Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 49-5 (2016) 285–290

Policy Derivation Methods for Critic-Only Reinforcement Learning in Continuous Action Spaces

Eduard Alibekov [*] **Jiri Kubalik** [*] **Robert Babuska** [*,**]

[*] *Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Prague, Czech Republic, {eduard.alibekov, jiri.kubalik@cvut.cz}@cvut.cz*
[**] *Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands, r.babuska@tudelft.nl*

Abstract: State-of-the-art critic-only reinforcement learning methods can deal with a small discrete action space. The most common approach to real-world problems with continuous actions is to discretize the action space. In this paper a method is proposed to derive a continuous-action policy based on a value function that has been computed for discrete actions by using any known algorithm such as value iteration. Several variants of the policy-derivation algorithm are introduced and compared on two continuous state-action benchmarks: double pendulum swing-up and 3D mountain car.

Keywords: reinforcement learning, continuous actions, multi-variable systems, optimal control, policy derivation

1. INTRODUCTION

Reinforcement Learning (RL) algorithms provide a way to optimally solve decision and control problems of dynamic systems (Sutton and Barto, 1998). An RL agent interacts with the system by measuring the states and applying actions according to a certain policy. After applying an action, the RL agent receives a scalar reward signal related to the immediate performance of the agent. The goal is to find an optimal policy, which maximizes the long-term cumulative reward.

The available RL algorithms can be broadly classified into critic-only, actor-only, and actor-critic methods (Konda and Tsitsiklis, 2000). Critic-only methods first find the optimal value function (abbreviated as V-function) and then derive an optimal policy from this value function. In contrast, actor-only methods search directly in the policy space. The two approaches can be combined into actor-critic architectures, where the actor and critic are both represented explicitly and learned simultaneously. The critic learns the value function and based on that it determines how the policy should be changed. Each class can be further divided into model-based and model-free algorithms, depending on the use of a system model throughout the learning process.

In this paper we consider the critic-only, model-based variant of RL in continuous state and action spaces. The typical learning process, depicted in Figure 1, consists of three steps:

(1) Data collection – using a model of the system or the system itself, samples in the form $(x_k, u, x_{k+1}, r_{k+1})$ are collected. Here, x_k is the system state, u is the control input (action), x_{k+1} is the state that the system reaches from state x_k after applying action u, r_{k+1} is the immediate reward for that transition.
(2) Computation of the optimal V-function – based on the samples, an approximation of the V-function is learnt, which for each system state predicts the cumulative long-term reward obtained under the optimal policy.
(3) Policy inference – based on the computed V-function, the policy is derived at each sampling time (or simulation step), so that the system can be controlled.

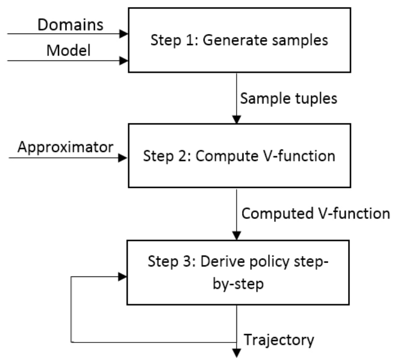

Fig. 1. Model-based critic-only reinforcement learning.

In this paper we address only the third step. Assuming that a parameterized approximation of the true unknown V-function has been computed, we derive the policy for a general, continuous-action input space. We aim at the maximization of the long-term reward and at the computational efficiency, so that the method can be applied to multidimensional action spaces.

The paper is organized as follows. Section 2 gives a brief introduction to reinforcement learning. The state-of-the-art and the proposed policy derivation methods are described in Section 3, experimentally demonstrated in Section 4 and discussed in detail in Section 5. Finally, Section 6 concludes the paper.

2. PRELIMINARIES

Define an n-dimensional state space $\mathcal{X} \subset \mathbb{R}^n$, and m-dimensional action space $\mathcal{U} \subset \mathbb{R}^m$. The model is described by the state transition function $x_{k+1} = f(x_k, u)$, with $x_k, x_{k+1} \in \mathcal{X}$ and $u \in \mathcal{U}$. The reward function assigns a scalar reward $r_{k+1} \in \mathbb{R}$ to the state transition from x_k to x_{k+1}:

$$\begin{aligned} x_{k+1} &= f(x_k, u) \\ r_{k+1} &= r(x_k, u) \end{aligned} \tag{1}$$

