An Introduction to Adaptive Critic Control: A Paradigm Based on Approximate Dynamic Programming

Derong Liu
Department of Electrical and Computer Engineering
University of Illinois, Chicago, IL 60607, USA

Abstract. **Adaptive critic control is an advanced control technology developed for nonlinear dynamical systems in recent years. It is based on the idea of approximate dynamic programming. Dynamic programming was introduced by Bellman in the 1950's for solving optimal control problems of nonlinear dynamical systems. Due to its high computational complexity, applications of dynamic programming have been limited to simple and small problems. The key step in finding approximate solutions to dynamic programming is to estimate the cost function in dynamic programming. The optimal control signal can then be determined by minimizing (or maximizing) the cost function. Due to their universal approximation capability, artificial neural networks are often used to represent the cost function in dynamic programming. The implementation of approximate dynamic programming usually requires the use of three modules-Critic, Model, and Action. These three modules perform the function of evaluation, prediction, and decision, respectively. This article introduces some basic algorithms of adaptive critic control and some recent development of the area. It will also outline some future perspectives of this new control technology.**

1. Introduction

There are several spectrums about dynamic programming. One can consider discrete-time systems or continuous-time systems, linear systems or nonlinear systems, time-invariant systems or time-varying systems, deterministic systems or stochastic systems, etc.

We first take a look at discrete-time nonlinear (time-varying) dynamical (deterministic) systems. Time-varying nonlinear systems cover most of the application areas and discrete-time is the basic consideration for digital computation.

Suppose that one is given a discrete-time nonlinear (time-varying) dynamical system

$$x(t+1) = F[x(t),u(t),t],\ t = 0, 1, 2, \ldots \quad (1)$$

where $x \in R^n$ represents the state vector of the system and $u \in R^m$ denotes the control action. Suppose that one associates with this system the performance index (or cost)

$$J[x(i),i]=\sum_{k=i}^{\infty}\gamma^{k-i}U[x(k),u(k),k] \quad (2)$$

where U is called the utility function and γ is the discount factor with $0<\gamma\leq 1$. Note that the function J is dependent on the initial time i and the initial state $x(i)$, and it is referred to as the cost-to-go of state $x(i)$. The cost in this case accumulates indefinitely; this kind of problems is referred to as infinite horizon problems in dynamic programming. On the other hand, in finite horizon problems, the cost accumulates over a finite number of steps. This article will consider infinite horizon problems. The objective of dynamic programming problem is to choose a control sequence $u(k)$, $k=i, i+1, \ldots$, so that the function J (i.e., the cost) in (2) is minimized. Dynamic programming is based on Bellman's principle of optimality [4], [9], [14]: An optimal (control) policy has the property that no matter what previous decisions have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions.

Suppose that one has computed the optimal cost $J^*[x(t+1),t+1]$ from time $t+1$ to the terminal time, for all possible states $x(t+1)$, and that one has also found the optimal control sequences from time $t+1$ to the terminal time. The optimal cost results when the optimal control sequence $u^*(t+1)$, $u^*(t+2)$, ..., is applied to the system with initial state $x(t+1)$. Note that the optimal control sequence depends on $x(t+1)$. If one applies an arbitrary control $u(t)$ at time t and then uses the known optimal control sequence from $t+1$ on, the resulting cost will be

$$U[x(t),u(t),t]+\gamma J^*[x(t+1),t+1]$$

where $x(t)$ is the state at time t and $x(t+1)$ is determined by (1). According to Bellman, the optimal cost from time t on is equal to

$$J^*[x(t),t]= \min(U[x(t),u(t),t]+\gamma J^*[x(t+1),t+1]).$$

The optimal control $u^*(t)$ at time t is the $u(t)$ that achieves this minimum, i.e.,

$$u^*(t) = \underset{u(t)}{\operatorname{argmin}}(U[x(t),u(t),t]+\gamma J^*[x(t+1),t+1]). \quad (3)$$

Equation (3) is the principle of optimality for discrete-time systems. Its importance lies in the fact that it allows one to optimize over only one control vector at a time by working backward in time.

In continuous-time nonlinear case, the system can be expressed as

$$dx(t) / dt = F(x(t),u(t),t),\ t \geq t_0 \quad (4)$$

where $F(x,u,t)$ is an arbitrary continuous function. The cost in this case is defined as

$$J(x(t_0),u) = \int_{t_0}^{\infty} U(x(t),u(t),t)dt \quad (5)$$

More generally, when stochastic features are considered, the system will be described by

$$dx = F(x,u,t)dt + G(x,u,t)dw,\ t \geq t_0 \quad (6)$$

where w is a stochastic process, usually Wiener process or Gaussian process. For a given initial state $x(t_0)=x_0$ and feedback control $u(x,t)$, the cost for the system (6) is defined as

$$J(x_0,u)= E\{\int_{t_0}^{\infty} U(x,u,t)dt \mid x(t_0) = x_0\} \quad (7)$$

with nonnegative utility function $U(x,u,t)$.

