A NEW HYBRID CRITIC-TRAINING METHOD FOR APPROXIMATE DYNAMIC PROGRAMMING

Thaddeus T. Shannon,
George G. Lendaris[1]
Northwest Computational Intelligence Laboratory
& Systems Science Ph.D. Program,
Portland State University
P.O. Box 751, Portland, OR 97207

ABSTRACT

A variety of methods for developing quasi-optimal intelligent control systems using reinforcement learning techniques based on adaptive critics have appeared in recent years. This paper reviews the family of approximate dynamic programming techniques based on adaptive critic methods and introduces a new hybrid critic training method.

Keywords: Adaptive Control, Approximate Dynamic Programming, Adaptive Critic Methods, Neural Networks

INTRODUCTION

A variety of Adaptive Critic Design techniques for training neuro-controllers have appeared in the literature recently, falling into model-based methods such as Dual Heuristic Programming (DHP), and non-model-based methods such as Action Dependent Heuristic Dynamic Programming (ADHDP) or Q-learning (Barto, et al. 1983, Werbos 1990, 1992, Santiago & Werbos, 1994, Prokhorov, Santiago & Wunsch 1995, Prokhorov & Wunsch 1997). The DHP method has been shown to be much more efficient for neuro-controller training and to produce superior designs to the non-model-based methods. However its implementation relies on having an explicit differentiable model of the plant's dynamics and the critic function implicitly estimated by the method is not guaranteed to be integrable.

This paper reviews the family of approximate dynamic programming techniques based on adaptive critic methods and introduces a new hybrid critic training method that guarantees the integrability of the critic function while capturing the performance of the model based training method. The performance of the hybrid method is compared to the HDP and DHP methods on a highly nonlinear multivariable discrete time benchmark problem proposed by Narendra.

[1] This work was supported by the National Science Foundation under grant ECS-9904378.

The first section provides a brief overview of adaptive critic based approximate dynamic programming and then delves into the details of the Dual Heuristic Programming (DHP) technique. The second section introduces an alternative classification of critic techniques based on the role of system models in the training process. In the following section we review previous results on the use of partial, approximate and qualitative models for critic based controller training. A new hybrid critic training method is then proposed, that incorporates the more efficient and accurate model based critic training rule from the DHP method with an HDP based critic architecture that insures the integrability of the critic function. We describe the specifics of this critic's estimation and in the final section compare its performance to that of a DHP critic.

APPROXIMATE DYNAMIC PROGRAMMING

Dynamic Programming is a general approach for sequential optimization applicable under very broad conditions. Fundamental to this approach is Bellman's Principle of Optimality (Bellman 1957): that an optimal trajectory has the property that no matter how an intermediate point is reached, the rest of the trajectory must coincide with an optimal trajectory as calculated with the intermediate point as the starting point. This principle is applied by formulating a "primary" utility function *U(t)* that embodies a control objective for a particular context in one or more measurable variables. A *secondary* utility function is then formed

$$J(t) = \mathop{}\limits_{k=0}^{\infty} \gamma^k U(t+k),$$

which embodies the desired control objective through time. This is Bellman's equation, and the point of Dynamic Programming is to select the sequence of actions (controls) that maximize or minimize *J(t)*. Unfortunately, this optimization is not computationally tractable for most real world problems, thus we are forced to consider potentially tractable approximation methods. A useful identity based on the above equation is the Bellman Recursion

$$J(t) = U(t) + \gamma\, J(t+1).$$

A promising collection of such approximation techniques based on estimating the function *J(t)* using this identity with neural networks as function approximators was proposed by Werbos (Werbos, 1990, 1992). These networks are often called Adaptive Critics, though this term can be applied more generally to any network that provides learning reinforcement to another entity (Widrow et al., 1973). As a practical matter, any computational structure capable of acting as a universal function approximator can be used in this role (i.e. neural networks, fuzzy rule structures, etc.). The gradient of the estimated *J(t)* can then be used to train or tune a controller. Since the gradient is the important aspect for controller training, some techniques use critics that estimate the derivatives of *J(t)* instead of the function value itself.

