phonemes.

This example shows that the information needed to recognize phonemes occurs on widely disparate timescales. A speech recognition device must be attentive to the burst, which occurs during a few milliseconds, and must also notice phenomena which occur over hundreds of milliseconds. Most of the information occurring on short timescales is irrelevant or redundant with information from previous times, but some is crucial. This example shows a need for networks to learn to notice and *remember* information of potential use for the future, possibly the distant future as measured by the timescales on which some relevant events occur. We shall argued that the difficulty of a temporal information processing problem has much to do with its requirement for such *distant contextual information.* Another obvious problem is that some information relevant the the judgement of whether a /k/ is present occurs long after the /k/ has given way to other phonemes. Therefore the target values should be demanded from the network at a time considerably later than they occur according to the phonetician's labels. This increases the time during which information must be remembered.

Of course, the burst of the /k/ is not typical of the contextual information which must be remembered, most of the time the signal is repetitive or otherwise uninformative. It is just as important to forget useless information rapidly as it is to remember useful information. The network must learn to selectively remember relevant contextual information only, even though there may be no sound way to judge relevance until the distant future.

4 Neural network capabilities in principle and in practice

It is easy to show that a single node connected to itself can be configured to act as a flip-flop memory element, and equally simple to show that a single node can be configured to perform the not-and Boolean function. It is possible to emulate any computer by building a machine out of these two components, so in principle the neural network model (1) can do any calculation of practical interest. Furthermore, extensive numerical studies have shown that this model typically produces complex motion involving long time-scales [19]. Unfortunately, all this does not imply that networks are easily trained from examples to do complex temporal tasks.

Most existing training methods work only in *Markovian* [24], or nearly Markovian environments. This means that the target values at any time step can be determined uniquely from the input and target values from one, or a small number of time steps in the recent past. If the present state of the environment does not contain enough information to enable a unique prediction of the targets at the following time step, then there is no hope unless the hidden nodes happen to encode the missing information. This may be the case if the necessary information lies somewhere in the past, and the network was clever enough to respond to that information by encoding it in some hidden nodes, and to arrange the dynamics so that this information is not lost before it is needed. The task of deciding what the hidden nodes should have done in the past to reduce errors in the future is called the *temporal credit assignment problem* [28].

Thus, the hidden nodes play an essential role in transmitting temporally distant relevant contextual information to the future. Unfortunately, hidden nodes complicate the expressions for the derivatives needed for back propagation. This is unsurprising given the complex task they must learn to perform. Worse still, inspection of the derivative formulas shows that unless certain improbable cancellations occur, the expressions for these derivatives are dominated by near-context information. In other words, the derivatives will not suggest a weight adjustment which makes use of distant context until an optimum based on recent context has been found to an absurd number of decimal places of accuracy. Therefore methods based on calculating dE/dw_{ij} do temporal credit assignment in a manner appropriate only for Markovian, or nearly Markovian environments.

5 Delay lines

It is possible to convert a non-Markovian environment to a Markovian one by introducing *delay lines*. This augments the current state with copies of the past $\tau_{\max}$ states, so that (1) becomes

$$y_{i,t+1} = \begin{cases} f(\sum_{\tau=0}^{\tau_{\max}} \sum_j w_{ij}^{(\tau)} y_{j,t-\tau}) & i \notin I \\ Y_{i,t+1} & i \in I \end{cases} . \qquad (5)$$

The delay lines ($w_{ij}^{(\tau)}$, $\tau > 0$) transmit all information $\tau_{\max}$ steps to the future. If distant context is not needed then $\tau_{\max}$ can be relatively small, making this a practical technique. If large $\tau_{\max}$ is required, the network is likely to be overwhelmed with large amounts of redundant and irrelevant information. This problem is compounded by the increased number of variable parameters (the delay-line weights) in the model. A model with too many parameters tends to *overfit* the training data, becoming highly sensitive to the idosyncracies of the data instead of features which will likely be present in new data [2].