For continuous-time systems, Bellman's principle of optimality of Bellman can be applied, too. The optimal cost $J^*(x_0)= \min J(x_0,u(t))$ will satisfy the Hamilton-Jacobi-Bellman Equation

$$-\partial J^* / \partial t = \min_{u(t)} (U(x,u,t) + (\partial J^* / \partial x)^T F(x,u,t)) \quad (8)$$

Meanwhile, the optimal control $u^*(t)$ will be the one that minimize the value of cost function,

$$u^*(t) = \underset{u(t)}{\operatorname{argmin}}(J(x,u)). \quad (9)$$

Dynamic programming is a very useful tool in solving optimization and optimal control problems. In particular, it can easily be applied to nonlinear systems with or without constraints on the control and state variables. Equations (3) and (9) are called the functional equation of dynamic programming and are the basis for computer implementation of dynamic programming. In the above, if the function F in (1), (4) or (6) and the cost function J in (2), (5) or (7) are known, the solution for $u^*(t)$ becomes a simple optimization problem. However, it is often computationally untenable to run true dynamic programming due to the backward numerical process required for its solutions, i.e., as a result of the well-known "curse of dimensionality" [4], [9]. The cost function J, which is the theoretical solution of the Hamilton-Jacobi-Bellman Equation, is very difficult to obtain, except for systems satisfying some very good conditions. Over the years, progress has been made to circumvent the "curse of dimensionality" by building a system, called "critic," to approximate the cost function in dynamic programming (cf. [2], [18], [21], [24], [29], [36], [38], [39]. [40]). The idea is to approximate dynamic programming solutions by using a function approximation structure such as neural networks to approximate the cost function.

2. Adaptive Critic Control Based on Approximate Dynamic Programming

In 1977, Werbos [35] introduced an approach for approximate dynamic programming that was later called adaptive critic designs (ACDs). ACDs have received increasing attention recently (cf. [2], [5]-[8], [11]-[13], [15]-[24], [28], [29], [33], [36]-[42]). In the literature, there are several synonyms used for "Adaptive Critic Designs" [2], [7], [11], [12], [16], [20], [21], [33], [42] including "Approximate Dynamic Programming" [28], [39], "Asymptotic Dynamic Programming" [22], "Adaptive Dynamic Programming" [18], [19], "Heuristic Dynamic Programming" [13], [37], "Neuro-Dynamic Programming" [5], "Neural Dynamic Programming" [28], [41], and "Reinforcement Learning" [31].

Figure 1 illustrates the concept of "Reinforcement Learning." The environment and the "Reinforcement Learning System" (RLS) are both assumed to have "memory" of the previous procedures. The goal of the reinforcement learning is to optimize the cost over future time, where the cost is the summation of $U(t)$.

The basic consideration of RLS for ACD was studied early to 1960's. In classical dynamic programming, the user supplies the utility function to be maximized and a stochastic model of the environment used to compute the expectation values. The mathematician then finds the function J, coming from a form of the Bellman equation. The key is to choose $u(t)$ that solves the simple, static maximization problem within that equation over infinite time. In [40], Werbos indicated that it must be helpful to use a universal function approximator-like a neural network-to approximate the function J or something very similar to it, in order to overcome the curse of dimensionality which keeps the classical dynamic programming from being useful on large problems.

In the linear case, everything seems to be good. Many methods could yield universal stable adaptive controller in the linear case. But in general, there is an obvious, overriding question we have to face: Can we develop nonlinear, stochastic learning controllers which meet all of those requirements, especially the stability of the control, when they are applied to linear plants?

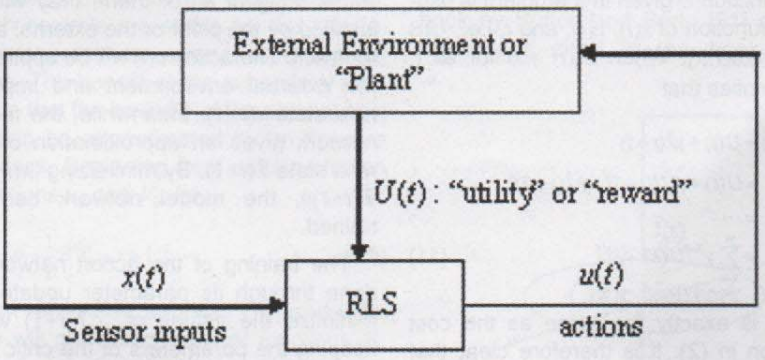

Figure 1. Learn from the environment

In [40], Werbos proposed the basic strategy in universal adaptive control design. One has to keep in mind that any adaptive control design will have the probability of extensions to a universal adaptive control. He suggested that there are at least six areas we have to do more about.

