Approximate Dynamic Programming for Self-Learning Control[1)]

Derong Liu

(*Department of Electrical and Computer Engineering, University of Illinois, Chicago, IL* 60607 *U.S.A.*)
(E-mail: dliu@ece.uic.edu)

Abstract This paper introduces a self-learning control approach based on 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, the 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 performance index in dynamic programming. The optimal control signal can then be determined by minimizing (or maximizing) the performance index. Artificial neural networks are very efficient tools in representing the performance index in dynamic programming. This paper assumes the use of neural networks for estimating the performance index in dynamic programming and for generating optimal control signals, thus to achieve optimal control through self-learning.

Key words Approximate dynamic programming, learning control, neural networks

1 Introduction

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

$$\boldsymbol{x}(t+1) = F[\boldsymbol{x}(t), \boldsymbol{u}(t), t],\ t = 0, 1, 2, \cdots \tag{1}$$

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

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

where U is called the utility function and γ is the discount factor with $0 < \gamma \leqslant 1$. Note that the function J is dependent on the initial time i and the initial state $\boldsymbol{x}(i)$, and it is referred to as the cost-to-go of state $\boldsymbol{x}(i)$. 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. The present paper will consider infinite horizon problems. Finite horizon problems can be treated as well. The objective of the present dynamic programming problem is to choose a control sequence $\boldsymbol{u}(k)$, $k = i, i+1, \cdots$, so that the function J (*i.e.*, the cost) in (2) is minimized. Dynamic programming is based on Bellman's principle of optimality[1~3]: 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 $J^*[\boldsymbol{x}(t+1), t+1]$ from time $t+1$ to the terminal time, for all possible states $\boldsymbol{x}(t+1)$, 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 $\boldsymbol{u}^*(t+1)$, $\boldsymbol{u}^*(t+2)$, $\cdots$, is applied to the system with initial state $\boldsymbol{x}(t+1)$. Note that the optimal control sequence depends on $\boldsymbol{x}(t+1)$. If one applies an arbitrary control $\boldsymbol{u}(t)$ at time t and then uses the known optimal control sequence from $t+1$ on, the resulting cost will be

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

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

$$J^*[\boldsymbol{x}(t), t] = \min_{\boldsymbol{u}(t)} (U[\boldsymbol{x}(t), \boldsymbol{u}(t), t] + \gamma J^*[\boldsymbol{x}(t+1), t+1])$$

1) Supported by the National Science Foundation (U.S.A.) under Grant ECS-0355364
Received February 18, 2004; in revised form July 27, 2004

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

$$\boldsymbol{u}^*(t) = \arg\min_{\boldsymbol{u}(t)}(U[\boldsymbol{x}(t), \boldsymbol{u}(t), t] + \gamma J^*[\boldsymbol{x}(t+1), t+1]) \tag{3}$$

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. 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. Equation (3) is called the functional equation of dynamic programming and is the basis for computer implementation of dynamic programming. In the above, if the function F in (1) and the cost function J in (2) are known, the solution for $\boldsymbol{u}^*(t)$ 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"[1,2]. 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. [4 ∼ 9]). The idea is to approximate dynamic programming solutions by using a function approximation structure such as neural networks to approximate the cost function.

2 Approximate Dynamic Programming

In 1977, Werbos[10] introduced an approach for approximate dynamic programming that was later called adaptive critic designs (ACDs). ACDs have received increasing attention recently (cf. [4 ∼ 9, 11 ∼ 27]). In the literature, there are several synonyms used for "Adaptive Critic Designs"[4,5,13∼15,18,21,24,27] including "Approximate Dynamic Programming"[9], "Asymptotic Dynamic Programming"[22], "Adaptive Dynamic Programming"[19,20], "Heuristic Dynamic Programming"[16,25], "Neuro-Dynamic Programming"[11], "Neural Dynamic Programming"[26], and "Reinforcement Learning"[28].

