01 Jan 1995

Adaptive Critic Based Neural Networks for Control (Low Order System Applications)

S. N. Balakrishnan

Missouri University of Science and Technology, bala@mst.edu

Victor Biega

Recommended Citation

S. N. Balakrishnan and V. Biega, "Adaptive Critic Based Neural Networks for Control (Low Order System Applications)," *Proceedings of the 1995 American Control Conference*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1995.

The definitive version is available at https://doi.org/10.1109/ACC.1995.529265

Proceedings of the
American Control Conference
Seattle, Washington • June 1995

ADAPTIVE CRITIC BASED NEURAL NETWORKS FOR CONTROL*
(Low Order System Applications)

S. N. Balakrishnan[1] and Victor Biega[2]
Department of Mechanical and Aerospace Engineering
and Engineering Mechanics
University of Missouri-Rolla
Rolla, MO 65401-0249
bala@umr.edu

Abstract

Dynamic Programming is an exact method of determining optimal control for a discretized system. Unfortunately, for nonlinear systems the computations necessary with this method become prohibitive. This study investigates the use of adaptive neural networks that utilize dynamic programming methodology to develop near optimal control laws. First, a one dimensional infinite horizon problem is examined. Problems involving cost functions with final state constraints are considered for one dimensional linear and nonlinear systems. A two dimensional linear problem is also investigated. In addition to these examples, an example of the corrective capabilities of critics is shown. Synthesis of the networks in this study needs no external training; they do not need any apriori knowledge of the functional form of control. Comparison with specific optimal control techniques show that the networks yield optimal control over the entire range of training.

I. Introduction

Optimization is a primary concern in most real world processes. Simple methods, such as that of linear quadratic regulators, can only be used to solve control mappings for infinite time linear problems. Typically there are two methods of solving nonlinear and finite time problems, two point boundary value problem (TPBVP) methods and Dynamic Programming. Each of these, however, has limitations.

TPBVP methods provide exact solutions, but sometimes they may be very difficult to solve for with nonlinear systems. In addition, the TPBVP methodology must be solved for each set of initial conditions. This requires determining a separate solution for each possible initial condition for a given system. Dynamic programming is, usually, an exact method of determining optimal control. This method of solution becomes very difficult to solve for in higher dimension and nonlinear systems.

Other methods of solution also have their advantages and disadvantages. Neighboring optimal control is beneficial in that the solution of a single TPBVP allows an approximate solution over a range of initial conditions. The disadvantage is that it can fail at a distance from the original TPBVP solution. Several authors have used neural networks to "optimally" solve nonlinear systems [1-4].

The method discussed in this study determines an optimal control law for a system by successively adapting two networks, an action and a critic network. This method determines the control law for an entire range of initial conditions. In addition the control law does not need to be determined mathematically. This method simultaneously determines and adapts the neural networks to the optimal control policy for both linear and nonlinear systems. In addition, it is important to know that the form of control does not need to be known in order to use this method.

II. Solution Method Development

A. Neural Network Background

Neural networks, or in the case of this study multi-layer perceptrons (MLP's), are known for their ability to model any mapping from input to output given a correctly chosen network structure. They are also able to adapt to new sets of input output pairs. This makes them ideal in adapting to an optimal control policy. For the problems in this study, we will be using MLP. The activation functions used are

$$f_1=\frac{2}{1+e^{(-net)}}-1 \quad f_2=0.5(f_1+1) \quad f_3=net \tag{1}$$

Assuming that there is some function to be minimized, it is then possible to adjust the weights of the MLP to model the appropriate mapping

[1] Professor
[2] Graduate Student
[*] Patent Application in Progress

using a standard gradient descent algorithm.

B. Problem Formulation

In this study, problems of the form

$$J=\phi(x(t_f))+\int_0^{t_f}\psi(x(\tau),u(\tau))\,d\tau \tag{2}$$

$$\dot{x}=f(x,u) \tag{3}$$

are being considered (t_f and x_o are assumed given). The first step taken is to discretize them into the form

$$J=\phi_D(x(N))+\sum_{k=0}^{N-1}\psi_D(x(k),u(k)) \tag{4}$$

$$x(k+1)=f_D(x(k),u(k)) \tag{5}$$

We assume N is known. The method which will be investigated in this study has advantages over the previous methods in that solutions are found over any user specified range of x, and these solutions are then available for the entire span of x. In addition, the user need not assume any predetermined form or function for the control law.