(1) Even in the "simple" textbook problem of supervised learning (learning a static mapping from an input vector X to a vector of targets or dependent variables Y), we need improved learning speed and generalization ability, exploiting concepts such as "syncretism" and "simultaneous recurrence."

(2) In adapting Model networks, we need to explore architectures which can represent probability distributions, not just deterministic input-output relations. We need to substantially improve "robustness over time." We need to explore Error Critics as an alternative to backpropagation through time (BTT), in order to combine real-time learning with the robust kind of memory now available only with BTT. The improvements needed in supervised learning should be extended to Model training as well.

(3) Substantial improvements in basic ADP technology are also needed. The work proposed in his paper addresses a portion of those needs.

(4) At a higher level, we need to develop, understand, test and apply designs for systems which perform "temporal chunking" in an adaptive way.

(5) Better systems are needed for "brain-like stochastic" search, in order to help decision-making system escape from local minima, without the artificial limitations of methods like genetic algorithms which are not truly brain-like. For example, suppose that we face a class of static function maximization problems, in which each specific problem is characterized by a set of parameters x; we are asked to find actions or decisions u which maximize $U(x,u)$. (As an example, x might represent the locations of cities in a traveling salesman problem.) We can train a network so that it inputs x and outputs possible u, in a stochastic way, and tries to ensure that it outputs u according to a Gibbs distribution, $c\exp(-kU/T)$, where T is a temperature parameter also input to the network.

(6) Principles like spatial symmetry and objective symmetry need to be implemented in new classes of neural networks, in order to make it possible for these networks to cope with the huge

volumes of input which biological organisms can handle. In effect, we need to work on neural adaptive designs which learn to do what some AI systems already do in a crude, brute-force manner: build up spatial "world models;" analyze objects one at a time, in a multiplexing kind of mode; perform spatial chunking; input and output networks of relations rather than fixed-length vectors.

Werbos [40] also discussed some existing control methods such as Heuristic Dynamic Programming (HDP), Dual Heuristic Programming (DHP) and Globalized DHP (GDHP) and introduce some alternative new variants of HDP, DHP and GDHP, which lead to some better controllers for universal ACD problem.

According to Werbos, to make the RLS model work well, we need three pieces of information to determine inside an RLS: (1) How to adapt the Critic network; (2) How to adapt the Model network; and (3) How to adapt the Action network.

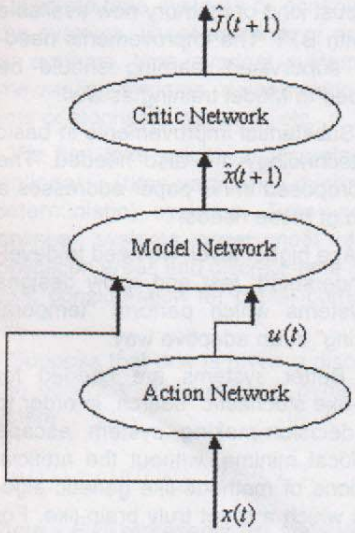

Figure 2. The three modules of an adaptive critic design

A typical design of ACDs consists of three modules-Critic, Model, and Action [21], [38], [39], [40] as shown in Figure 1. The critic network will give an estimation of the cost function J, which is guaranteed to be a Lyapunov function, at least for deterministic systems. Lyapunov stability theory in general has influenced huge sections of control theory, physics, and many other disciplines. More narrowly, within the disciplines of control theory and robotics, many researchers have tried to stabilize complex systems by first deriving Lyapunov functions for those systems. In some cases, the Lyapunov functions have been derived analytically by solving the multi-period optimization problem in an analytic fashion.

After one has derived such an application-specific Lyapunov function, one can then use a design exactly like Figures 1 and 2, except that the Lyapunov function replaces the square tracking error or J. Theoretically, by replacing the square tracking error with some other prespecified error measure, one arrives at a whole new class of stability properties, and a whole new set of restrictions on the plant.

From a practical point of view, it becomes more and more difficult to derive such Lyapunov functions analytically, as one tries to control more and more complex nonlinear systems, such as elastic, light-weight and flexible robot arms. The difficulties here are analogous to the difficulty of trying to solve simple algebraic equations analytically. As one progresses from quadratic equations, to cubic equations, to quartic equations, to sixth order equations, and so on, one eventually reaches a point where closed-form analytic methods simply cannot give you any solutions. At some point, one has to use computer-based numerical methods instead of analytical methods.