A typical design of ACDs consists of three modules–Critic, Model, and Action[5,8,9], as shown in Fig. 1. The present work considers the case where each module is a neural network. In the ACD scheme shown in Fig. 1, 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 error measure over time

$$\|E_h\| = \sum_t E_h(t) = \frac{1}{2}\sum_t [\hat{J}(t) - U(t) - \gamma\hat{J}(t+1)]^2 \tag{4}$$

where $\hat{J}(t) = \hat{J}[\boldsymbol{x}(t), \boldsymbol{u}(t), t, W_C]$ 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 $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$, and t, *i.e.*, $U(t) = U[\boldsymbol{x}(t), \boldsymbol{u}(t), t]$. When $E_h(t) = 0$ for all t, (4) implies that

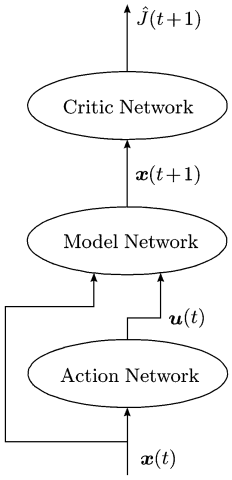

Fig. 1 The three modules of an adaptive critic design

$$\hat{J}(t) = U(t) + \gamma \hat{J}(t+1) = U(t) + \gamma[U(t+1) + \gamma \hat{J}(t+2)] = \cdots = \sum_{k=t}^{\infty} \gamma^{k-t} U(k)$$

which is exactly the same as the cost function in (2). It is therefore clear that minimizing the error function in (4), 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 training of the critic network in this case is achieved by minimizing the error function defined in (4), for which many standard neural network training algorithms can be utilized[29]. Note that in Fig. 1, the output of the critic network $\hat{J}(t+1) = \hat{J}[\boldsymbol{x}(t+1), \boldsymbol{u}(t+1), t+1]$ is an approximation to the cost function J at time $t+1$. The model network in Fig. 1 learns the nonlinear function F given in Equation (1); it is trained previously off-line[5,9] or trained in parallel with the critic and action networks[22]. After the critic network's training is finished, the action network's training starts with the objective of minimizing $\hat{J}(t)$, through the use of the action signal $\boldsymbol{u}(t) = \boldsymbol{u}[\boldsymbol{x}(t), t, W_A]$. 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 J function 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. During the training of action network, the three networks will be connected as shown in Fig. 1; 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 neural network (cf. Fig. 1). This implies that the model network in Fig. 1 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 Fig. 1 by eliminating the model network.

3 Literature survey

In this paper, we will survey some literature in two directions: Reinforcement learning and adaptive critic designs. Even though both types of literature provide approximate solutions to dynamic programming, research in these two directions has been somewhat independent[30] in the past.

The main research results in reinforcement learning can be found in a recent book by Sutton and Barto[28] and the references cited in the book. The most famous algorithms in reinforcement learning are the temporal difference algorithm[31] and the Q-learning algorithm[32]. 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.

To those who want to do research in adaptive critic designs, some helpful papers include [5, 6, 18]. Reference [6] 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 [7 ∼ 10]. 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"[7,23]. Reference [6] makes significant contributions to model-free adaptive critic designs. Using the approach of [6], the model network in Fig. 1 is not needed anymore. Several practical examples are included in [6] for demonstration which include single inverted pendulum[33] and triple inverted pendulum. Reference [18] is also about model-free adaptive critic designs. Two approaches for the training of critic network are provided in [18]: A forward-in-time

approach and a backward-in-time approach. Fig. 2 shows the diagram of forward-in-time approach. In this approach, we view $\hat{J}(t)$ in (4) 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. Fig. 3 shows the diagram 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 [6] can be considered as a backward-in-time approach. In Figs. 2 and 3, $\boldsymbol{x}(t+1)$ is the output from the plant.