The standard classification of these adaptive critic methods is based on the critic's inputs and outputs. In Heuristic Dynamic Programming (HDP) the critic's outputs are estimates of the value of *J(t)*. In Dual Heuristic Programming (DHP) the critic's outputs are estimates of the derivatives of *J(t)*. In the *action dependent* versions of HDP and DHP, the critic's inputs are augmented with the controller's output (action), hence ADHDP and ADDHP.

These approaches to approximate dynamic programming utilize at least two distinct training loops, a controller training loop and a critic training loop (Lendaris et al. 1999, Lendaris & Shannon 2000). In the neurocontrol context, the controller training loop adapts a neural network to be an approximately optimal controller. Specifically, the controller is trained to optimize the secondary utility function *J(t)* for the problem context. Since the controller outputs control actions u(t), a gradient based learning algorithm requires estimates of the derivatives $\frac{\partial J(t)}{\partial u_i(t)}$ for controller training. The critic is trained based on the consistency of its estimates through time judged using the Bellman Recursion. The exact implicit relationship is a function of the type of critic used and the structure of the primary utility function.

In the DHP method the critic estimates the derivatives of *J(t)* with respect to the system states, i.e. $\lambda_i(t) = \frac{\partial J(t)}{\partial R_i(t)}$. From Bellman's Recursion we have

$$\frac{\partial}{\partial R_i(t)} J(t) = \frac{\partial}{\partial R_i(t)} \left(U(t) + \gamma J(t+1) \right),$$

so the identity used for this critic's training is (in tensor notation)

$$\lambda_i(t) = \frac{\partial U(t)}{\partial R_i(t)} + \frac{\partial U(t)}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial R_i(t)} + \gamma \lambda_k(t+1) \left[\frac{\partial R_k(t+1)}{\partial R_i(t)} + \frac{\partial R_k(t+1)}{\partial u_m(t)} \frac{\partial u_m(t)}{\partial R_i(t)} \right].$$

To evaluate the right hand side of this equation we need a model of the system dynamics that includes all the terms from the Jacobian matrix of the coupled plant-controller system, e.g. $\frac{\partial R_j(t+1)}{\partial R_i(t)}$ and $\frac{\partial R_j(t+1)}{\partial u_i(t)}$.

Controller training then utilizes the chain rule and the system model to translate critic outputs into estimates of $\frac{\partial J(t)}{\partial u_i(t)}$, i.e.

$$\frac{\partial J(t)}{\partial u_k(t)} = \frac{\partial U(t)}{\partial u_k(t)} + \gamma \qquad \lambda_i(t+1) \frac{\partial R_i(t+1)}{\partial u_k(t)}.$$

The entire process can be characterized as a simultaneous optimization problem; gradient based optimization of the critic function approximator together with gradient based optimization of controller parameters based on the *J(t)* estimates obtained from the critic. Different strategies have been utilized to get both these optimizations to converge. A number of authors propose alternating between optimization steps for the critic approximator and optimization of the controller (Santiago & Werbos, 1994, Prokhorov, Santiago & Wunsch 1995, Prokhorov 1997, Prokhorov & Wunsch 1997). In past work we have noted that taking simultaneous steps in both optimization processes does not appear to introduce significant instabilities into the dual convergence problem (Lendaris et al. 1999, Lendaris & Shannon 2000). Since the simultaneous stepping approach is about twice as fast as the alternating approach we recommend its use.

As these techniques rely on gradient based optimization of *J(t)*, they inherently suffer from the problem of (unsatisfactory) local optima. Global optimization of *J(t)* in general is subject to the "No Free Lunch Theorem". What approximate dynamic programming techniques offer is a tractable method for local hill climbing on the *J(t)* landscape of controller parameter space. Initialized at a random point in parameter space, these methods may be trapped by a local optimum at an unsatisfactory control law. We can attempt to avoid this case by applying what ever problem specific knowledge is available a priori to the choice of initial controller parameters, in the hope of being near a satisfactorily high hill (or deep valley).