The present article considers the case where each module is a neural network. In the ACD scheme shown in Figures 1 and 2, the critic network outputs the function $\hat{J}$, which is an estimate of the function J in equation (2). This is done by minimizing the following square tracking error measure over time

$$\| E_h \| = \sum_t E_h(t) = \frac{1}{2}\sum_t [\hat{J}(t) - U(t) - \gamma \hat{J}(t+1)]^2 \quad (10)$$

where $\hat{J}(t) = \hat{J}[x(t), u(t), t, W_c]$ and W_c represents the parameters of the critic network. The function U is the same utility function as the one in (2) which indicates the performance of the overall system. The function U given in a problem is usually a function of $x(t)$, $u(t)$, and t, i.e., $U(t) = U[x(t), u(t), t]$. When $E_h(t) = 0$ for all t, (10) implies that

$$\begin{aligned} \hat{J}(t) &= U(t) + \gamma \hat{J}(t+1) \\ &= U(t) + \gamma[U(t+1) + \gamma \hat{J}(t+2)] \\ &= \cdots \\ &= \sum_{k=t}^{\infty} \gamma^{k-t} U(k) \end{aligned} \quad (11)$$

which is exactly the same as the cost function in (2). It is therefore clear that minimizing the error function in (10), we will have a neural network trained so that its output $\hat{J}$ becomes an estimate of the cost function J defined in (2).

The model network in Figure 2 learns the nonlinear function F given in equation (1); it is trained previously off-line [21], [39], or trained in parallel with the critic and action networks [22].

After the model network is trained, the critic network will be modified. The critic network give an estimate of the cost. The training of the critic network in this case is achieved in minimizing the error function defined in (10), for which many standard neural network training algorithms can be utilized [10]. Note that in Figure 2, the output of the critic network $\hat{J}(t+1) = \hat{J}(x(t+1), x(t+1), t+1)$ is an approximation to the cost function J at time $t+1$ and $\{x(t+1)\}$ is not a real trajectory but a prediction of the states by running the model network before running of the real plant.

After the critic network's training is finished, the action network's training starts with the objective of minimizing $\hat{J}(t+1)$, through the use of the action signal $u(t) = u[x(t), t, W_A]$. Once an action network is trained this way, i.e., trained by minimizing the output of critic network, we will have a neural network trained so that it will generate as its output an optimal, or at least, a suboptimal control action signal depending on how well the performance of the critic network is. Recall that the goal of dynamic programming is to obtain an optimal control sequence as in (3), which will minimize the function J in (2). The key here is to interactively build a link between present actions and future consequences via an estimate of the cost function.

After the action network's training cycle is completed, one may check the system performance, then stop or continue the training procedure by going back to the critic network's training cycle again, if the performance is not acceptable yet. This process will be repeated until an acceptable system performance is reached. The three networks will be connected as shown in Figure 2. As a whole thing of all of them, they will be justified by the plant or the external environment. The action $u(t)$ will be applied to the external environment and imply a new state $x(t+1)$. Meanwhile, the model network gives an approximation of the next state $\hat{x}(t+1)$. By minimizing $\|x(t+1) - \hat{x}(t+1)\|$, the model network can be trained.

The training of the action network is done through its parameter updates to minimize the values of $\hat{J}(t+1)$ while keeping the parameters of the critic and the model networks fixed. The gradient information is propagated backward through the critic network to the model network and then to the action network,

as if the three networks formed one large feedforward network (cf. Figure 2). This implies that the model network in Figure 2 is required for the implementation of adaptive critic designs in the present case. Even in the case of known function F, one still needs to build a model network so that the action network can be trained. In the next section, we will survey some new developments that include the simplification of the structure in Figure 2 by eliminating the model network.