Define a finite set of discrete control input values $U = \{u_1, u_2, \ldots, u_N\}$ drawn from $\mathcal{U}$. The V-function can be computed by solving the Bellman equation:

$$V(x) = \max_{u \in U} [r(x, u) + \gamma V(f(x, u))] \tag{2}$$

where γ is the discount factor (a user-defined parameter). There are several methods to approximate the V-function for continuous state spaces. In this paper, we use the fuzzy V-iteration algorithm (Busoniu et al., 2010) as it is guaranteed to converge and the fuzzy approximator allows us to interpret the values at each fuzzy set core directly as the V-function value. The approximation of the V-function after convergence is denoted by $\hat{V}(x)$. The policy is defined by the following mapping:

$$h : \mathcal{X} \to \mathcal{U} \tag{3}$$

The most straightforward way to derive a policy corresponding to the approximate value function $\hat{V}(x)$ is (Bertsekas and Tsitsiklis, 1996):

$$\hat{h}(x) \in \arg\max_{u \in U} \left[r(x, u) + \gamma \hat{V}(f(x, u)) \right] \tag{4}$$

However, this policy will be discrete-valued and generally will not perform well on control problems with continuous actions. Therefore, we propose alternative methods, whose aim is to provide a better policy than (4).

3. POLICY DERIVATION METHODS

The goal is to find a continuous-action policy based on the computed approximate V-function. The measure to be maximized by the policy is the average of the rewards obtained during a long-horizon control experiment. Using long-horizon experiments allows us to effectively measure the steady state error. The policy derivation problem can then be formulated as the following maximization:

$$\begin{gathered} \hat{h}(x) = \arg\max_{h(x)} \left[\lim_{N \to \infty} \frac{1}{N} \left(\sum_{k=1}^{N-1} r(x_k, h(x_k)) \right) \right] \\ \text{with } x_k = f(x_{k-1}, h(x_{k-1})) \quad \forall x_0 \in \mathcal{X} \end{gathered} \tag{5}$$

where N is the number of steps in the control experiment and the limit is assumed to exist. In the sequel we present three different policy-derivation methods.

3.1 Related work

The problem of deriving policies for continuous-action spaces has not been sufficiently addressed in the literature. The most common approach is to discretize the action space, compute the value for all the discrete actions, and select the one that corresponds to the largest V-function value. One of the earliest references to this approach is (Santamaria et al., 1996). A drawback of this method is that the exhaustive search over all available action combinations quickly becomes intractable as the size of the action space grows.

Another similar approach is based on sampling (Sallans and Hinton, 2004; Kimura, 2007). Using Monte-Carlo estimation, this approach can find a near-optimal action without resorting to exhaustive search over a discretized action space. However, for a good performance this method requires a large number of samples and is therefore computationally inefficient.

A different approach relies on translating the continuous action selection step into a sequence of binary decisions. Each decision whether to decrease or increase the control action eliminates a part of the action space. This process stops once a predefined precision is reached. More details can be found in (Pazis and Lagoudakis, 2009). The main drawback of this algorithm is that its performance quickly degrades as the dimensionality of the state and input space grows (Pazis and Lagoudakis, 2011).

3.2 Evaluation over a fine grid of actions

This method relies on a fine resampling of the action space *a posteriori* (i.e., after learning the V-function). Define

$$A = \bar{U}_1 \times \bar{U}_2 \times \cdots \times \bar{U}_m, \quad A \subseteq \mathcal{U} \tag{6}$$

where each set $\bar{U}_i$ contains points equidistantly distributed along the ith dimension of the action space. Set A therefore contains all combinations of the control inputs for which the value function is evaluated and the result for the current state x is stored in array G:

$$G[u] = r(x, u) + \gamma \hat{V}(f(x, u)) \tag{7}$$

An example of the result for a two-dimensional action space for one specific x is shown in Fig. 2.

This additional sampling allows to control the system by applying actions more precisely, while it does not require extensive computation during learning. The policy derivation step is formalized in Algorithm 1.

3.3 Chebyshev polynomial approximation over the action space

The main idea of this method is that a smooth approximation over the action space provides more accurate control and avoids chattering in the case of unstable system equilibria. We use Chebyshev polynomials of the first kind, which are defined by the following recurrent relation:

$$\begin{aligned} T_0(\bar{u}) &= 0 \\ T_1(\bar{u}) &= \bar{u} \\ T_{n+1}(\bar{u}) &= 2\bar{u}T_n(\bar{u}) - T_{n-1}(\bar{u}) \end{aligned}$$