Dynamic Reinforcement Driven Error Propagation Networks with Application to Game Playing

Tony Robinson and Frank Fallside
Cambridge University Engineering Department,
Trumpington Street, Cambridge, England.
(ajr@dsl.eng.cam.ac.uk)

ABSTRACT

This paper discusses the problem of the reinforcement driven learning of a response to a time varying sequence. The problem has three parts: the adaptation of internal parameters to model complex mappings; the ability of the architecture to represent time varying input; and the problem of credit assignment with unknown delays between the input, output and reinforcement signals. The method developed in this paper is based on a connectionist network trained using the error propagation algorithm with internal feedback. The network is viewed both as a context dependent predictor of the reinforcement signal and as a means of temporal credit assignment. Several architectures for these networks are discussed and insight into the implementation problems is gained by an application to the game of noughts and crosses.

INTRODUCTION

Of the three major types of learning: supervised; reinforcement driven; and unsupervised; it is reinforcement driven learning which is most applicable to to the formulation of models of animal behaviour and to the design of 'intelligent' machines which must operate with no a priori knowledge of their environment. Under this scheme a model is presented with a time varying input and it generates a time varying output. Feedback for adaptation comes from a single scalar which measures the past performance of the model. In this paper is it assumed that there is a maximum frequency at which these signals change and so the signals may be sampled at a constant rate without loss of information. It is also assumed that the input and output signals have a fixed dimensionality, and can therefore be represented by a sequence of vectors. These are common assumptions in the field of digital signal processing.

Connectionist reinforcement driven learning of arbitrary functions has three main prerequisites:

- The computational power of the model must be sufficient to represent the desired mapping and a suitable learning algorithm must exist. Models with restricted computational power, such as the linear mapping of the input space to the output space, are insufficient for learning complex tasks. However, the class of non-linear functions

known as error propagation networks or multi-layer perceptrons [Rumelhart *et al.*, 1986] have the power to represent arbitrary mappings, and the parameters in these models may be trained using the technique of gradient descent.

- The model must have the capacity for sequence recognition and generation, and for a self-contained system the storage must be provided internally. Within the framework of error propagation networks there are many candidates. The first recurrent error propagation network was formulated by Rumelhart, Hinton and Williams [1986] and employs full connection from all units to the next time frame using an external buffer to store past activations during training. Partial connectivity and partial feedback of the error signal has been presented by Jordan [1986] and Robinson and Fallside [1987a] which has the advantage that no external buffer is needed. Full feedback of the error signal without an external buffer but with considerably increased internal storage has been formulated by Robinson and Fallside [1987a] and implemented by Williams and Zipser [1988].

- Finally, a mechanism is needed for credit assignment, that is which elements of the output vector and at what delay are to be assigned the credit for the reinforcement signal. Sutton and Barto have provided a means for credit assignment in a single node network by applying a first order filter to the input and defining the error signal as the difference of successive outputs [Sutton and Barto, 1981] or as the difference of successive predictions of future reinforcement [Sutton and Barto, 1987]. Barto, Sutton and Anderson [1983] developed a two node network where the function of the second node is to assign credit to the output of the first node. Sutton [1984] evaluates several of these models for temporal credit assignment. In the case where the reinforcement signal is a complex function of the input and output signal an error propagation network may be used to learn the required mapping. Munro [1987] and Jordan [1988] have both trained a network in this manner by presenting random pairs of inputs and outputs.

Whilst previous work has incorporated two of these aspects, it is the aim of this paper to combine all three. The following description assumes familiarity with error propagation networks, begins by describing feedback within these networks and proceeds to the joining of two networks so that they may be trained with a reinforcement signal. Two architectures for this type of network are given, and one of these is applied to the game of noughts and crosses.

ARCHITECTURES

There are many approaches to the architecture and training of error propagation networks with feedback. Common to all of these is that three distinct vectors can be identified: the input vector, $u(t)$; a state vector, $x(t)$; and an output vector, $y(t)$. The vectors $u(t-1)$ and $x(t-1)$ are used as input to an error propagation net whose output is $y(t)$ and $x(t)$, as in figure 1. In some models the state vector has common elements with the output vector or the vector of hidden unit activations, but this paper will consider the general case where the only necessary relationship is though the mapping made by the network.