ADAPTIVE CRITIC DESIGN TECHNIQUES AND THE ROLE OF MODELS

An alternative way of distinguishing adaptive critic methods is to consider the role of system models in the training loops of each method:

HDP
The critic estimates J(t) based on the system state R(t). Critic training is based on the identity

$$J(t) = U(t) + \gamma\ J(t+1),$$

which requires no system model to check. Controller training is based on finding the derivatives of J(t) with respect to the control variables. In HDP we obtain these derivatives through the chain rule $\frac{\partial J(t)}{\partial u_i(t)} = {}_{j=1}^{n} \frac{\partial J(t)}{\partial R_j(t)} \frac{\partial R_j(t)}{\partial u_i(t)}$. We combine estimates of the derivatives of J(t) with respect to the states , obtained via backpropagation through the critic network, with the derivatives of the states with respect to the controls using the chain rule. This final set of derivatives comes from a differentiable model, e.g. an explicit analytic model or a neural model. Thus **HDP uses a model for controller training but not critic training.**

ADHDP (Q-learning)
Training for the critic network is the same as for HDP. Training for the controller is simplified in that the control variables are inputs to the critic, thus derivatives of J(t) with respect to the controls are obtained directly from backpropagation through the critic. Thus **ADHDP uses no models in the training process.**

DHP
Here the critic estimates the derivatives of J(t) with respect to the system states, i.e. $\lambda_i(t) = \frac{\partial J(t)}{\partial R_i(t)}$. The identity used for critic training is the first order Bellman recursion (in tensor notation):

$$\lambda_i(t) = \frac{\partial U(t)}{\partial R_i(t)} + \frac{\partial U(t)}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial R_i(t)} + \lambda_k(t+1)\left[\frac{\partial R_k(t+1)}{\partial R_i(t)} + \frac{\partial R_k(t+1)}{\partial u_m(t)} \frac{\partial u_m(t)}{\partial R_i(t)}\right].$$

To evaluate the right hand side of this equation we need a full model of the system dynamics. This includes all the terms from the Jacobian matrix of the coupled plant-controller system, e.g. $\frac{\partial R_j(t+1)}{\partial R_i(t)}$ and $\frac{\partial R_j(t+1)}{\partial u_i(t)}$.

Controller training is much like in HDP, except that the controller training loop directly utilizes the critic outputs along with the system model. So **DHP uses models for both critic and controller training.**

ADDHP
This methods utilizes the DHP critic training process, but gets the derivatives needed for controller training directly from the critic's output. Therefore **ADDHP uses a model for critic training but not for controller training.**

One of the promising applications for adaptive critic methodologies is in adaptive control contexts for non-stationary plants. In these contexts there will necessarily be a third training loop that updates a differentiable model in an ongoing plant identification process. Any limitations experienced in this on-line adaptation context may derive from an inability to adequately track changes in the plant, rather than from our ability to continuously solve the approximate dynamic programming problem. Thus, understanding the effects of model error on controller training is important.

PARTIAL, NOISY AND QUALITATIVE MODELS

Model based methods such as DHP have been shown to be much more efficient for training neuro-controllers and to produce superior designs to non-model based methods such as Q-learning; however, associated with this is a requirement for an explicit differentiable model. When such a model is not already available, the cost of developing one may well offset the benefits of increased speed and accuracy of this method. Various strategies can be used to reduce the time and effort needed to develop a model, e.g., only estimate a partial model, use a very rough estimation procedure, or attempt to capture only the qualitative behavior of the system. What information must necessarily be included in a model depends on how the model is used in the training process.

