Automatica 59 (2015) 9–18

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Complete stability analysis of a heuristic approximate dynamic programming control design☆

Yury Sokolov [a], Robert Kozma [a,1], Ludmilla D. Werbos [b], Paul J. Werbos [b]

[a] Department of Mathematical Sciences, The University of Memphis, USA
[b] IntControl LLC, Arlington, and CLION, The University of Memphis, USA

ARTICLE INFO

Article history:
Received 10 August 2013
Received in revised form
22 April 2015
Accepted 17 May 2015

Keywords:
Adaptive dynamic programming
Action-Dependent Heuristic Dynamic Programming
Adaptive control
Adaptive critic
Neural network
Gradient descent
Lyapunov function

ABSTRACT

This paper provides new stability results for Action-Dependent Heuristic Dynamic Programming (ADHDP), using a control algorithm that iteratively improves an internal model of the external world in the autonomous system based on its continuous interaction with the environment. We extend previous results for ADHDP control to the case of general multi-layer neural networks with deep learning across all layers. In particular, we show that the introduced control approach is uniformly ultimately bounded (UUB) under specific conditions on the learning rates, without explicit constraints on the temporal discount factor. We demonstrate the benefit of our results to the control of linear and nonlinear systems, including the cart–pole balancing problem. Our results show significantly improved learning and control performance as compared to the state-of-the-art.

1. Introduction

Adaptive Dynamic Programming (ADP) addresses the general challenge of optimal decision and control for sequential decision making problems in real-life scenarios with complex and often uncertain, stochastic conditions without the presumption of linearity. ADP is a relatively young branch of mathematics; the pioneering work (Werbos, 1974) provided powerful motivation for extensive investigations of ADP designs in recent decades (Barto, Sutton, & Anderson, 1983; Bertsekas & Tsitsiklis, 1996; Lendaris, 2009; Si, Barto, Powell, & Wunsch, 2004; Vrabie & Lewis, 2009; Wang, Liu, Wei, Zhao, & Jin, 2012; Werbos, 1992; Zhang, Liu, Luo, & Wang, 2013). ADP has not only shown solid theoretical results to optimal control but also successful applications (Venayagamoorthy, Harley, & Wunsch, 2003). Various ADP designs demonstrated powerful results in solving complicated real-life problems, involving multi-agent systems and games (Al-Tamimi, Lewis, & Abu-Khalaf, 2007; Valenti, 2007; Zhang, Wei, & Liu, 2011).

The basic ADP approaches include heuristic dynamic programming (HDP), dual heuristic dynamic programming (DHP) and globalized DHP (GDHP) (Prokhorov & Wunsch, 1997; Werbos, 1974, 1990; White & Sofge, 1992). For each of these approaches there exists an action-dependent (AD) variation (White & Sofge, 1992). For several important cases, the existence of stable solution for ADP control has been shown under certain conditions (Abu-Khalaf & Lewis, 2005; Lewis & Liu, 2012; Vrabie & Lewis, 2009; Zhang, Zhang, Luo & Liang, 2013).

The stability of ADP in the general case is an open and yet unsolved problem. There are significant efforts to develop conditions for stability in various ADP designs. We solved the stability problem for the specific ADHDP control case using the Lyapunov approach, which is a classical method of investigating stability of dynamical processes. Here we address a discrete time dynamical system, where the dynamics is described by a difference equation. The discrete time Lyapunov function is used to prove the stability of the controlled process under certain conditions. In this paper we generalize the results of Liu, Sun, Si, Guo, and Mei (2012) for deriving stability conditions for ADHDP with traditional three layer Multi-Layer Perceptron (MLP). The work (Liu et al., 2012)

☆ This research was supported in part by National Science Foundation (NSF) to Robert Kozma in the CRCNS Program, grant #DMS-13-11165. The material in this paper was partially presented at the 2013 International Joint Conference on Awareness Science and Technology, November 2–4, 2013, Aizu-Wakamatsu, Japan. This paper was recommended for publication in revised form by Associate Editor Raul Ordóñez under the direction of Editor Miroslav Krstic.

E-mail addresses: ysokolov@memphis.edu (Y. Sokolov), rkozma@memphis.edu (R. Kozma), l.dalmat@gmail.edu (L.D. Werbos), werbos@ieee.org (P.J. Werbos).

[1] Tel.: +1 901 678 2497; fax: +1 901 678 2480.

http://dx.doi.org/10.1016/j.automatica.2015.06.001
0005-1098/

derives a stability condition for the system with weights adapted between the hidden and output layers only, under the assumption that networks have large enough number of neurons in the hidden layers.

The approach presented in Liu et al. (2012), in effect, is equivalent to a linear basis function approach: it is easy but it leads to scalability problems. The complexity of the system is growing exponentially for the required degree of approximation of a function of given smoothness (Barron, 1994). Additional problems arise regarding the accuracy of parameter estimation, which tends to grow with the number of parameters, all other factors are kept the same. If we have too many parameters for a limited set of data, it leads to overtraining. We need more parsimonious model, capable of generalization, hence our intention is to use fewer parameters in truly nonlinear networks, which is made possible by implementing more advanced learning algorithm. In the present work we focus on studying the stability properties of the ADP system with MLP-based critic, when the weights are adapted between all layers. By using Lyapunov approach, we study the uniformly ultimately bounded property of the ADHDP design. Preliminary results of our generalized stability studies have been reported in Kozma and Sokolov (2013), where we showed that our general approach produced improved learning and convergence results, especially in the case of difficult control problems.