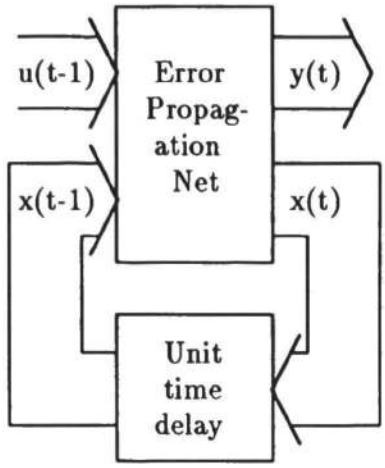

Figure 1: The Dynamic Net

The error propagation network is deliberately shown as a 'black box'. All that is required of the non-linear function contained within it is that given the partial derivative of the cost function (or 'energy') with respect to the value of each element of the vectors $y(t)$ and $x(t)$, it is possible to calculate the same derivative with respect to each element of the vectors $u(t-1)$ and $x(t-1)$ and also with respect to every parameter (or 'weight') within the network. Whilst it is most common to populate the network with nodes which compute a weighted sum of their input vector and pass this through a sigmoidal non-linearity, many other node types are also possible [Robinson, 1989].

A reinforcement driven dynamic net can be formed from two such dynamic nets and this architecture is given in figure 2. The first dynamic net (net Y) computes the overall output of the system from the sequence of input vectors. This net corresponds to the 'Associative Search Element' of Sutton and Barto. After training the complete behaviour of the network is specified by this network alone. However, during training a second dynamic net (net Z) is used for credit assignment of the reinforcement signal. This network corresponds to the 'Adaptive Critic Element' of Sutton and Barto. The function of the second net is to compute the expected reinforcement by modelling the behaviour of the environment in which the first net is placed (including of course the effect of the first net on the environment).

Unlike the two phase training schemes presented by Munro and Jordan, in this architecture both networks are trained at the same time. This is done by formulating the problem in terms of a single cost function which is a linear combination of two quantities: the expected squared difference between the prediced reinforcement signal and the observed reinforcement signal; and the expected squared difference between the prediced reinforcement signal and the desired reinforcement signal. Thus the cost function is minimised if the model can accurately predict the reinforcement signal, and that this reinforcement signal is close to the desired reinforcement (assumed to be the 'high' state).

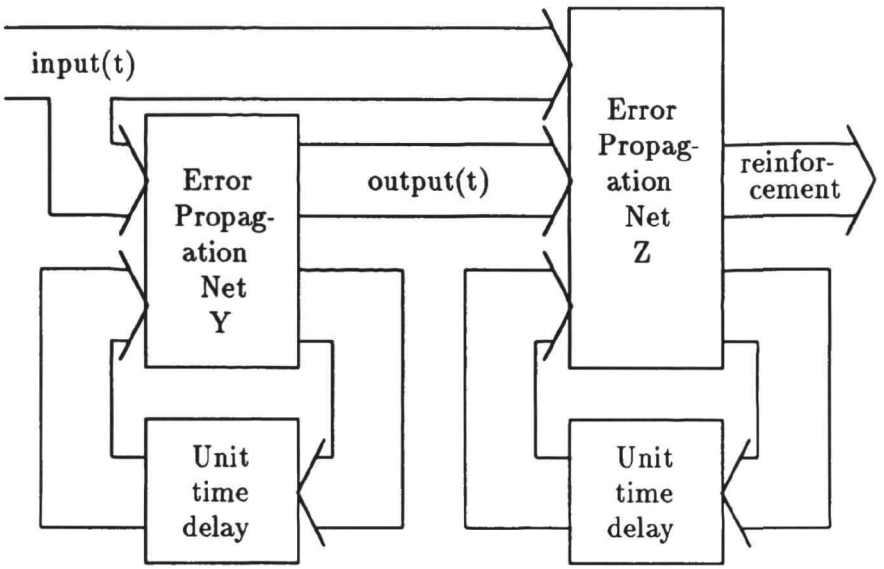

Figure 2: The Reinforcement Driven Dynamic Net

The training may be achieved in two passes for each input vector. In one pass the prediced reinforcement signal is compared with the observed reinforcement signal to calculate the derivative of the cost function with respect to the observed reinforcement output. This derivative signal is propagated back through net Z for all previous times that have influence on the observed reinforcement output and the corresponding derivative of the weights in this net is calculated. In the other pass the prediced reinforcement signal is compared with the desired reinforcement signal and this signal is propagated back through both nets and the derivative with respect to the weights in net Y is calculated. As with standard error propagation networks the derivatives may be used immediately to update the weights in a stochastic gradient descent, or alternatively the weights maybe changed after every pass through the complete training set.