C. Dynamic Programming Background (Exact Results)

The cost function in Eq. 4 can be written as

$$J(x(t))=U(x(t),u(x(t)))+\langle J(x(t+1))\rangle \tag{6}$$

Here, J(x(t)) is the cost associated with going from time t to the final time. U(x(t),u(x(t))) is the utility, which is the cost from going from time t to time t+1. Finally, <J(x(t+1))> is assumed to be the minimum cost associated with going from time t+1 to the final time.

If both sides of the equation are differentiated and we define

$$\lambda(x(t))\equiv\frac{\delta J(x(t))}{\delta x(t)} \tag{7}$$

then

$$\lambda(x(t))=\frac{\delta U(x(t),u(t))}{\delta x(t)}+\frac{\delta U(x(t),u(t))}{\delta u(t)}\frac{\delta u(x(t))}{\delta x(t)}$$

$$+\left\langle\lambda(x(t+1))\frac{\delta x(t+1)}{\delta x(t)}\right\rangle+\left\langle\lambda(x(t+1))\frac{\delta x(t+1)}{\delta u(t)}\frac{\delta u(x(t))}{\delta x(t)}\right\rangle \tag{8}$$

From this it can be seen that if <λ(x(t+1))>, U(x(t),u(t)) and the system model derivatives are known then λ(x(t)) can be found.

Next, the optimality equation is defined as

$$\frac{\delta J(x(t))}{\delta u(t)}=0 \tag{9}$$

Dynamic programming uses these equation to aid in solving an infinite horizon policy or to determine the control policy for a finite horizon problem.

D. Training Methods (Approximation Techniques)

As mentioned earlier, this study uses Eq. 8 in order to determine the optimal control policy. The basic training takes place in two stages, the training of the action network (the network modeling u(x(t))) and the training of the critic network (the network modeling, or approximating λ(x(t))). Both networks are assumed to be feedforward MLP's. Training of the action network can be described by the diagram shown in Figure 1.

To train the action network for time step t, first x(t) is randomized and the action network outputs u(t). The system model is then used to find x(t+1) and (δx(t+1))/(δu(t)). Next, the critic from t+1 is used to find λ(x(t+1)). This information is used to update the action network. This process is continued until a predetermined level of convergence is reached.

To train the critic network for the time step t, x(t) is randomized and the output of the critic λ(x(t)) is found. The action network from step t calculates u(t) and (δu(t))/(δx(t)). The model is then used to find (δx(t+1))/(δx(t)), (δx(t+1))/(δu(t)) and x(t+1). The critic from step t+1 is then used to find λ(x(t+1)). After this, Eq. 8 is used to find λ^*(x(t)), the target value for the critic. This process is continued until a predetermined level of convergence is reached.

III. Applications

In this section of the study, four specific examples will be dealt with. The first of these is an infinite horizon one dimensional linear problem. The second of these is a finite horizon one dimensional problem. Next a finite horizon nonlinear one dimensional problem is investigated. Finally this method is applied to finding the optimal control policy for a two dimensional linear finite horizon problem. In addition to these examples, an example of the corrective capabilities of the critics that have been developed is shown.

A. First Application (Infinite Time 1-D Linear)

The first application deals with a problem

$$x(t+1)=x(t)+2u(t) \tag{10}$$

and a cost function of the form

$$J=\sum_{t=0}^{\infty}[x^2(t)+u^2(t)] \tag{11}$$

As a first step in the solution, any stabilizing controller is defined. In the case of this problem, the initial control will be defined as

$$u(t)=-0.2x(t) \tag{12}$$

Next, a neural network is designed and the initial weights of this network are randomized. This network functions as the adaptive critic.

For this infinite horizon problem the cost associated with state x(t) at time t should be equal to the cost associated with state x(t) at time t+1, therefore a single critic can be used to calculate both λ(x(t)) and λ(x(t+1)). Defining U(x(t),u(t)) as

$$U(u(t),x(t))=x^2(t)+u^2(t) \tag{13}$$

allows us to obtain the derivatives of the utility function. This, in combination with the critic outputs and the system model derivatives, allows the use of Eq. 8 to determine the target value for the critic λ^*(x(t)). This target value is calculated for random values of x(t) until the critic network converges.

