Neurocomputing 271 (2018) 38–47

Contents lists available at ScienceDirect

Neurocomputing

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

Experiments of conditioned reinforcement learning in continuous space control tasks

Borja Fernandez-Gauna[a,b], Juan Luis Osa[a,b], Manuel Graña[a,b,*]

[a] Computational Intelligence Group, University of the Basque Country, San Sebastian, Spain
[b] ACPySS, San Sebastian, Spain

ARTICLE INFO

Article history:
Received 24 December 2015
Revised 5 May 2016
Accepted 1 August 2016
Available online 6 July 2017

Keywords:
Reinforcement learning
Actor-Critic methods
Conditioned learning
Control wind turbines

ABSTRACT

The key issue that prevents application of Reinforcement Learning (RL) methods in complex control scenarios is lack of convergence to meaningful decision policies (i.e. policies that differ significatively from random decisions), due to the huge state-action spaces to be explored. Providing the agent with initial domain knowledge alleviates this problem. This is known as Conditioned RL (CRL). In high-dimensional continuous state-action space and reward domains, CRL is often the only feasible approach to reach meaningful decision policies. In these kind of systems, RL is carried out by Actor-Critic approaches, and the state-action value functionals are modeled by Value Function Approximations (VFA). CRL methods make use of an existing reference controller, i.e. the teacher controller, which provides the initial domain knowledge to the agent under training. The teacher-controller can be used in two ways to build the VFA of the state-action value and state transition functions which determine the action selection policy: (1) providing the desired output for a supervised learning process, or (2) directly using it to build them. We have carried out experiments to compare CRL methods, and unconditioned Actor-Critic agents in three different control benchmark scenarios. Results show that both agent conditioning approaches result in significant performance improvements. Undertight computational time constraints, CRL approaches were able to learn efficient policies, while the unconditioned agents were not able to find any acceptable policy in the benchmark control scenarios.

1. Introduction

Reinforcement Learning (RL) methods are gaining popularity as an alternative approach to traditional control techniques [1–10]. Among the most prominent and promising approaches are model-free online learning methods, which learn in a step-wise fashion by interacting with the environment to be controlled. They do not require an accurate model of the environment and, most importantly, they do not require the designer to have a great experience on the specific control task. The learning agent receives a reward as feedback and its goal is to maximize these rewards. Early RL algorithms assumed discrete sets of states and actions, which made them ill-suited for complex realistic control scenarios. In recent years, though, the focus of researchers has shifted to control tasks defined in multi-dimensional continuous state-action spaces. Learning continuous space domain functions is dealt with by Value Function Approximation (VFA) [11–13], which provides a mean of obtaining approximations of a continuous valued function from a discrete set of samples. Actor-Critic [14,15] designs are the best suited to work with continuous states and actions because they keep separate structures to represent the value function and the action selection policy being learned. Actor-Critic learn using *policy iteration*: first, the Critic estimates the value of the current policy using a *policy-evaluation* method, and then the Actor updates the policy using some *policy-improvement* method.

Current policy-evaluation methods focus on Least-Squares Temporal Difference methods, such as *LSTD* (λ) [16–18], whereas the state-of-the-art in policy-improvement methods is based on the natural gradient [19–21]. These methods are very efficient data-wise, but computationally very expensive and, not well-suited for high-dimensional state-action spaces [22], nevertheless they can be substituted succesfully using computationally inexpensive methods such as *TD* (λ) or *Continuous Action-Critic Learning Automaton* (CACLA).

RL methods face a very difficult issue when applied to high-dimensional continuous state-action spaces: how to find an acceptable policy from scratch. Absence of prior information means that the initial agent's policy will likely perform very poorly. Since RL methods aim to maximize the rewards received by exploring the state-action space, the likelihood of reaching an acceptable policy, i.e. a policy guaranteeing rewards above a threshold, is very low in

* Corresponding author.
E-mail address: manuelgrana@ehu.es (M. Graña).

http://dx.doi.org/10.1016/j.neucom.2016.08.155
0925-2312/

high-dimensional continuous domains. Thus, the agent needs to incorporate domain knowledge before it starts the state-action space exploration, this process is known as Conditioned RL (CRL). In this paper, we have compared two different CRL methods that provide an initial policy. Computational experiments were conducted on three different control tasks defined on continuous domains: airplane pitch control, underwater vehicle control, and Variable-Speed Wind-Turbine (VSWT) control. We have used CACLA with two different critics: the time difference *TD* (λ) and the recently proposed *True-Online TD* (λ) (*TOTD* (λ)). We have also tested VFAs with different feature dimensionality to assess its effect on function approximation accuracy, which remains an open question in recent literature [12,22].

The main contributions o reported in this paper are the following ones:

- Papers in the literature have very simplistic CRL methods working on toy examples, we compare two conditioning methods, assessing their impact on the policies learned by Actor-Critic learning architectures on realistic control tasks,
- Most papers in the literature assume an ad-hoc definition of the VFA parameters, we assess the impact on the Actor-Critic agent performance of the VFA resolution (i.e. number of features),
- We provide the first evaluation of the performance of the novel True Online Temporal-Difference policy evaluation algorithm in an Actor-Critic architecture compared to the classical Temporal-Difference (λ) policy evaluation method.

The contents of the paper are the following: First, in Section 2 we review the basic concepts and techniques involved in approaching RL of continuous control tasks [23]. In Section 3 we describe the learning and conditioning methods used in the experiments, and in Section 4 we review the main details of the benchmark control scenarios used as benchmarks. The results are discussed in Section 5 and, finally, Section 6 delivers the conclusions.