Partial Models
By a partial model we mean a model which only explains some of the causal interactions between state variables. A general hypothesis is that a differentiable model that includes any subset of the state variables that would constitute an observable system should be sufficient for DHP training. Such a *model* should be useable for carrying out the DHP process, even in those situations where additional plant states might be observable, and even when such additional state variables are used as inputs to the controller and critic networks.

Noisy Models
Noisy models are ubiquitous in real life. A model based on regression analysis or any other statistical estimation technique is approximate at every operating point. The numerical values derived from such models may be considered "noisy" values in that they are (hopefully) close to the "true" values and if properly estimated, randomly distributed around the true values. We can perform controlled experiments to yield these kinds of approximations by mixing noise with the "true" values derived from a known analytic model. Two kinds of mixing are possible, additive mixing, and multiplicative mixing.

Qualitative Models
Based on the above observations, it is tempting to investigate the use of greatly simplified qualitative models. Such models only determine the sign of the derivatives at each operating point, positive, negative or zero. Since our current experiments are based on plants defined by analytic equations we simply replace all positive derivatives with the value 1 and all negative derivatives with the value -1.

Other authors have hypothesized that qualitative models should be adequate for the pole-cart problem, as there is only a single control variable and straightforward dynamics. Our experience has been that training with such models is generally successful, though usually slower than training with exact models, and is capable of producing controllers of the same quality as when training with exact models.

The size of the positive and negative values used will vary the "gain" of the training process - thus linking this choice to the selection of learning rates for the training process. Larger values in a qualitative model might require smaller learning rates and vice versa.

Another view of a simplified, qualitative model is that we have constructed a classifier of the plant's qualitative behavior. Our classifier tells us whether a particular variable will increase, decrease or remain unchanged based on the value of some other variable. Estimating such a classifier as a practical matter should be simpler than estimating a quantitative model for the plant. All one needs to determine are the class boundaries in the state space. An important question to explore is with what precision must the class boundaries be known for the qualitative model to be useful?

PREVIOUS RESULTS WITH QUALITATIVE MODELS

Previous work using the Pole-Cart problem and Narendra's benchmark problem have shown that the models used in DHP training can be far from perfect and still lead to successful controller training (Shannon 1999). A variety of options exist for simplifying system identification and modeling in the DHP context. Estimation of exact parameter values in system models may not be as important as producing unbiased estimates. In off-line training contexts, the "noise" in these estimated models may even be beneficial in the controller training process, due to its tendency to anneal the critic and controller networks out of unsatisfactory local optima.

As reported in (Shannon 1999, Shannon & Lendaris 1999), our experience has been that training with such (imperfect) models is generally successful, though usually slower than training with exact models, and is capable of producing controllers of the same quality as when training with exact models. Narendra's problem is more interesting than the Pole-Cart problem in this context as it is Multiple-Input-Multiple-Output (MIMO) with significant non-linearity and time delays and requires following an *a priori* unknown reference trajectory.

Further experiments with Narendra's system suggested that one could make do with very rough models. To demonstrate this, we took the earlier developed qualitative model and introduced a substantial "don't know" zone between classes. Results with these models were comparable to those using the exact class boundaries, with equivalent RMS error on the test trajectory and on the best performance. This showed that successful training is possible using only qualitative information for those regions of state space in which the plant's dynamics are unambiguous.

A HYBRID ADAPTIVE CRITIC

Based on our experiences using simplified system models with model based techniques, we believe that the trade off between the efficiency and accuracy gains of model based critic training methods and the cost of developing simple system models tilts heavily towards using model based methods whenever possible. Our proposed critic training method allows a scalar valued (HDP type) critic to be trained using the first order Bellman recursion normally used for DHP critic training. This gives us a real, finite valued *J* function approximation together with the more accurate derivative estimates associated with the DHP method. The critic is trained to make consistent λ estimates. As these estimates are obtained from the critic via backpropagation, the weight update process is based on "propagating" error signals from the network's inputs rather than from the network's outputs.