In practice the two nets may be combined to achieve a more compact net, as in figure 3. This is desirable as the back-propagation of errors is a linear process and so the two error signals may be combined and propagated back as a single signal, so reducing the computation. A simple example of this network has previously been presented [Robinson and Fallside, 1987b] in which the net received a high reinforcement if the output had the same sign as the previous input, otherwise the reinforcement was low. Thus it has already been demonstrated that these networks can model a unit time delay.

A GENERAL GAME PLAYING PROGRAM

A subclass of the general net outlined above can be used for game playing, in which case the reinforcement signal is defined only at the end of the game. The technique adopted here is to play a game using the network of figure 2, storing the intermediate activations of all units. At the end of the game two separate computations are performed, one to make a more accurate

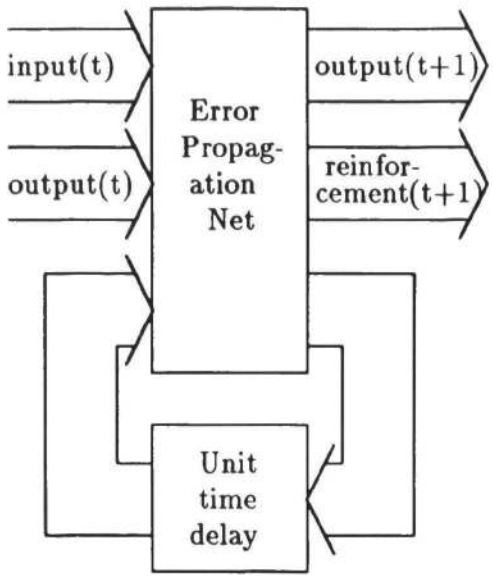

Figure 3: A Compact Reinforcement Driven Dynamic Net

prediction of the reinforcement signal the end of the game, and the second to bias this signal to the high reinforcement state.

NOUGHTS AND CROSSES

The game of 'noughts and crosses' or 'tic tac toe' was chosen for several reasons:

- It is well known and regarded as a 'simple' childrens game. However, it may be classified as 'difficult' when judged by the current standard of connectionist learning procedures.
- The mapping of a board position onto the optimal next move is a complex non-linear function requiring the learning of disjoint pattern classes.
- The state of the game is uniquely defined by the board position, so that net Y does not need any state units.
- The board may be represented in relatively few bits. Each of the nine locations may be unoccupied or occupied by either a 'O' or a 'X'. Thus an upper bound on the number of legal states is $3^9 = 19683$, which can be represented in 15 bits.
- The game has a short duration as no player may place more than five pieces on the board. Thus as far as assigning credit or blame for the outcome of the game, the error signal must be propagated back through a maximum of five states.

The 'opponent' to the net was a simple algorithm that would win by completing a line of two if possible, otherwise a piece would be placed randomly. If the net places pieces randomly, as is

the case before any learning, then the net wins about 30% of the games which are not drawn. A suitably experienced player would never loose and only occasionally draw against this algorithm.

IMPLEMENTATION

Two 3x3 matrices were used to represent the board position. One has each element set high if the corresponding board position is occupied, the other is used to record the owner of the piece placed on the occupied site. The output representation was another 3x3 matrix, the legal move with the largest value in this matrix (after the addition of noise as discussed later) was taken as the move to be made. No attempt was made to take advantage of the symmetry of the game.

The machine was trained a 65 processor array of T800 transputers running at about 50 Mflops. Forty games were played per processor per update, so the weights were updated on gradient information collected over 2600 games. Each net had 144 hidden units with a sigmoidal activation function ranging from -1 to $+1$. The target values for the outputs were chosen to be in the linear region of the activation function, $+0.1$ for positive reinforcement and -0.1 for negative reinforcement.

PROBLEMS

Reinforcement driven learning is a harder task than supervised learning for the simple reason that less information is provided about the desired output. For the game playing program presented here there is the additional problem that changes to the weights change the response given to early moves, so the whole style of the game can change. For example, the initial moves are random, so the prediction and maximisation of the reinforcement signal is carried out for nearly fully populated boards of randomly placed pieces. However, towards the end of the learning period the game length has become shorter and there are correspondingly fewer pieces on the board. Thus the prediction and maximisation functions must relearn for this new set of training data. Because the form of the training set is dependent on the current performance, the net does not perform a gradient descent in a single function throughout the training, but performs a gradient descent in a continually changing function. Thus there is no guarantee of convergence or stability.

The algorithm used as an opponent to the net employed a random number generator to pick a legal move if it could not place a piece to win the game. This randomness means that some responses would be given more often than others in an unpredictable way and this hinders the learning by the introduction of noise into the error signal.