6 Prediction with feedforward networks

In Markovian environments, or environments which can be made Markovian by introducing a small number of delay lines, the hidden nodes do not need to perform any memory function. The problem becomes feedforward in nature: A mapping must be learned which outputs the targets for the next time step given the inputs and targets for the current step (or past few steps). After training, this network is operated as a recurrent network; the input values at any time include target values computed earlier. However for training purposes, the network can be regarded simply as a feedforward network learning a mapping. Hidden nodes may be needed to enable the network to represent the desired mapping, but in feedforward networks they are not burdened with a responsibility to detect and preserve relevant contextual information, so the training difficulties discussed earlier do not arise.

This method has been used to good effect in a variety of prediction problems. Lapedes and Farber [13] trained a feedforward network to predict a chaotic time series generated by the Glass-Mackey equation, a scalar delay-differential equation. In order to uniquely specify a solution to this equation, it is necessary to (arbitrarily) specify a portion of the trajectory over a finite time interval, the length of which is the "delay" parameter in the equation. Thus the equation is local in time. However, the complete specification of such a trajectory fragment requires an infinite number of parameters (one for each real-valued instant in the interval), so the phase space of the system is infinite-dimensional. (A phase space of a system is a minimal set of variables in terms of which the system is rendered Markovian.) However, the solutions to this equation do not wander about the entire phase space, but instead evolve toward a fractal attractor of dimension between 2 and 3. Once on the attractor, the system should be predictable from its values at roughly 3 different time steps. The presence of a low-dimensional attractor is a typical property of high-dimensional deterministicly chaotic systems, but there is no general method for deriving coordinates for the attractor's niche in phase space in terms of the original coordinates of the problem.

Lapedes and Farber used a feedforward network with 2 layers of hidden units to map 5 past values of a Glass-Makey trajectory to 1 future value. The past values were uniformly selected from the delay interval. They established that it is better to predict a nearby future value and iterate the mapping to predict the more distant future than to predict the distant future directly. This is reasonable because it should be easier to teach the network to emulate the underlying local equation than to model the equation together with its solutions over long time periods. Their results were substantially better than other standard prediction methods, including polynomial fits, iterated polynomials, and linear mappings.

A similar technique was used by Adomaitis, et. al. [1] to predict properties of an electrochemical reaction governed in principle by local differential equations. Weigend, Huberman and Rumelhart [27] compared a simple feedforward network, a radial basis functions network, and a standard method in a sunspot-cycle prediction problem and a "computational ecology" model of computers competing for resources. They provide a quantitative measure of the extent to which the network makes use of its nonlinearity. They report that radial basis function methods require more training data to obtain valid generalization, but they have a great speed advantage. Moody and Darken [14] report a similar tradeoffs in a comparison between radial basis functions and standard sigmoidal networks for predicting the Glass-Mackey system. Jones, et. al. [9] claim to have a method which combines the advantages of sigmoidal and radial basis functions networks.

In a joint project with Morio Yoda and Masakazu Takeoka of Nikko Securities Company, Takashi Kimoto and Kazuo Asakawa of the Fujitsu Computer-Based Systems Laboratory in Kawasaki have used these techniques to predict when to buy or sell stocks used to compute the Japanese TOPIX index [11]. The results were better than a simple buy-and-hold strategy. Of course, little is known of any equations governing economic systems, but the success of this experiment suggests that the laws of economics, such as there be, are at least somewhat local in time. Chen, Cowan, Grant, and Billings applied this type of method to model unemployment levels in (then) West Germany [5], but the ability of their model to make new predictions has not been tested. Collard trained a similar network on commodity market data from 1988 and reports that its predictions for 1989 could have produced a tidy profit [6]. The input data for this experiment included non-economic information such as weather data as well as economic data of relevance to the commodoties market.

As mentioned earlier extensive use of delay lines to bring contextual information to bear multiplies the number of parameters in the model and brings in irrelevant and redundant information as well. Waibel, et al. [12, 8] have added constraints to a network of this type in order to combat this problem in a speech recognition application. They reason that the network should always look for the same features of the speech signal at every time step. Therefore the weights associated with one delay line should be constrained to equal the corresponding weights on another. This trick, *tying weights*, reduces the number of parameters and increases the relevance of the information sent forward from the past. (In fact this system uses much more elaborate constraints than indicated here.)