After the critic converges, a new neural network is initialized to act as the action network. For this problem a neural network with two hidden layers and three neurons per layer is chosen. The action network is then trained using a gradient descent algorithm. After the action network converges, the critic is again trained using the new action network. (Note that the weights of the critic are not randomized. Instead, the weights from the previous critic are used as the initial weights.) This process is repeated until optimal control has been reached.

Figure 2 shows the action network output used in the problem as well as the optimal control determined from the ricatti equation and the initial control. Notice that as the action network is refined it converges to the optimal solution. Figure 3 shows the critic network output for the infinite horizon problem. Notice that once again as the critic network is refined, it converges to the optimal value for the critic. Figure 4 shows a comparison of the system state being controlled by both the optimal control and the control determined by this adaptive critic based method for x(0)=-20. Note that this initial condition was chosen arbitrarily. The neural network has determined the near optimal control law for each point within its training range.

B. Second Application (Finite Time 1-D Linear)

The second application considers a one-dimensional linear finite horizon problem with a system of the form

$$x(t+1)=0.3679x(t)+0.6321u(t) \tag{14}$$

and a cost function of the form

$$J=\frac{1}{2}[x(10)]^2+\frac{1}{2}\sum_{t=0}^{9}[x^2(t)+u^2(t)] \tag{15}$$

Initial value of x is unity. The first step with this problem is to define the appropriate utility functions. After this, Eq. 8 is used in order to adapt the critics and the action networks. Figure 5 shows the optimal control for step 9, and the adaptive critic determined control for step 9. The critic for step 9 is shown in Figure 6. Steady state control is reached at a few time steps later. Figure 7 shows the application of both the optimal control law and the adaptive critic based control law to the initial condition x(0)=2. Once again, the initial condition is arbitrary. **It could be chosen to be any point within the training range.**

C. Third Application (Finite Time 1-D Nonlinear)

The third application of the adaptive critic based control investigates a one dimensional nonlinear finite horizon problem with a system of the form

$$x(t+1)=x(t)+0.1x^2(t)+0.1u(t) \tag{16}$$

and a cost function of the form

$$J=\frac{1}{2}x^2(10)+\frac{1}{2}\sum_{t=0}^{9}0.1u^2(t) \tag{17}$$

As in the linear problem, a network is first initialized to act as the action network. In this case the network structure contained two hidden layers with four neurons each. After convergence of the action network a new network is initialized for the critic network. The critic network is then trained using Eq. 8. The process of training the action and critic networks is then repeated for the remainder of the time steps.

In order to compare the adaptive critic method with another control policy, Figure 8 shows the trajectory for the system controlled with an optimal control, the system controlled by the adaptive critic method, and the system controlled by neighboring optimal control determined from point x(0)=0.95. As usual, the initial condition is arbitrary. It is chosen to be any point in the range for which the neural network was trained.

D. Fourth Application (Rendezvous Problem-Finite Time 2-D Linear)

The fourth problem involves what could be considered a typical rendezvous problem. The system is described by the equation

$$x(t+1)=\begin{vmatrix}1 & 0.5\\ 0 & 1\end{vmatrix}x(t)+\begin{vmatrix}0.125\\ 0.5\end{vmatrix}u(t) \tag{18}$$

and the cost function is

$$J=\frac{1}{2}x^{T}(10)\begin{vmatrix}100 & 0\\ 0 & 100\end{vmatrix}x(10) \\ +\frac{1}{2}\sum_{t=0}^{9}\left(x^{T}(t)x(t)+u^{2}(t)\right) \tag{19}$$

The one step cost functions and utility functions are defined as in the previous problems. Figure 9 shows a comparison between the optimal control determined by conventional methods and the control law determined by the adaptive critic. Once again, the initial condition chosen is arbitrary.

E. Adaptive Capabilities of Critics

One of the additional benefits of adaptive critic based control is that the critics can be used to update a control which has become nonoptimal. This is done by allowing the critic to constantly update the control network after the correct critic has been determined.

To demonstrate this, the control from the rendezvous problem was multiplied by random factors between 0.8 and 1.2. (This was done by multiplying the final matrix in the neural network by the random factor.) After this, the critic was allowed to use 100 points from the system model in order to update the control policy. The altered control path and the corrected control path are shown along with the optimal control policy in Figure 10.