For continuous-time system, the cost function J is also the key for dynamic programming. By minimizing J one gets the optimal cost function J^*, which is the Lyapunov function of the system. As a consequence of the Bellman's optimality principle, J^* satisfies the Hamilton-Jacobi-Bellman Equation (9). But usually, one cannot get the analytic solution of Hamilton-Jacobi-Bellman Equation. Even to find an accurate numerical solution is very difficult (the so-called "curse of dimensionality"). In 1994, Saridis and Wang [24] studied the non-linear stochastic systems described by

$$dx = F(x,t)dt + B(x,t)udt + G(x,t)dw,\ t_0 \le t \le T \tag{12}$$

with the performance cost

$$J(x_0,u) = E\left\{\int_{t_0}^{\infty}[U(x,t)+\|u\|^2]dt + \phi(x(T),T)\,|\,x(t_0)=x_0\right\} \tag{13}$$

where U and ϕ are nonnegative functions. Instead of solving Hamilton-Jacobi- Bellman Equation, they introduced the following equation

$$V_t + L_u V + U(x,t) + \|u\|^2 = \nabla V \tag{14}$$

An upper bound V^* and a lower bound $^*V$ of the optimal cost J^* are found by solving equation (14). Then a control law $u(x)$ can be obtained by applying V^* (or $^*V$) as Lyapunov function for the system. This leads to the so-called "suboptimal control" of the system. It was proved that such controls are stable for the infinite-time approximate optimal control problems. The benefit of the suboptimal control is that the bound V of the optimal cost J^* can be approximated by an iterative process. Beginning from certain chosen functions u_0 and V_0, let

$$u_i = -\frac{1}{2}B^T V_{i-1},\ i = 1,2,\cdots. \tag{15}$$

Then repeatedly applying (14) and (15), one will get a sequence of functions V_i. This sequence $\{V_i\}$ will converges to the bound V^* (or $^*V$) of the cost function J^*. Consequently, u_i will approximate the suboptimal control when i tends to infinity. Although suboptimal control is not the optimal control law, it does work. The important things are the sequences $\{V_i\}$ and $\{u_i\}$ are obtainable by computation and they approximate the optimal cost and the optimal control law, respectively.

In [5], Bertsekas and Tsitsiklis give an overview of the neuro-dynamic programming. They provide the background, give a detailed introduction to dynamic programming, discuss the neural network architectures and methods for training them, and develop general convergence theorems for stochastic approximation methods as the foundation for the analysis of various neuro-dynamic programming algorithms. They provide the core neuro-dynamic programming methodology, including many mathematical results and methodological insights. They suggested many useful methodologies to apply in neuro-dynamic programming, like Monte Carlo simulation, on-line and off-line temporal difference methods, Q-learning algorithm, optimistic policy iteration methods, Bellman error methods, approximate linear programming, approximate dynamic programming with cost-to-go function, etc.

A particularly impressive success that greatly motivated subsequent research, was the development of a backgammon playing program by Tesauro [32]. Here a neural network was trained to approximate the optimal cost-to-go function of the game of backgammon by using simulation, that is, by letting the program play against itself. Unlike chess programs, this program did not use lookahead of many steps, so its success can be attributed primarily to the use of a properly trained approximation of the optimal cost-to-go function.

3. Recent Developments and Future Perspectives

The main research results in reinforcement learning can be found in a recent book by Sutton and Barto [31] and the references cited in the book. Even though both reinforcement learning and adaptive critic designs provide approximate solutions to dynamic programming, research in these two directions has been somewhat independent [3] in the past. The most famous algorithms in reinforcement learning are the temporal difference algorithm [30] and the Q-learning algorithm [34]. Compared to adaptive critic designs, the area of reinforcement learning is more mature and has a vast amount of literature. The main constraint in most of the reinforcement learning literature is the use of look-up tables for representation of the cost function in dynamic programming which implies discrete state variables with finite number of values.

The most important recent advances of adaptive critic control start with the lieterature [21] and [29]. Reference [21] provides a detailed summary of the major developments in adaptive critic designs up to 1997. Before that, major references are papers by Werbos such as [35], [38], [39]. Werbos has pointed out many times that "adaptive critic designs/approximate dynamic programming may be the only approach that can achieve truly brain-like intelligence" [23], [36]. Reference [29] makes significant contributions to model-free adaptive critic designs. Using the approach of [29], the model network in Figure 2 is not needed anymore. Several practical examples are included in [29] for demonstration which include single inverted pendulum [1] and triple inverted pendulum. Reference [16] is also about model-free adaptive critic designs. Two approaches for the training of critic network are provided in [16]: A forward-in-time approach and a backward-in-time approach. Figure 3 shows the diagram of forward-in-time approach. In this approach, we view $\hat{J}(t)$ in (10) as the output of the critic network to be trained and choose $U(t)+\gamma\hat{J}(t+1)$ as the training target. Note that $\hat{J}(t)$ and $\hat{J}(t+1)$ are obtained using state variables at different time instances. Figure 4 shows the diagram

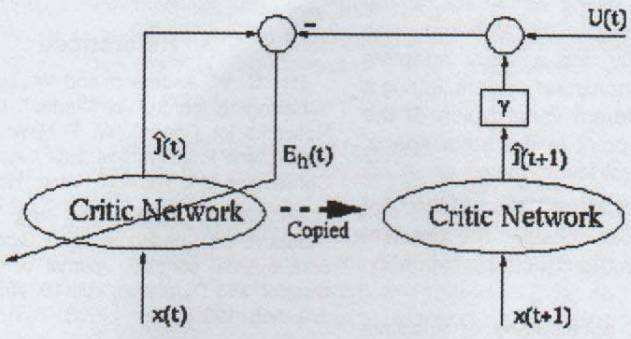

Figure 3. Forward-in-time approach.