2. Background

Reinforcement Learning (RL) models the interaction between the agent and the environment as a Markov Decision Process (MDP) [14] defined by a tuple $\langle X, U, P, R\rangle$, where X is the state space, U is the action space, P is the stochastic state transition function $P: X \times U \times X \to [0, 1]$, and R is the reward function $R: X \times U \times X \to \mathbb{R}$. Both X and U can be discrete sets of states/actions or continuous state/action spaces, or a hybrid composition of discrete and continuous variables. Most of the early RL literature is devoted to discrete state-action spaces, however this representation becomes useless when tackling with complex feedback control tasks. Therefore, we assume in this paper that the input state space is $X \subseteq \mathbb{R}^n$ (being n the number of state variables), and that the action space is $U \subseteq \mathbb{R}^m$ (being m the number of control variables) equivalent to the output control variables [24]. In control applications, the goal of the learning agent is to learn an optimal policy $\pi^*(\mathbf{x})$: $X \to U$ that maximizes the expected reward return from state $\mathbf{x}$, known as the *value function* of state $\mathbf{x}$:

$$V^{\pi}(\mathbf{x}) = E^{\pi}\left\{\sum_{k=1}^{\infty} r_{t+k}\gamma^{k-1} | \mathbf{x}_t = \mathbf{x}\right\}, \tag{1}$$

where r_t is the reward received by the agent in time-step t, weighted by the discount parameter $\gamma \in (0, 1]$. This value function must be estimated in order to assess the wellness of a policy π. Online model-free learning methods learn from iterative interaction with the environment: At each time-step t, the state $\mathbf{x}_t$ is observed, the agent selects an action $\mathbf{u}_t$ receiving a reward r_t assessing the value of the new state $\mathbf{x}_{t+1}$ reached by the environment-agent system. At the end of each time-step, the agent updates its optimal policy estimation with the information provided by the last transition $\langle \mathbf{x}_t, \mathbf{u}_t, r, \mathbf{x}_{t+1}\rangle$.

Actor-Critic methods. The learning scheme in Actor-Critic architectures is a policy estimation iteration that updates the policy using a two-step procedure: first, the policy value function V^{π} estimate is updated (*policy evaluation*), secondly, the policy π is improved using the value estimate (*policy improvement*). The value function V^{π} and the policy π are represented in separate memory structures: the Critic learns V^{π}, whereas the Actor learns π. After updating the value function, the critic provides feedback to the actor. This feedback usually consists on the Temporal-Difference (TD) error $\delta_t = r_t + \gamma\hat{V}(\mathbf{x}_{t+1}) - \hat{V}(\mathbf{x}_t)$, where $\hat{V}(.)$ denotes the current estimation of the value function, so that the term $\left(\gamma\hat{V}(\mathbf{x}_{t+1}) - \hat{V}(\mathbf{x}_t)\right)$ is our current discounted estimation of the incremental value of the current state.

Value Function Approximation. We use Value Function Approximation (VFA) for the representation of a value function or policy with continuous inputs/outputs. VFAs use a set of feature activation functions $\phi : X \to \mathbb{R}^F$ mapping points in the input state space X into feature vectors $[\phi_1(\mathbf{x}), \phi_2(\mathbf{x}) \ldots \phi_F(\mathbf{x})]$, where F is the number of features used to represent the input state space. The most interesting VFAs are linear models (i.e. Radial Basis Function networks) because they offer some convergence properties over non-linear VFA models (i.e. neural networks) [12]. A linear VFA, approximation to a given function $f(\mathbf{x})$ can be expressed as $\hat{f}(\mathbf{x}) = \theta^T\phi(\mathbf{x})$, where θ is the linear combination parameter vector $\theta = [\theta_1, \theta_2, \ldots, \theta_F]$. These parameters θ_i are estimated by interpolation of the feature activation functions outputs over the complete continuous input space. The VFA may be used to model of any state-dependent function, such as the value function V^{π} or the Actor's policy π. Multivariate state functions, i.e., an actor that outputs several control variables, may be represented as a set of univariate policies: $\pi(\mathbf{x}) = \{\pi_1(\mathbf{x}), \pi_2(\mathbf{x}), \ldots, \pi_m(\mathbf{x})\}$ which are modeled by VFA independently.

By setting an activation threshold, the number of *active* features (with non-null activation values) for a given state $\mathbf{x}$ is commonly reduced, obtaining a sparse representation on a small subset of the features. This means that only a small number of features must be updated at each time-step. We have used a network of Gaussian Radial Basis Functions (RBF) $\phi_i(\mathbf{x})$, each of which is associated to a center point $\mathbf{c}_i$ in the input space X:

$$\phi_i(\mathbf{x}) = e^{-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{2\sigma^2}}, \tag{2}$$

where σ^2 controls the width of the Gaussian bell shaped function. We have distributed the center points uniformly over the input space forming a grid of $F = \prod_{i=1}^{n} F_i$ center points, where F_i is the number of feature centers used to approximate the i-th state variable. We illustrate this approximation with an example of a 2-input space function approximated as a grid of Gaussian RBFs with 3 features per state variable: Fig. 1a represents the 9 normalized activation functions ($F = F_1 * F_2 = 3 * 3 = 9$), and Fig. 1b represents the output surface of the VFA for some interpolation parameter vector θ.

Exploration. Model-free online learning methods must be able to explore different policies in order to improve the current one, so that the policy decision needs to introduce some perturbation to the actual greedy decision to choose the action with expected maximum value, i.e. the greedy decision. Therefore, learning processes need to balance *exploitation* and *exploration* in the decision making processes. Exploitation is carried out taking the action with the highest expected return, whereas exploration is carried out taking some other randomly chosen action. One way to control de balance between exploration and exploitation is by adding