The rest of the paper is organized as follows. First we briefly outline theoretical foundations of ADHDP. Next we describe the learning algorithm based on gradient descent in the critic and action networks. This is followed by the statements and the proofs of our main results on the generalized stability criteria of the ADP approach. Finally, we illustrate the results using examples of two systems. The first one is a simple linear system used in Liu et al. (2012), and the second example is the inverted pendulum system, similar to He (2011). We conclude the paper by outlining potential benefits of our general results for future applications in efficient real-time training and control.

2. Theoretical foundations of ADHDP control

2.1. Basic definitions

Let us consider a dynamical system (plant) with discrete dynamics, which is described by the following nonlinear difference equation:

$$x(t+1) = f\left(x(t), u(t)\right), \tag{1}$$

where x is the m-dimensional plant state vector and u is the n-dimensional control (or action) vector.

Previously we reported some stability results for ADP in the general stochastic case (Werbos, 2012). In this paper we focus on the deterministic case, as described in Eq. (1) and introduce action-dependent heuristic dynamic programming (ADHDP) to control this system. The original ADHDP method has been used in the 1990s for various important applications, including the manufacturing of carbon–carbon composite parts (White & Sofge, 1992). ADHDP is a learning algorithm for adapting a system made up of two components, the critic and the action, as shown in Fig. 1. These two major components can be implemented using any kind of differentiable function approximator. Probably the most widely used value function approximators in practical applications (as surveyed in Lewis & Liu, 2012) are neural networks, linear basis function approximators, and piecewise linear value functions such as those used by Powell (2011). In this work we use MLP as the universal function approximator.

The optimal value function, J^* is the solution of the Bellman equation (White & Sofge, 1992), which is a function of the state variables but not of the action variables. Here we use function

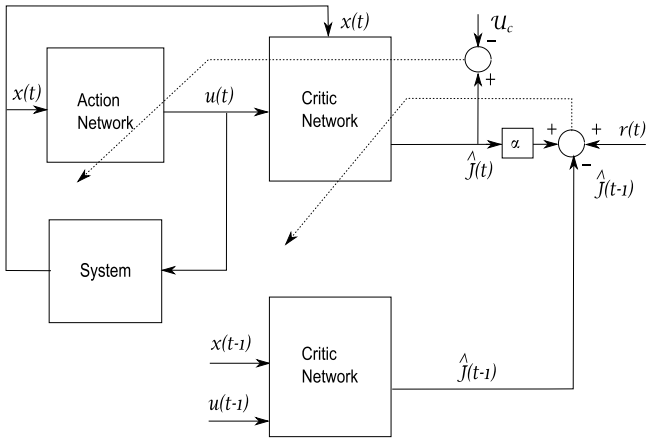

Fig. 1. Schematics of the implemented ADHDP design.

J, which is closely related to J^*, but J is a function of both the state and the action variables. Function J is often denoted by J' in the literature, following the definition in White and Sofge (1992, Chapter 3). The critic provides the estimate of function J, which is denoted as $\hat{J}$. Function Q, used in traditional Q-learning (Si et al., 2004) is the discrete-variable equivalent of J.

The action network represents a control policy. Each combination of weights defines a different controller, hence by exploring the space of possible weights we approximate the dynamic programming solution for the optimal controller. ADHDP is a method for improving the controller from one iteration to the next, from time instant t to $t+1$. We also have internal iterations, which are not explicit (He, 2011; Lewis & Liu, 2012). Namely, at a given t, we update the weights of the neural networks using supervised learning for a specific number of internal iteration steps.

In ADHDP, the cost function is expressed as follows; see, e.g., Lewis and Liu (2012):

$$J(x(t), u(t)) = \sum_{i=t}^{\infty} \alpha^{i-t} r(x(i+1), u(i+1)), \tag{2}$$

where $0 < \alpha \leq 1$ is a discount factor for the infinite horizon problem, and $r(x(t), u(t))$ is the reward or reinforcement or utility function (He, 2011; Zhang, Liu et al., 2013). We require $r(t) = r(x(t), u(t))$ to be a bounded semidefinite function of the state $x(t)$ and control $u(t)$, so the cost function is well-defined. Using standard algebra one can derive from (2) that $0 = \alpha J(t) + r(t) - J(t-1)$, where $J(t) = J(x(t), u(t))$.

2.2. Action network

Next we introduce each component, starting with the action component. The action component will be represented by a neural network (NN), and its main goal is to generate control policy. For our purpose, MLP with one hidden layer is used. At each time step this component needs to provide an action based on the state vector $x(t) = (x_1(t), \ldots, x_m(t))^T$, so $x(t)$ is used as an input for the action network. If the hidden layer of the action MLP consists of N_{h_a} nodes; the weight of the link between the input node j and the hidden node i is denoted by $\hat{w}^{(1)}_{a_{ij}}(t)$, for $i = 1, \ldots, N_{h_a}$ and $j = 1, \ldots, m$. $\hat{w}^{(2)}_{a_{ij}}(t)$, where $i = 1, \ldots, n, j = 1, \ldots, N_{h_a}$ is the weight from j's hidden node to i's output. The weighted sum of all inputs, i.e., the input to a hidden node k is given as $\sigma_{a_k}(t) = \sum_{j=1}^{m} \hat{w}^{(1)}_{a_{kj}}(t) x_j(t)$. The output of hidden node k of the action network is denoted by $\phi_{a_k}(t)$.

For neural networks a variety of transfer functions are in use, see, e.g. Zhang, Liu et al. (2013). Hyperbolic tangent is a common