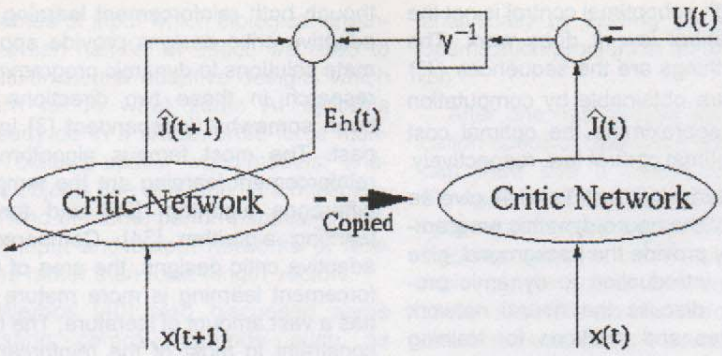

Figure 4. Backward-in-time approach.

of backward-in-time approach. In this approach, we view $\hat{J}(t+1)$ in (4) as the output of the critic network to be trained and choose $[\hat{J}(t)-U(t)]/\gamma$ as the training target. The training approach of [29] can be considered as a backward-in-time approach. In Figures 3 and 4, $x(t+1)$ is the output from the model network.

Some theoretical results for adaptive critic control have been obtained recently [18], [19], [22]. These references investigated the stability and optimality for some special cases of adaptive critic control. In [18], Murray et al. studied the (deterministic) continuous-time systems

$$dx(t)/dt = F(x)+B(t)u,\ x(t_0) = x_0 \qquad (16)$$

with the cost function

$$J = \int_{t_0}^{\infty} U(x,u)dt \qquad (17)$$

where $U(x,u) = q(x)+u^T r(x)u$ is a nonnegative function and $r(x) > 0$. Similar to [24], an iterative process is proposed to find the control law. But this time the optimal cost and optimal control law are approximated. For the plant (16) and performance cost (17), the Hamilton-Jacobi-Bellman Equation can be simplified to

$$u^*(x) = -\frac{1}{2}r^{-1}(x)B^T(x)\left[\frac{dJ^*(x)}{dx}\right]^T \qquad (18)$$

Applying (17) and (18) repeatedly, one will get sequences of estimations of the optimal cost function J^* and the optimal control u^*. By taking advantage of the intrinsic adaptivity of the algorithm, one could potentially use a linear adaptive controller on a nonlinear system, letting it adapt to a different linearization of the plant at each point in the state space. Since the control law is based on dV_i/dx, not $V_i(x)$, any approximation of the cost functional should consider the gradient error as well as the direct approximation error.

Most of the applications of adaptive critic control are in the area of aircraft flight control [2], [6], [7], [20]. Some other applications have also been reported recently such as in power systems [33], in communication networks [17], [42], and in engine control [11], [12]. Interested readers should also read reference [13], especially the proposed training strategies for the critic network and the action network. In addition, the authors of [29] provide the MATLAB programs of their algorithms free of charge. New comers to the field of adaptive critic control should take a look at the challenging control problems listed in [1]. There have also been two invited sessions on Adaptive Critic Control co-organized by the author of this article at the IEEE International Symposium on Intelligent Control (Houston, 2003 and Taiwan, 2004). Finally, references [25]-[27] present an approach for finite horizon dynamic programming called "Neural Dynamic Optimization."

Future research in the field of adaptive critic control/approximate dynamic programming calls for major breakthroughs in both theory and applications. In the theoretical aspect, a complete set of theories is needed for this area which includes stability, convergence, optimality, and qualitative analysis. On the other hand, applications with significant impact and economic benefits are wanting. There are currently on-going investigations in both of these two areas in the United States.

References

[1] C. W. Anderson and W. T. Miller III, "Challinging control problems," In Neural Networks for Control (W. T. Miller III, R. S. Sutton, and P. J. Werbos, Eds.). Appendix A. Cambridge, MA: The MIT Press, 1990.

[2] S. N. Balakrishnan and V. Biega, "Adaptive-critic-based neural networks for aircraft optimal control," Journal of Guidance, Control, and Dynamics, vol. 19, pp. 893-898, July-Aug. 1996.