IV. Conclusions

It has been shown that neural networks can be used to determine near optimal control policies for low order linear and nonlinear systems. In the case of nonlinear systems, this could be beneficial as an alternative to the TPBVP methodology. This architecture requires no external training data and yields optimal control through the entire range of operation. This study has also shown how critics can be used as a redundancy to check and correct nonoptimal control.

Acknowledgement

This study was funded by NSF Grant ECS-9313946 and by the Missouri Department of Economic Development Center for Advanced Technology Program. The authors also thank Dr. Paul Werbos for the informative technical interactions and encouragement.

References

[1] Ismail, F., S. Wahsh, A. Mohamed and H. Elsimary, "Neural Network Application to an Optimal Control of a Variable Reluctance Motor", *Proc. of the 35th Midwest Symposium on Circuits and Systems*, 1992, Vol. 2, pp 1048-1051.

[2] Nikolauo, M. and V. Hanagandi, "Control of Nonlinear Dynamical Systems Modeled by Recurrent Neural Networks",*AIChE Journal,* Nov. 1993, Vol. 39, No. 11, pp. 1890-1894.

[3] Yamada,T. and T. Yabuta, "Nonlinear Neural Network Controller for Dynamic System",*IECON '90. 16th Annual Conf. of IEEE Industrial Electronics Society*, 1990, Vol. 2, pp 1244-1249.

[4] White, D. A. and D. Sofge, "Handbook of Intelligent Control" Van Nostrand Reinhold 1992.

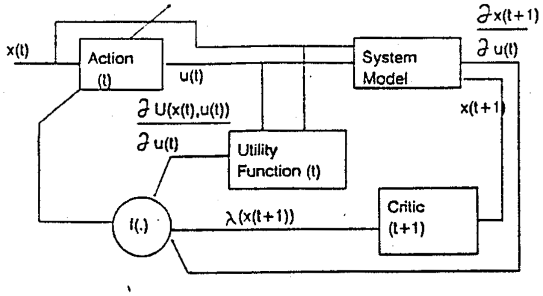
x(t)
Action (t)
u(t)
System Model
$\frac{\partial x(t+1)}{\partial u(t)}$
x(t+1)
$\frac{\partial U(x(t),u(t))}{\partial u(t)}$
Utility Function (t)
f(.)
λ(x(t+1))
Critic (t+1)