Hybrid Training Equations

A single hidden layer, feed-forward neural network's output can be expressed as

$$y = f\left(a + \sum_j \alpha_j g_j\left(b_j + \sum_i \beta_{j,i} x_i\right)\right),$$

where *f()* is the output layer activation function and *g()* the hidden layer activation function. The partial derivatives of the output are then

$$\frac{\partial y}{\partial x_i} = f' \bullet \left[\sum_j \alpha_j g'_j \beta_{ji} \right],$$

which are easily evaluated via backpropagation. Instead of adjusting the network weights to minimize output error, we instead minimize the error for the network's partial derivatives. The update equations are given by

$$\frac{\partial}{\partial a}\left(\frac{\partial y}{\partial x_i}\right) = \frac{\partial f'}{\partial a} \bullet \left(\sum_j \alpha_j g'_j \beta_{j,i} \right),$$

$$\frac{\partial}{\partial \alpha_J}\left(\frac{\partial y}{\partial x_i}\right) = \frac{\partial f'}{\partial \alpha_J} \bullet \left(\sum_j \alpha_j g'_j \beta_{j,i} \right) + f' g'_J \beta_{J,i},$$

$$\frac{\partial}{\partial b_J}\left(\frac{\partial y}{\partial x_i}\right) = \frac{\partial f'}{\partial b_J} \bullet \left(\sum_j \alpha_j g'_j \beta_{j,i} \right) + f' \alpha_J \frac{\partial g'_J}{\partial b_J} \beta_{J,i}, \text{ and}$$

$$\frac{\partial}{\partial \beta_{J,i}}\left(\frac{\partial y}{\partial x_i}\right) = \frac{\partial f'}{\partial \beta_{J,i}} \bullet \left(\sum_j \alpha_j g'_j \beta_{j,i} \right) + \left[\sum_j \alpha_j \left(\frac{\partial g'_j}{\partial \beta_{J,i}} \beta_{j,i} + g'_j \frac{\partial \beta_{j,i}}{\partial \beta_{J,i}} \right) \right] \bullet f',$$

where

$$\frac{\partial g'_J}{\partial b_J} = g''_J, \qquad \frac{\partial f'}{\partial a} = f'', \qquad \frac{\partial g'_J}{\partial \beta_{J,i}} = g''_J x_i,$$

$$\frac{\partial f'}{\partial \alpha_J} = f'' \bullet g_J \left(b_J + \sum_i \beta_{J,i} x_i \right), \text{ and}$$

$$\frac{\partial f'}{\partial \beta_{J,i}} = f'' \alpha_J \frac{\partial g_J}{\partial \beta_{J,i}} = f'' \alpha_J g'_J x_i .$$

The critic is trained by generating *J(t)* and *J(t+1)* estimates, computing the associated partial derivatives $\frac{\partial J(t)}{\partial R(t)}$ and $\frac{\partial J(t+1)}{\partial R(t+1)}$ via backpropagation, generating an error by plugging these partial derivatives into the first order Bellman recursion

$$e_i(t) = \left[\frac{\partial U(t)}{\partial R_i(t)} + \frac{\partial U(t)}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial R_i(t)} + \lambda_k(t+1) \left[\frac{\partial R_k(t+1)}{\partial R_i(t)} + \frac{\partial R_k(t+1)}{\partial u_m(t)} \frac{\partial u_m(t)}{\partial R_i(t)} \right] - \lambda_i(t) \right]^2 .$$

The network weights are then adjusted to make the recursion error as small as possible using a delta rule, e.g.

$$\alpha_J(t+1) = \alpha_J(t) + \eta\left[{}_{i} \; \frac{\partial e_i(t)}{\partial \alpha_J(t)} \right] = \alpha_J(t) + \eta\left[{}_{i} \; \frac{\partial \lambda_i(t)}{\partial \alpha_J(t)} \frac{\partial e_i(t)}{\partial \lambda_i(t)} \right].$$