[3] A. G. Barto, "Reinforcement learning and adaptive critic methods," in Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches (Chapter 12), Edited by D. A. White and D. A. Sofge, New York, NY: Van Nostrand Reinhold, 1992.

[4] R. E. Bellman, Dynamic Programming, Princeton, NJ: Princeton University Press, 1957.

[5] D. P. Bertsekas and J. N. Tsitsiklis, Neuro-Dynamic Programming, Belmont, MA: Athena Scientific, 1996.

[6] C. Cox, S. Stepniewski, C. Jorgensen, R. Saeks, and C. Lewis, "On the design of a neural network autolander," International Journal of Robust and Nonlinear Control, vol. 9, pp. 1071-1096, Dec. 1999.

[7] J. Dalton and S. N. Balakrishnan, "A neighboring optimal adaptive critic for missile guidance," Mathematical and Computer Modeling, vol. 23, pp. 175-188, Jan. 1996.

[8] P. G. DeLima and G. G. Yen, "Multiple model fault tolerant control using globalized dual heuristic programming," Proceedings of the 18th IEEE International Symposium on Intelligent Control, Houston, TX, Sept. 2003, pp. 523-528. (Invited paper)

[9] S. E. Dreyfus and A. M. Law, The Art and Theory of Dynamic Programming, New York, NY: Academic Press, 1977.

[10] S. Haykin, Neural Networks: A Comprehensive Foundation, Upper Saddle River, NJ: Prentice Hall, 1999.

[11] H. Javaherian, D. Liu, Y. Zhang, and O. Kovalenko, "Adaptive critic learning techniques for automotive engine control," Proceedings of the American Control Conference, Boston, MA, June 2004, pp. 4066-4071.

[12] N. V. Kulkarni and K. KrishnaKumar, "Intelligent engine control using an adaptive critic," IEEE Transactions on Control Systems Technology, vol. 11, pp. 164-173, Mar. 2003.

[13] G. G. Lendaris and C. Paintz, "Training strategies for critic and action neural networks in dual heuristic programming method," Proceedings of the 1997 IEEE International Conference on Neural Networks, Houston, TX, June 1997, pp. 712-717.

[14] F. L. Lewis and V. L. Syrmos, Optimal Control, New York, NY: John Wiley, 1995.

[15] D. Liu and H. D. Patiño, "A self-learning ship steering controller based on adaptive critic designs," Proceedings of the IFAC Triennial World Congress, Beijing, China, July 1999, vol. J, pp. 367-372.

[16] D. Liu, X. Xiong, and Y. Zhang, "Action-dependent adaptive critic designs," Proceedings of the INNS-IEEE International Joint Conference on Neural Networks, Washington, DC, July 2001, pp. 990-995.

[17] D. Liu, Y. Zhang, and H. Zhang, "A self-learning call admission control scheme for CDMA cellular networks," IEEE Transactions on Neural Networks, vol. 16, no. 5, Sept. 2005. (to appear)

[18] J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, "Adaptive dynamic programming," IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and

Reviews, vol. 32, pp. 140-153, May 2002.

[19] J. J. Murray, C. J. Cox, and R. E. Saeks, "The adaptive dynamic programming theorem," in Stability and Control of Dynamical Systems with Applications, D. Liu and P. J. Antsaklis, Editors, Boston, MA: Birkhäuser, 2003, pp. 379-394.

[20] D. V. Prokhorov, R. A. Santiago, and D. C. Wunsch, "Adaptive critic designs: A case study for neurocontrol," Neural Networks, vol. 8, pp. 1367-1372, 1995.

[21] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," IEEE Transactions on Neural Networks, vol. 8, pp. 997-1007, Sept. 1997.

[22] R. E. Saeks, C. J. Cox, K. Mathia, and A. J. Maren, "Asymptotic dynamic programming: Preliminary concepts and results," Proceedings of the 1997 IEEE International Conference on Neural Networks, Houston, TX, June 1997, pp. 2273-2278.

[23] R. A. Santiago and P. J. Werbos, "New progress towards truly brain-like intelligent control," Proceedings of the World Congress on Neural Networks, San Diego, CA, June 1994, vol. I, pp. 27-33.

[24] G. N. Saridis and F.-Y. Wang, "Suboptimal control of nonlinear stochastic systems," Control-Theory and Advanced Technology, vol.10, no. 4, pp. 847-871, 1994.

[25] C.-Y. Seong and B. Widrow, "Neural dynamic optimization for control systems-Part I: Background," IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 31, pp. 482-489, Aug. 2001.