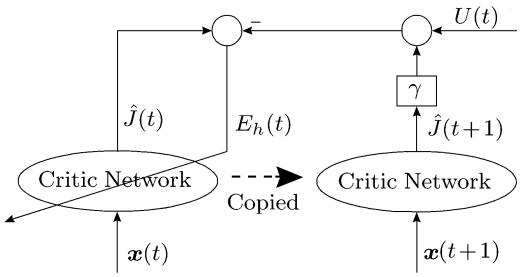

Fig. 2 Forward-in-time approach

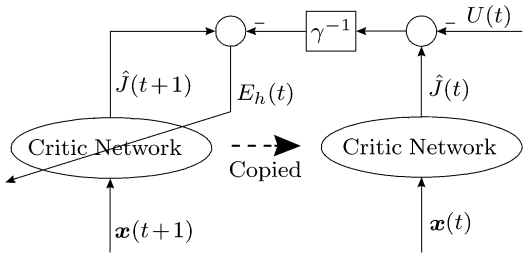

Fig. 3 Backward-in-time approach

Some theoretical results for adaptive critic designs have been obtained recently[19,20,22]. These references investigated the stability and optimality for some special cases of adaptive critic designs. Most of the applications of adaptive critic designs are in the area of aircraft flight control[4,12,13,21]. Some other applications have also been reported recently such as in power systems[24], in communication networks[27], and in engine control[14,15]. Interested readers should also read reference [16], especially the proposed training strategies for the critic network and the action network. In addition, the authors of [6] provide the MATLAB programs of their algorithms free of charge. New comers to the field of adaptive critic designs should start with the challenging control problems listed in [33]. Finally, references [34 ∼ 36] present an approach for finite horizon dynamic programming called "Neural Dynamic Optimization."

Future research in the field of adaptive critic designs/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. Interested readers can contact the author of this paper or any authors from the reference list.

References

1 Bellman R E. *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957
2 Dreyfus S E, Law A M. *The Art and Theory of Dynamic Programming*, New York, NY: Academic Press, 1977