Figure 1: Action Network Training Diagram

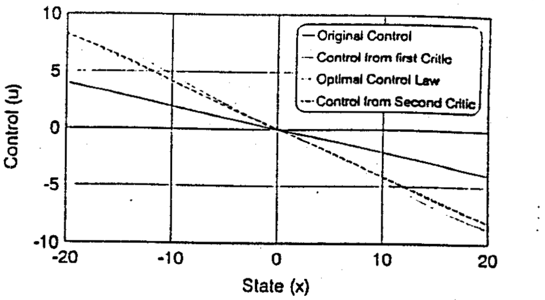

Figure 2: Control Law for Infinite Horizon Problem

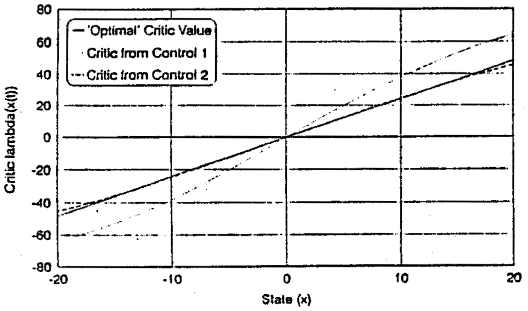

Figure 3: Critic for Infinite Horizon Problem

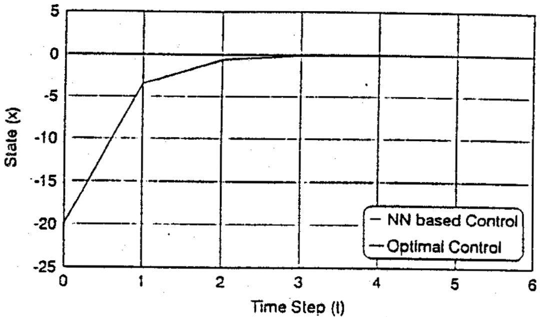

Figure 4: Controlled System with Initial Condition x(0)=-20

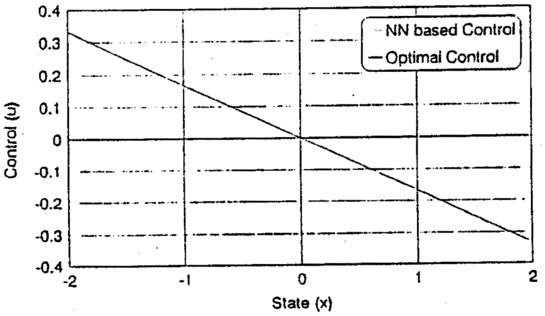

Figure 5: Optimal Control for Step 9

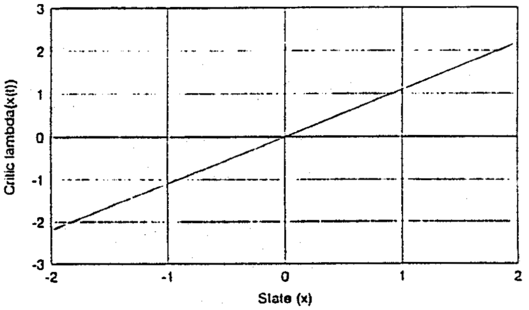

Figure 6: Critic for Step 9, Determined from Control for Step 9

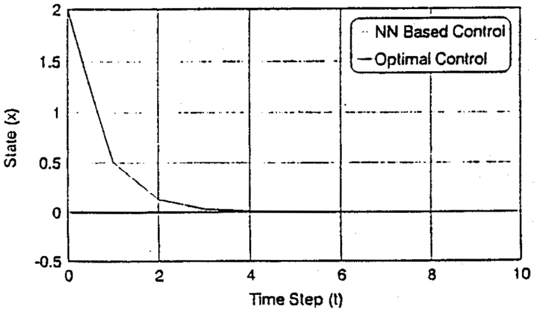

Figure 7: State of the System for Optimal Control and Adaptive Critic (Neural Network) Based Control

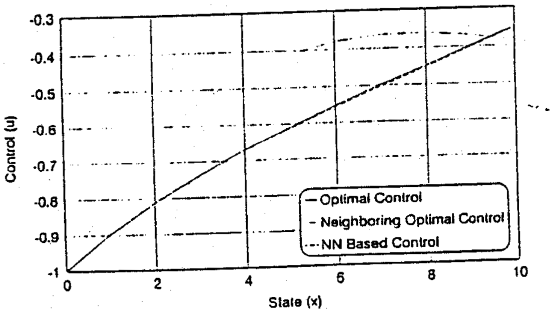

Figure 8: Controlled Nonlinear Problem for Initial Condition x(0)=-1

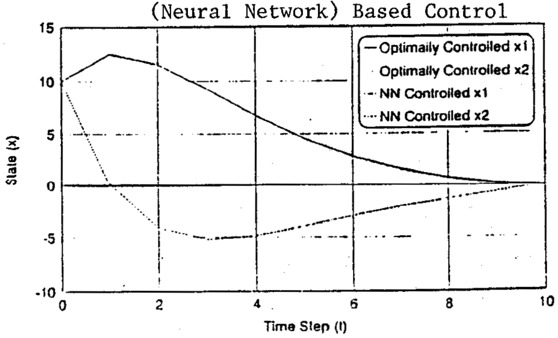

Figure 9: Controlled System with x1(0)-10, x2(0)-10

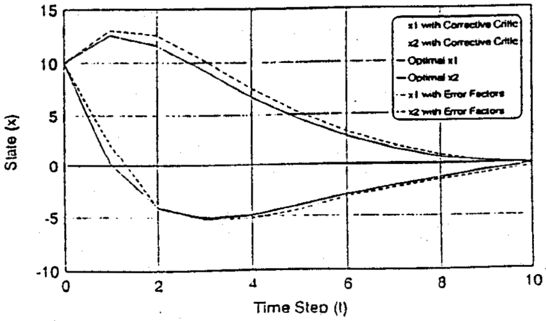

Figure 10: Controlled System with x1(0)-10, x2(0)-10