Reinforcement Learning in Continuous Action Spaces through Sequential Monte Carlo Methods

Avinash Kumar Chaurasia
Computer Science
Paderborn University
Paderborn, Germany
avinashk@mail.uni-paderborn.de

***Abstract*—In real-world domains, dealing with continuous state and action spaces is fairly conventional. While many approaches to applying Reinforcement Learning algorithms to continuous state problems have been recommended, similar techniques are pretty inconceivable when it comes to applying them to continuous action spaces, where, in addition to computing a good approximation of the value function, a fast method for identifying the highest-valued action is indispensable. In this paper, an innovative actor-critic methodology is proposed in which the policy taken by the actor is estimated using Sequential Monte Carlo methods. In this approach, the importance sampling stage is based on the critic's learned values, whereas the resampling step changes the actor's policy. The suggested approach has been analytically compared to various learning algorithms in a variety of domains and, in this paper, results from a control problem have been presented, in which a boat must be steered across a river.**

***Index Terms*—Sequential Monte Carlo, Reinforcement Learning, Markov Decision Processes, State–action–reward–state–action.**

I. Problem Analysis

The majority of the research on Reinforcement Learning (RL) [1] has focused on providing solutions to Markov Decision Processes (MDP's). Learning in real-world settings, on the other hand, essentially requires dealing with continuous action and spaces. Although many studies have been done to deal with problems in continuous spaces, the same attention was not given to tasks involving continuous actions. While various tasks can be (suboptimally) solved by coarsely discretizing the action variables, problems requiring high-precision control and actions slightly different from the optimal one resulting in very low utility values often require a different approach. Because RL algorithms have to experience each available action several times in order to estimate its utility, utilizing highly fine discretizations may be prohibitively costly for the learning process [2].

Several approaches to mitigating these problems have adopted the actor-critic architecture [3] [4]. The main idea behind the actor-critic approach is to clearly describe the policy (which is stored by the actor) with a memory structure that is independent of the one used for the value function (stored by the critic). The agent's policy in a given state is a probability distribution over the action space, which is normally represented by parametric functions such as Gaussians [5], neural networks [6] and fuzzy systems [7]. The critic's role is to criticize the actor's actions based on the estimated value function, and as a result, the actor alters its policy via a stochastic gradient on its parameter space. By adopting this approach, the actor gradually modifies its policy so that actions with higher utility values are chosen more frequently until the learning process converges to the optimal policy [2].

In this paper, Lazaric et al. [2] have proposed to approximate the sequence of probability distributions implemented by the actor using the Sequential Monte Carlo (SMC) method [8], hence resulting in a novel actor-critic algorithm called SMC-learning. The actor depicts its stochastic policy with a finite set of random samples (i.e., actions) instead of using a parametric function, that evolves according to the values stored by the critic through simple resampling and moving mechanisms. Actions are first selected from a prior distribution, and then they are resampled per an importance sampling estimate based on the critic's learned utility values. The set of possible actions becomes increasingly more crowded around actions having larger utilities as a result of the resampling and moving steps, encouraging a deep study of the most promising action-space areas, allowing SMC-learning to uncover true continuous actions [2].

II. Introduction to Reinforcement Learning

An agent interacts with an unknown environment in reinforcement learning problems. The agent observes the state, takes an action, and is then rewarded at every time step. The purpose of the agent in RL algorithms is to learn a policy(mapping from states to actions) that maximizes the long-term value of the system [2]. Markov Decision Processes (MDP) are mathematical frameworks for describing an environment in RL, and they may be used to formulate practically all RL problems. A set of finite environment states $\mathcal{S}$, a set of feasible actions A(s) in each state, a real valued reward function R(s), and a transition model P(s' | s, a) comprise an MDP. An agent's policy is defined by a probability distribution $\pi(a \mid s)$ which determines the likelihood of taking action an in state s. As given in [2], the action value function $Q^\pi(s,a) = E\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s = s_1, a = a_1, \pi\right]$, where $r_1 = \mathcal{R}(s,a)$ formalizes the utility of taking an action in state s and then following a policy. The aim of RL methodologies is to maximize the action-value function in every state. The Bellman equation: $Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{T}(s,a,s') \max_{a'} Q^*(s',a')$ can be solved in order to compute the optimal value-function [2].

