Paper:

Random-TD Function Approximator

Hassab Elgawi Osman

Image Science and Engineering Lab, Tokyo Institute of Technology
4259 Nagatsuta, Midori-ku, Yokohama 226-8503, Japan
E-mail: osman@isl.titech.ac.jp

[Received July 7, 2008; accepted January 9, 2009]

In this paper, adaptive controller architecture based on a combination of temporal-difference (TD) learning and an on-line variant of Random Forest (RF) classifier is proposed. We call this implementation Random-TD. The approach iteratively improves its control strategies by exploiting only relevant parts of action and is able to learn completely in on-line mode. Such capability of on-line adaptation would take us closer to the goal of more robust and adaptable control. To illustrate this and to demonstrate the applicability of the approach, it has been applied to a non-linear, non-stationary control task, Cart-Pole balancing and on high-dimensional control problems –Ailerons, Elevator, Kinematics, and Friedman–. The results demonstrate that our hybrid approach is adaptable and can significantly improves the performance of TD methods while speeding up the learning process.

Keywords: adaptive control, function approximation (FA), TD-learning, random forests (RF)

1. Introduction

Building adaptive controllers for robots or mechanisms in an unknown and unstructured environment has been a great challenge. The performance of animals in the simplest motor tasks turn out to be extremely difficult to reproduce in artificial mechanical devices, simulated or real. Based on reinforcement learning (RL), agents can be partially successful to perform mode-free learning of action policies for some control problems. The biggest problems of the RL algorithm when applied to a real system are the curse of dimensionality, and its reliance on a number of assumptions about the nature of environment in which the learning takes place. e.g., it is usually assumed that the process to be controlled is either open loop stable[1] or of slow dynamic. Unfortunately, these assumptions are usually violated by any control application domain operating in the real world.

Due to the above limitations in adaptability of the current RL algorithms interest to a hybrid that combines different learning strategies model arose as well. The success of RL method in application to the intelligent control of continuous control systems is turned out to be depend on the ability to combine proper function approximation (FA) method with temporal-difference (TD) methods such as Q-learning and value iteration. See for example [8, 10, 11]. Currently, stochastic policy gradient methods are also attracting much attention [12]. This paper proposes the use of a behavior-based control architecture and investigates on some techniques inspired by Nature- a combination of reinforcement and supervised learning algorithms to accomplish the sub-goals of a mission of building an adaptive controller. In the presented approach, the above mentioned limitations are solved by a hybrid architecture combining the temporal-difference (TD) learning algorithm with an on-line variant of random forests (RF) learner that we have developed recently [5]. We call this implementation Random-TD. The approach iteratively improves its value function by exploiting only relevant parts of action space, and is able to learn completely in on-line mode, and since we do not freeze the learning, it can adapt to a changing situation. Note that this kind of adaptability is not possible in supervised batch learners or the standard RL. Such capability of on-line adaptation would take us closer to the goal of more robust and adaptable control. We provide a motivation for this new approach in terms of a reduction in the Bellman error. Learning is expected to be faster than in TD learning, which uses a stochastic search. Our results below on high-dimensional control problems –Ailerons, Elevator, Kinematics, and Friedman– and on a nonlinear, non-stationary control task –Cart-Pole balancing– confirms this prediction, and suggest that the learning algorithm could become scalable and produce large, adaptive systems.

2. Random-TD Architecture

Figure 1 demonstrates the architecture of the proposed Random-TD. It consists of two modules, a learning module and a hybrid module. These two modules interact with each other. The learning module is in the first level; in which both on-line RF and TD learner learn their control policies, based on its current policy and state. Then, each learner submits its decision of the selected action or the preference of actions to the hybrid module, where the Random-TD learns the combined policies. The 'Action Selector' chooses an action based on a value function and

1. There is no set procedures to tune controllers.

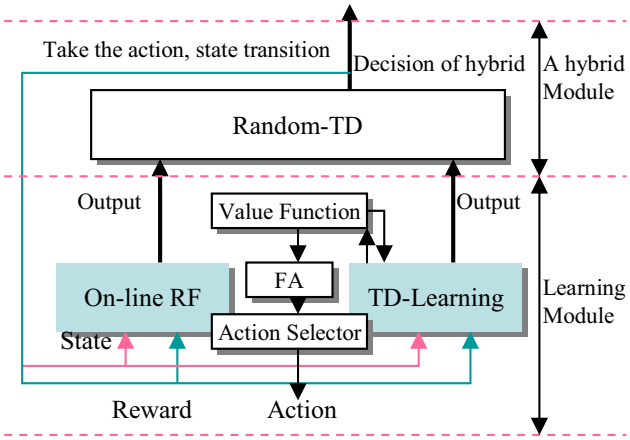

Fig. 1. Random-TD architecture. It consists of two modules, a learning module and a hybrid module. These two modules interact with each other.