COMPARING HYBRID CRITIC PERFORMANCE TO HDP AND DHP

The Narendra benchmark system (Narendra & Mukhopadhyay 1994) is defined by the state equations:

$$\begin{aligned}
x_1(t+1) &= 0.9x_1(t)\sin[x_2(t)] + \left[2 + 1.5\frac{x_1(t)u_1(t)}{1+x_1^2(t)u_1^2(t)}\right]u_1(t) + \left[x_1(t) + \frac{2x_1(t)}{1+x_1^2(t)}\right]u_2(t), \\
x_2(t+1) &= x_3(t)\left[1+\sin[4x_3(t)]\right] + \frac{x_3(t)}{1+x_3^2(t)}, \\
x_3(t+1) &= \left[3+\sin[2x_1(t)]\right]u_2(t).
\end{aligned}$$

The observable states are often taken to be $x_1(t)$ and $x_2(t)$. The plant is stable at the origin with constant control values. Linearized around the origin, it is controllable, observable and of minimum phase. A standard reference signal for evaluating controller performance is:

$$\begin{aligned}
\tilde{x}_1(t) &= 0.75\sin\left[\frac{2\pi t}{50}\right] + 0.75\sin\left[\frac{2\pi t}{10}\right], \\
\tilde{x}_2(t) &= 0.75\sin\left[\frac{2\pi t}{30}\right] + 0.75\sin\left[\frac{2\pi t}{20}\right].
\end{aligned}$$

Utilizing the reference signal, and recognizing the time delays in the system, we crafted the primary utility function

$$U(t) = \left(x_1(t) - \tilde{x}_1(t)\right)^2 + \left(x_2(t+1) - \tilde{x}_2(t+1)\right)^2.$$

Our basic controller has 5 inputs, $x_1(t)$, $x_2(t)$, $x_3(t)$, $\tilde{x}_1(t+1)$, and $\tilde{x}_2(t+2)$, 6 hidden layer elements and 2 outputs, $u_1(t)$ and $u_2(t)$. The critic network has 4 inputs, $x_1(t)$, $x_2(t+1)$, $\tilde{x}_1(t)$, and $\tilde{x}_2(t+1)$, 6 hidden layer elements, and 2 outputs, $\frac{\partial J(t)}{\partial x_1(t)}$ and $\frac{\partial J(t)}{\partial x_2(t+1)}$. All the processing elements in both networks use hyperbolic tangent activation functions and have bias terms. Training of both networks is performed simultaneously.

Baseline DHP training is carried out using a random reference signal generated by selecting values from the interval [-1.5, 1.5] via a uniform distribution every four time steps. This procedure generates a random, stair-step signal that provides persistent excitation for training. The performance of the trained controller is evaluated after 40,000 training steps using the sinusoidal reference trajectory given above. This evaluation is a generalization test, as the controller never sees a non-random reference trajectory during the training process. Controllers trained by this method vary in performance, with an average RMS error of 0.26 (over at least 100 designs) and with the better controllers producing RMS error of about 0.22.

The HDP method is significantly less effective on this problem. Trained controller performance has twice the tracking error, average RMS error ranging from 0.60 to 0.70 with training requiring upwards of 250,000 steps.

The same HDP critic network trained using the hybrid method produces controller performance identical to the DHP method. The average RMS error of 0.27 obtained for multiple training runs is statistically indistinguishable from the DHP results, while the best controllers obtained from raw training again producing an error of about 0.22. Training time for the hybrid case was 50% longer than for DHP (60,000 steps).

CONCLUSION

Our hybrid critic training method allows us to do model based critic training with a critic that estimates the secondary utility function. Previous model based critic training methods have only been used with critics that estimate the partial derivatives of the secondary utility function. Another feature of adaptive critic based approximate dynamic programming techniques is the potential to use the critic function as a guarantor of system stability, e.g. (Prokhorov 1997, Prokhorov & Feldkamp 1999). This is directly practical if the critic function estimates the secondary utility rather than the derivatives.