A strict pick-the-biggest rule to convert the output of the net into a symbolic form was found to lead to unstable behaviour during training. This is because the magnitude of the difference between the largest and second largest element is unimportant which results in a discontinuity in the weight space. For example, a small difference of activity in an output unit might change the move made during a game, and change the outcome of the game. So, for the same reason as step activation functions can not be used within a network, a step response in interpreting the

output must be avoided. A probabilistic representation of the the output vector was used to improve the stability. This was implemented by adding random noise to the output vector before choosing the largest element. The noise was generated by the difference of two random numbers with range 0.1. Thus if one output was more than 0.2 above all the others this noise has no effect, otherwise the noise results in random decisions which, when averaged, blur out the discontinuities in the weight space. An alternative deterministic solution to this problem has been proposed by Boothroyd [1989, personal communication] in which a connectionist net is used to warp the output space closer to the form expected by the pick-the-biggest rule. Some preliminary investigations have been reported by Robinson [Robinson, 1989]. In a 'real world' environment, such as that of autonomous robot control, an analogue output may be appropriate and this would avoid the problem.

RESULTS

The initial performance of the net was to win about 30% of the games played. After playing 300,000 games this figure improved to 59%, and playing a further 3,000,000 games produced no further improvement. Whilst the performance of the net is lower than the optimal performance, the net did learn sufficiently to perform better than the opponent algorithm which it played against.

CONCLUSION

This paper has presented a scheme for implementing reinforcement driven learning for arbitrary sequences of input and output vectors. Three necessary conditions have been identified: the ability to make arbitrary mappings; the ability to store contextual information; and the ability to do credit assignment. This approach has used error propagation networks for the mappings, feedback of state information to provide context and a new cost function to perform credit assignment. The new cost function is a linear sum of that required to form a good predictor of the reinforcement signal and that required to maximise the reinforcement signal.

A dynamic reinforcement driven error propagation network has been applied to the game of noughts and crosses. The final performance was slightly better than the opponent algorithm but lower than the optimal performance. This has raised issues related to changing environmental conditions during training, statistical fluctuations in gradient descent techniques and the interfacing of a distributed machine to a symbolic environment.

ACKNOWLEDGEMENTS

One of the authors, Tony Robinson, would like to acknowledge financial support from the UK Science and Engineering Research Council, Cambridge University Engineering Department and Trinity Hall, Cambridge. Technical support was received from the ParSiFal project IKBS/146 which developed the transputer array, and considerable academic support was received from all members of the connectionist group in Cambridge University Engineering Department.

REFERENCES

[Barto *et al.*, 1983] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):834–846, 1983.

[Boothroyd, 1989] C. B. Boothroyd. January 1989. Department of Material Science, Cambridge University. Personal communication.

[Jordan, 1986] Micheal I. Jordan. *Serial Order: A Parallel Distributed Processing Approach.* ICS Report 8604, Institute for Cognitive Science, University of California, San Diego, May 1986.

[Jordan, 1988] Michael I. Jordan. *Supervised learning and systems with excess degrees of freedom.* COINS Technical Report 88-27, Massachusetts Institute of Technology, May 1988.

[Munro, 1987] P. W. Munro. A dual back-propagation scheme for scalar reinforcement learning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, 1987.

[Robinson, 1989] A. J. Robinson. *Dynamic Error Propagation Networks.* PhD thesis, Cambridge University Engineering Department, February 1989.

[Robinson and Fallside, 1987a] A. J. Robinson and F. Fallside. Static and dynamic error propagation networks with application to speech coding. In Dana Z. Anderson, editor, *Proceedings of Neural Information Processing Systems*, American Institute of Physics, Denver, November 1987.

[Robinson and Fallside, 1987b] A. J. Robinson and F. Fallside. *The Utility Driven Dynamic Error Propagation Network.* Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, 1987.

[Rumelhart *et al.*, 1986] 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. I: Foundations.*, chapter 8, Bradford Books/MIT Press, Cambridge, MA, 1986.

[Sutton, 1984] Richard S. Sutton. *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts, Department of Computer and Information Science, February 1984.

[Sutton and Barto, 1981] Richard S. Sutton and Andrew G. Barto. An adaptive network that constructs and uses and internal model of its world. *Cognition and Brain Theory*, 4(3):217–246, 1981.

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

[Williams and Zipser, 1988] R. J. Williams and D. Zipser. *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks.* ICS Report 8805, Institute for Cognitive Science, University of California, San Diego, October 1988.