3 Lewis F L, Syrmos V L. *Optimal Control*, New York, NY: John Wiley, 1995
4 Balakrishnan S N, Biega V. Adaptive-critic-based neural networks for aircraft optimal control, *Journal of Guidance, Control, Dynamics*, 1996, **19**(7-8): 893∼898
5 Prokhorov D V, Wunsch D C. Adaptive critic designs, *IEEE Transactions on Neural Networks*, 1997, **8**(9): 997∼1007
6 Si J, Wang Y-T. On-line learning control by association and reinforcement, *IEEE Transactions on Neural Networks*, 2001, **12**(3): 264∼276
7 Werbos P J. 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, 1987, 7∼20
8 Werbos P J. 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
9 Werbos P J. Approximate dynmic 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
10 Werbos P J. Aduanced forecasting methods for global crisis warning and models of intelligence, *General Systems Yearbook*, 1977, **22**: 25∼38
11 Bertsekas D P, Tsitsiklis J N. Neuro-Dynamic Programming, Belmont, MA: Athena Scientific, 1996
12 Cox C, Stepniewski S, Jorgensen C, Saeks R, Lewis C. On the design of a neural network autolander, *International Journal of Robust Nonlinear Control*, 1999, **9**: 1071∼1096
13 Dalton J, Balakrishnan S N. A neighboring optimal adaptive critic for missile guidance, *Mathematical and Computer Modeling*, 1996, **23**(1): 175∼188
14 Javaherian H, Liu D, Zhang Y, Kovalenko O. Adaptive critic learning techniques for automotive engine control, In: Proceedings of the American Control Conference, Boston, MA, 2004, **6**: 4066∼4071
15 Kulkarni N V, KrishnaKumar K. Intelligent engine control using an adaptive critic, *IEEE Transactions on Control Systems Technology*, 2003, **11**(3): 164∼173
16 Lendaris G G, Paintz C. Training strategies for critic and action neural networks in dual heuristic progrmming method, In: Proceedings of the 1997 IEEE International Conference on Neural Networks, Houston, TX, 1997, **6**: 712∼717
17 Liu D, Patiño. A self-learning ship steering controller based on adaptive critic designs, In: Proceedings of the IFAC Triennial World Congress, Beijing, China, 1999, J: 367∼372
18 Liu D, Xiong X, Zhang Y. Action-dependent adaptive critic designs, In: Proceedings of the INNS-IEEE International Joint Conference on Neural Networks, Washington, DC, 2001, **7**: 990∼995
19 Murray J J, Cox C J, Lendaris G G, Saeks R. Adaptive dynamic programming, *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, 2002, **32**(5): 140∼153
20 Murray J J, Cox C J, Saeks R E. The adaptive dynamic programming theorem, In: Stability and Control of Dynamical Systems with Applications, D. Liu and P. J. Antsaklins, Editors, Boston, MA: Birkhäuser, 2003, 379∼394
21 Prokhorov D, Santiago R A, Wunsch C C. Adaptive critic designs: A case study for neurocontrol, *Neural Networks*, 1995, **8**: 1367∼1372
22 Saeks R E, Cox C J, Mathia K, Maren A J. Asymptotic dynamic programming: In: Preliminary concepts and results, Proceedings of the 1997 International Conference on Neural Networks, Houston, TX, 1997, **6**: 2273∼2278
23 Santiago R A, Werbos P J. New progress towards truly brain-like intelligent control, *Proceedings of the World Congress on Neural Networks*, San Diego, CA, 1994, **1**(6): 27∼33
24 Venayagamoorthy G K, Harley R G, Wunsch D G. Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol or a turbogenerator, *IEEE Transactions on Neural Networks*, 2002, **13**(5): 764∼773
25 Werbos P J. Consistency of HDP applied to a simple reinforcement learning problem, *Neural Networks*, 1990, **3**: 179∼189
26 Yang L, Enns R, Wang Y-T, Si J. 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
27 Zhang Y, Liu D. Call admission control for CDMA cellular networks using adaptive critic designs, In: Proceedings of the 18th IEEE International Symposium on Intelligent Control, Houston, TX, 2003, **9**: 511∼516 (Invited paper)
28 Sutton R S, Barto A G. Reinforcement Learning: An Introduction, Cambridge, MA: The MIT Press, 1998
29 Haykin S. Neural Networks: A Comprehensive Foundation, Upper Saddle River, NJ: Prentice Hall, 1999
30 Barto A G. 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
31 Sutton R S. Learning to predict by the methods of temporal differences, *Machine Learning*, 1988, **3**: 9∼44

32 Watkins C J C H, Dayan P, Q-learning. *Machine Learning*, 1992, **8**: 279∼292
33 Anderson C W, Miller W T III. Challenging control problems, In: Neural Networks for Control (W. T. Miller III, R. S. Sutton, and P. J. Werbos, Eds.). Appendix A. The MIT Press, Cambridgem, MA, 1990
34 Seong C-Y, Widrow B. Neural dynamic optimization for control systems–Part I: Background, *IEEE Transactions on Systems, Man, and Cybernetics–Part B*: Cybernetics, 2001, **31**(8): 482∼489
35 Seong C-Y, Widrow B. Neural dynamic optimization for control systems–Part II: Theory, *IEEE Transactions on Systems, Man, and Cybernetics–Part B*: Cybernetics, 2001, **31**(8): 490∼501
36 Seong C-Y, Widrow B. Neural dynamic optimization for control systems–Part III: Applications, *IEEE Transactions on Systems, Man, and Cybernetics–Part B*: Cybernetics, 2001, **31**(8): 502∼513

Derong Liu Received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, Indiana, in 1994, the master degree in electrical engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, P.R.China, in 1987, and the bachelor degree in mechanical engineering from the East China Institute of Technology (now Nanjing University of Science and Technology), Nanjing, P.R.China, in 1982. From 1982 to 1984, he was a product design engineer at China North Industries Corporation, Jilin, P.R.China. From 1987 to 1990, he was an instructor at the Graduate School of the Chinese Academy of Sciences, Beijing, P.R.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. In addition, 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.