7 An algorithm for non-Markovian problems

The popular back propagation algorithm for feedforward networks trains hidden nodes, and has a natural extension to temporal problems, but it turns out that this algorithm's ability to utilize contextual information diminishes exponentially with time. The "Moving Targets" algorithm of Rohwer [20, 21, 22] reduces this to a linear decrease, but suffers from serious practical difficulties. The basic idea of this algorithm is to treat hidden nodes as target nodes with variable target values. This allows errors to be allocated directly to the hidden nodes, so that the sum in the error measure (3) can be extended to

$$E = \tfrac{1}{2} \sum_{(itp) \in T \cup H} \{y_{itp} - Y_{itp}\}^2. \tag{6}$$

The "moving target" variables, $Y_{itp} for(itp) \in H$, are lumped in with the weights in the minimization problem; they are initialized randomly and optimized by a derivative-based procedure. If minimization is successful, the moving targets are discarded and the weights retained. Errors on target nodes can be traded for errors on hidden nodes at possibly quite distant time steps if that helps to minimize this sum. This provides greater flexibility in temporal credit assignment than is possible with the standard method in which the weights are the only variables.

The moving targets algorithm has been successfully applied to a problem which requires contextual information from 100 time steps in the past. The training data for this example contains 2 sequences. In each sequence a single input node is given a value of 1.0

at time step 100. It is 0.0 at all other times in sequence 2, and 0.0 at all other times in sequence 1 except at time step 1, when it is 1.0. A single target node is asked to respond with 0.0 at all times for sequence 2, and for time steps 0 to 100 of sequence 1, but with 1.0 after time step 100 in sequence 1. Thus, the input sequences are distinguished only by an event at time 1, and the targets are identical until time 101. Using 1 hidden node it is easy to "hand-wire" a weight matrix which will solve this problem; the hidden node needs to "turn on" in response to the first input 1.0-value in sequence 1, and to stay on (using a positive self-weight) for all time. That way the two sequences will be distinguished at time step 100 by the state of the hidden node. When the moving targets algorithm is applied to this problem, the network quickly adjusts so that the largest errors are on the target nodes at time step 100 in each training sequence. The moving target value of the hidden node settles to 0.5 for most time steps. As training progresses, the moving target values on the hidden nodes at time step 100 increase for sequence 1 and decrease for sequence 2, thereby providing the distinction needed to reduce the target node error at time 101. The moving targets at time step 99 then respond similarly in order to accommodate the errors at step 100. This process carries on until the moving targets are distinguished at time step2, at which point they can be "anchored" on a distinction in the inputs at step 1.

Although this example demonstrates that this algorithm has considerable capabilities for non-Markovian problems, practical experience shows that it has serious disadvangages. In a large problem the minimization process is beset by a large number of moving target variables which must be optimized. Presumably for this reason, the minimization proceeds at an impractically slow pace, and local minima are frequently encountered.

8 Reinforcement learning

Reinforcement learning techniques have been an important approach to temporal credit assignment problems for many years [25, 26]. In these problems one subnetwork learns to form a world model while another learns an optimal *policy* for reacting to a given state of the environment, in order to maximize a reinforcement signal at a later time. These methods provide a structure into which other algorithms can be incorporated, and may prove useful for extending the context sensitivity of back propagation. They show particular promise for robotics and other control applications [10, 3].

9 Conclusions

Simple neural network models have the power to do arbitrary computations with time-varying data, and are amenable to learning from examples. However, existing training methods are either unable to handle problems requiring attention to distant temporal context, or are highly impractical from a computational point of view. Nevertheless there is a usefully large class of problems in which distant temporal context is not particularly important. Methods using techniques for training feedforward networks show considerable promise for this type of application. Early results suggest that many prediction problems from physics, chemistry, and economics may be profitably approached in this way.

References

[1] R. A. Adomaitis, R. M. Farber, J. L. Hudson, I. G. Kevrekidis, M. Kube, and A. S. Lapedes. Applications of neural nets to system identification and bifurcation anjalysis of real world experimental data. Technical Report LA-UR-90-515, Los Alimos National Laboratory, Los Alamos, NM, 1990.