Our belief is that adaptive critic techniques are ideally suited for neuro-fuzzy implementations (Shannon & Lendaris 20000). Fuzzy controllers and models more easily allow the incorporation of a priori knowledge, while neural networks may be more natural as function approximators in the critic role. It is also important to notice the applicability of these techniques to adaptive control problems. For non-stationary plants, the controller, critic and model can all be continuously update on-line to track changes in the plant's dynamics (Lendaris & Shannon 2000).

REFERENCES

Barto, A.G., R.S. Sutton & C.W. Anderson, (1983), "Neuron-like Adaptive Elements That Can Solve Difficult Learning Control Problems", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, pp. 834-846.

Bellman, R.E., (1957), *Dynamic Programming*, Princeton University Press.

Lendaris, G. & T. Shannon, (2000), "Designing (Approximate) Optimal Controllers Via DHP Adaptive Critics & Neural Networks", in *The Handbook of Applied Computational Intelligence*, Karayiannis, Padgett & Zadeh, eds., CRC Press.

Lendaris, G.G., T.T. Shannon & A. Rustan, (1999), "A Comparison of Training Algorithms for DHP Adaptive Critic Neuro-Control", in *Proceedings IJCNN'99*, Washington D.C., IEEE.

Narendra, K. & S. Mukhopadhyay, (1994), "Adaptive Control of Nonlinear Multivariable Systems Using Neural Networks", *Neural Networks*, vol. 7, #5, pp. 737-752.

Prokhorov, D., (1997), *Adaptive Critic Designs and their Application*, Ph.D. Dissertation, Department of Electrical Engineering, Texas Tech University.

Prokhorov, D. & D. Wunsch, (1997), "Adaptive Critic Designs", *IEEE Transactions on Neural Networks*, vol. 8 (5), pp. 997-1007.

Prokhorov, D., R. Santiago & D. Wunsch, (1995), "Adaptive Critic Designs: A Case Study for Neurocontrol", *Neural Networks*, vol. 8 (9), pp. 1367 - 1372.

Prokhorov, D.V. & L.A. Feldkamp, (1999), "Generalized Adaptive Critics and their Applications", presented at IJCNN'99, session 6.5, Washington D.C..

Santiago, R. & P.J. Werbos, (1994), "New Progress Towards Truly Brain-Like Control", *Proceedings of WCNN'94*, San Diego, CA, pp. 27-33.

Shannon T.T., (1999), "Partial, Noisy and Qualitative Models for Adaptive Critic Based Neurocontrol", in *Proceedings of IJCNN'99*, Washington D.C., IEEE.

Shannon T.T., G.G. Lendaris, (1999), "Qualitative Models for Adaptive Critic Neurocontrol", in *Proceedings of SMC'99*, IEEE.

Shannon T.T., G.G. Lendaris, (2000), "Adaptive Critic Based Approximate Dynamic Programming for Tuning Fuzzy Controllers", in *Proceedings of IEEE-FUZZ 2000*, IEEE.

Werbos, P.J., (1990), "A Menu of Designs for Reinforcement Learning Over Time", in Miller, W.T., R.S. Sutton, & P.J. Werbos eds., *Neural Networks for Control*, MIT Press, Cambridge, MA, pp. 67- 95.

Werbos, P.J., (1992), "Approximate Dynamic Programming for Real-Time Control and Neural Modeling", in D.A. White & D.A. Sofge eds., *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, Van Nostrand Reinhold, New York, pp. 493 - 525.

Werbos, P.J., (2000), "Stable Adaptive Control Using New Critic Designs", in *The Handbook of Applied Computational Intelligence*, Karayiannis, Padgett & Zadeh, eds., CRC Press.

Widrow, B., N Gupta & S. Maitra, (1973), "Punish/Reward: Learning with a Critic in Adaptive Threshold Systems", *IEEE Transactions on Systems, Man & Cybernetics*, vol. 3 (5), pp. 455-465.