[26] C.-Y. Seong and B. Widrow, "Neural dynamic optimization for control systems-Part II: Theory," IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 31, pp. 490-501, Aug. 2001.

[27] C.-Y. Seong and B. Widrow, "Neural dynamic optimization for control systems-Part III: Applications," IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 31, pp. 502-513, Aug. 2001.

[28] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, Editors, Handbook of Learning and Approximate Dynamic Programming, New York: Wiley-IEEE Press, 2004.

[29] J. Si and Y.-T. Wang, "On-line learning control by association and reinforcement," IEEE Transactions on Neural Networks, vol. 12, pp. 264-276, Mar. 2001.

[30] R. S. Sutton, "Learning to predict by the methods of temporal differences," Machine Learning, vol. 3, pp. 9-44, 1988.

[31] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, Cambridge, MA: The MIT Press, 1998.

[32] G. J. Tesauro, "Practical issues in temporal difference learning," Machine Learning, Vol. 8, pp. 257-277, 1992.

[33] G. K. Venayagamoorthy, R. G. Harley, and D. G. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," IEEE Transactions on Neural Networks, vol. 13, pp. 764-773, May 2002.

[34] C. J. C. H. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, pp. 279-292, 1992.

[35] P. J. Werbos, "Advanced forecasting methods for global crisis warning and models of intelligence," General Systems Yearbook, vol. 22, pp. 25-38, 1977.

[36] P. J. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-17, pp. 7-20, Jan./Feb. 1987.

[37] P. J. Werbos, "Consistency of HDP applied to a simple reinforcement learning problem," Neural Networks, vol. 3, pp. 179-189, 1990.

[38] P. J. Werbos, "A menu of designs for reinforcement learning over time," in Neural Networks for Control (Chapter 3), Edited by W. T. Miller, R. S. Sutton, and P. J. Werbos, Cambridge, MA: The MIT Press, 1990.

[39] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches (Chapter 13), Edited by D. A. White and D. A. Sofge, New York, NY: Van Nostrand Reinhold, 1992.

[40] P. J. Werbos, "Stable adaptive control using new critic designs," http://xxx.lanl.gov/abs/adap-org/9810001, March 1998. [Online].

[41] L. Yang, R. Enns, Y.-T. Wang, and J. Si, "Direct neural dynamic programming," in Stability and Control of Dynamical Systems with Applications (Chapter 10), Edited by D. Liu and P. J. Antsaklis, Boston, MA: Birkhauser, 2003.

[42] Y. Zhang and D. Liu, "Call admission control for CDMA cellular networks using adaptive critic designs," Proceedings of the 18th IEEE International Symposium on Intelligent Control, Houston, TX, Sept. 2003, pp. 511-516. (Invited paper)

Derong Liu received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, Indiana, in 1994, the M.S. degree in electrical engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1987, and the B.S. degree in mechanical engineering from the East China Institute of Technology (now Nanjing University of Science and Technology), Nanjing, China, in 1982. From 1982 to 1984, he was a product design engineer at China North Industries Corporation, Jilin, China. From 1987 to 1990, he was an instructor at the Graduate School of the Chinese Academy of Sciences, Beijing, China. From 1993 to 1995, he was a staff fellow at General Motors Research and Development Center, Warren, Michigan. From 1995 to 1999, he was an Assistant Professor in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, New Jersey. He joined the University of Illinois at Chicago in 1999 as an Assistant Professor of Electrical Engineering and Computer Science, where he is now an Associate Professor of Electrical and Computer Engineering, of Bioengineering, and of Computer Science. He is coauthor (with A. N. Michel) of the books Dynamical Systems with Saturation Nonlinearities: Analysis and Design (New York: Springer-Verlag, 1994) and Qualitative Analysis and Synthesis of Recurrent Neural Networks (New York: Marcel Dekker, 2002). He is coeditor (with P. J. Antsaklis) of the book Stability and Control of Dynamical Systems with Applications (Boston, MA: Birkhauser, 2003).

Dr. Liu was a member of the Conference Editorial Board of the IEEE Control Systems Society (1995-2000), served as an Associate Editor for IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications (1997-1999), and served as an Associate Editor for IEEE Transactions on Signal Processing (2001-2003). Since 2004, he has been an Associate Editor for IEEE Transactions on Neural Networks. He has served and is serving as a member of the organizing committee and the program committee of several international conferences. He was recipient of the Michael J. Birck Fellowship from the University of Notre Dame (1990), the Harvey N. Davis Distinguished Teaching Award from Stevens Institute of Technology (1997), and the Faculty Early Career Development (CAREER) award from the National Science Foundation (1999). He is a Fellow of the IEEE and a member of Eta Kappa Nu.