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

[3] Carlos Brodie. Back propagation through predictive forward models. MSc thesis, Dept. of Artificial Intelligence, Edinburgh University, 1990.

[4] D. S. Broomhead and David Lowe. Multi-variable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[5] S. Chen, C. F. N. Cowan, P. M. Grant, and S. A. Billings. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, to appear.

[6] Joe Collard. A back propagation artificial neural network commodity trader. Technical report, Martingale Research Corporation, 100 Allentown Parkway, Suite 211, Allen, Texas 75002, USA, 1990.

[7] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183, 1989.

[8] John B. Hampshire II and Alex Waibel. Connectionist architectures for multispeaker phoneme recognition. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 203–210. Morgan Kaufmann, San Mateo CA, 1990.

[9] R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee, P. S. Lewis, and S. Qian. Function approximation and time series prediction with neural networks. Technical Report LA-UR-90-21, Los Alimos National Laboratory, Los Alamos, NM, 1989.

[10] M. I. Jordan and R. A. Jacobs. Learning to control an unstable system with forward modelling. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 324–331, San Mateo CA, 1990. Morgan Kaufmann.

[11] Takashi Kimoto, Kazuo Asakawa, Morio Yoda, and Masakazu Takeoka. Stock market prediction system with modular neural networks. Technical report, Fujitsu Computer-Based Systems Laboratory, 1015 Kamikodanaka, Nakahara-Ku, Kawasaki 211, Japan, 1990.

[12] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–44, 1990.

[13] Alan Lapedes and Robert Farber. How neural nets work. In Dana Z. Anderson, editor, *Neural Information Processing Systems, Denver CO 1987*, pages 442–457. American Institute of Physics, New York, 1988.

[14] John Moody and Christian J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281–294, 1989.

[15] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vettering. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, 1988.

[16] S. Renals. Radial basis functions network for speech pattern classification. *Electronics Letters*, 25:437–439, 1989.

[17] Steve Renals and Richard Rohwer. Learning phoneme recognition using neural networks. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 413–416, Glasgow, 1989.

[18] Steve Renals and Richard Rohwer. Phoneme classification experiments using radial basis functions. In *Proceedings International Joint Conference on Neural Networks*, volume I, pages 461–468, Washington DC, 1989.

[19] Steve Renals and Richard Rohwer. A study of network dynamics. *Journal of Statistical Physics*, 58:825–847, 1990.

[20] R. Rohwer. The 'moving targets' training algorithm. In L. B. Almeida and C. J. Wellekens, editors, *Lecture Notes in Computer Science 412, Neural Networks*, pages 100–109. Springer-Verlag, Berlin, 1990.

[21] R. Rohwer. The 'moving targets' training algorithm. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 558–565, San Mateo CA, 1990. Morgan Kaufmann.

[22] R. Rohwer. The 'moving targets' training algorithm. In J. Kindermann and A. Linden, editors, *Distributed Adaptive Information Processing (DANIP)*, pages 175–196, Munich, 1990. R. Oldenbourg Verlag.

[23] 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 Procressing*, volume 1, pages 318–362. MIT Press, Cambridge MA, 1986.

[24] Jurgen H. Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report FKI-126-90, Technische Universitat Munchen, Institut fur Informatik, Arcisstr. 21, 8000, Munchen 2, Germany, 1990.

[25] R. S. Sutton and A. G. Barto. An adaptive network that constructs and uses an internal model of its environment. *Cognition and Brain Theory Quarterly*, 4:217–246, 1981.

[26] Richard Sutton. Integrated architectures for learning, planning, and reacting based on approxmating dynamic programming. In *Proc. 7th International Conference on Machine Learning*, 1990.

[27] Andreas S. Weigend, Bernardo A. Huberman, and David E. Rumelhart. Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1:193–209, 1990.

[28] R. Williams. Towards a theory of reinforcement-learning connectionist systems. Technical Report NU-CCS-88-3, College of Computer Science, Northeastern University, Boston, MA, 1988.