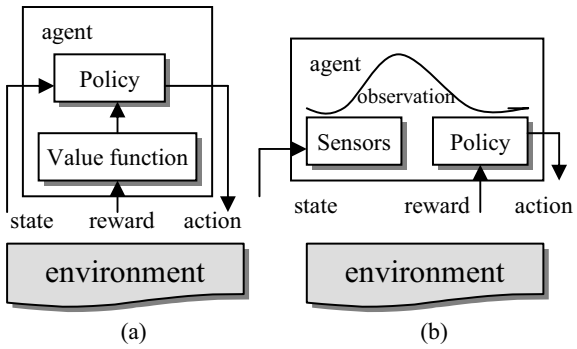

Fig. 2. Reinforcement learning framework. In (a) a fully observable world, the agent can estimate a value function for each state and use its actions. In (b) a partially observable world, in which the agent does not know which state it is in due to sensor limitations; for the value function, the agent updates its policy parameters directly.

exploration and exploitation strategies. The hybrid module is at the second level, where the input information from learning module is dynamically aggregated. After that, the hybrid module sends a final decision of action back to the learning module. Then every learner in the learning module takes the action, transits to a new state, and obtains an instant reward. Then, each learner updates its policy. Repeat for a number of steps or until several criteria are satisfied. Because both the TD-algorithm and RF run on-line, we are freed from the curse of dimensionality in a sense that memory requirements need not be exponential in the number of dimensions. The overall effect is particularly efficient computationally.

3. Reinforcement Learning for Control Systems

Let us consider a dynamic control system:

$$\frac{ds(t)}{dt} = f(s(t),(a(t)). \quad \ldots\ldots\ldots\ldots (1)$$

As illustrated in **Fig. 2** RL solves sequential decision making problems modeled by stochastic process *Markov Decision Processes* (MDPs) of a tuple $\langle S,A,P,R\rangle$ where S is the space of possible *states* of the environment, A is a set of *actions* available to the agent (or control input), $P : S \times A \times S \to [0,1]$ defines a conditional probability distribution over *state transitions* given an action, and $R : S \times A \to R$ is a *reward function* (payoff) assigning an immediate reward for an action. The main idea of RL is that through trial and error, rather than being implicitly told continuous interactions between an agent and its dynamic environment can be made in order to satisfy the goal of autonomy and the adaptability. We refer the reader to a textbook [9] for a good introduction to the problem addressed by RL and the details of standard solutions.

3.1. Hybrid MDP

In our modeling, we assume a large but limited state space represented by control input matrix Ψ, and at any given time step $t = 0,1,2,\ldots$, an agent perceives its *state* s_t and selects an *action* a_t. By exploring state space, an agent tries to learn the best control *policy*, $\pi : S \to A$, which maps states to actions by estimating Bellman error. The system (environment) responds by giving the agent some (possibly zero) numerical *reward* $r(s_t)$ and changing to state $s_{t+1} = \delta(s_t,a_t)$. Estimate approximation for reward represented as a vector $R \in \Re^{|s|}$, are incremented by $r_t\phi_t$ and $\phi_t(\phi_t - \gamma\phi_{t+1})$, where ϕ_t, ϕ_{t+1} are control variable vectors at time step t and $t+1$, and the transition probabilities under π can be represented as a matrix $P \in \Re^{|s|\times|s|}$. The state transition may be determined solely by the current state and the agent's action or may also involve stochastic processes. Given two Markov decision process M_1 and M_2 which share the same state space S but have two different action spaces A_1 and A_2, respectively, a new Markov decision process M_{12} can be defined by the composition of M_1 and M_2 such that at each time step, the learning agent selects one action from A_1 and one from A_2. The transition and cost function of the composite process M_{12} are defined by

$$\delta_{12}(s,(a_1,a_2)) = \delta_2(\delta_1(s,a_1),a_2) \quad \text{and} \quad \ldots (2)$$

$$r_{12}(s,(a_1,a_2)) = r_1(s,a_1) + \gamma r_2(\delta_1(s,a_1,a_2)). \quad \ldots (3)$$

3.2. TD Learning

The TD learning [7, 9] is an algorithm used to learn a value function. It is also called "bootstrapping," method, because the value is updated partly using an existing estimate and not a final reward. For how TD learning works, see **Algorithm 1** (step 2-6). The value function learned this way is an estimate of the expected total reward received for an episode from a given state s under policy π.