III. Proposed Algorithm

The actor-critic architecture underpins the SMC Learning in which the actor maintains a density distribution $\pi^t(a \mid s)$ for each state s that determines the agent's policy at time instant t. Initially, when there is no prior knowledge of the problem, the actor normally considers a uniform distribution over the action space at the start of the learning process, resulting in an exploratory strategy. The critic gathers data for the estimation of the value function as the learning process advances and informs the actor about the most promising actions. The actor, on the other hand, adjusts its policy to increase performance and gradually minimize the exploration to reach the best policy. As defined in paper [2], the concept of SMC learning is as follows: for each state s, N samples from a proposal distribution $\pi^0(a \mid s)$ are used to establish the set of available actions A(s):

$$\mathcal{A}(s) = \{a_1, a_2, \cdots, a_N\}, \quad a_i \sim \pi^0(a \mid s)$$

. Each sampled action a_i is assigned an importance weight $w_i \in \mathcal{W}(s)$ with a value of 1/N, allowing the prior density to be approximated as [2]:

$$\pi^0(a \mid s) \simeq \sum_{i=1}^{N} w_i \cdot \delta(a - a_i)$$

where $a_i \in \mathcal{A}(s)$,w$_i \in \mathcal{W}(s)$,δ represents the Dirac Delta measure. As the quantity of samples grows to infinity, this representation becomes functionally identical to the original probability density function's functional description. This means that the actor can approximate the policy given by the density $\pi^0(a \mid s)$ by randomly selecting actions from $\mathcal{A}(s)$, where the (normalized) weights are the selection probabilities. It is possible to define the appropriate probability distribution across the continuous action space, commonly referred to as the target distribution, given the continuous action-value function assessed by the critic and a suitable exploration approach (e.g., the Boltzmann exploration). As the learning process continues, the critic's predicted action values become more and more reliable, and the agent's policy should evolve to choose more frequent actions with higher utilities. This means that the target distribution changes in each state as a result of the knowledge gathered throughout the learning process, and the actor must adjust its approximation accordingly [2].

SMC methods can be used to approximate sequences of probability distributions using random samples that are generated over time using importance sampling and resampling techniques when no information about the shape of the target distribution is provided. The goal of importance sampling is to adjust the sample weights to account for variations between the target distribution p(x) and the proposal distribution π(x) that was used to produce the samples. The discrete weighted distribution $\sum_{i=1}^{N} w_i \cdot \delta(x - x_i)$ improves the target distribution's approximation by setting the each weight W_i corresponding to the ratio $p(x_i)/\pi(x_i)$. The actor now performs the important sampling phase, which updates the weights of the actions based on the utility values approximated by the critic. When the normalized weights of some samples are either tiny or extremely large, the target density differs dramatically from the proposal density used to generate the samples. This also indicates that the set of available actions contains a handful of samples with an extremely low estimated utility. To avoid such a situation, the actor must resample new actions from the current weighted approximation of the target distribution to expand the set of accessible actions [2].

As depicted in fig 1, SMC-learning iterates through three key main steps: the actor's action selection, the critic's update of the action-value function, and lastly the actor's policy update. The following sections provide an overview of the three steps:

```
for all $s \in \mathcal{S}$ do
    Initialize $\mathcal{A}(s)$ by drawing $N$ samples from $\pi^0(a|s)$
    Initialize $\mathcal{W}(s)$ with uniform values: $w_i = 1/N$
end for
for each time step $t$ do
    Action Selection
    Given the current state $s_t$, the actor selects action $a_t$ from $\mathcal{A}(s_t)$ according to $\pi^t(a|s) = \sum_{i=1}^{N} w_i \cdot \delta(a - a_i)$
    Critic Update
    Given the reward $r_t$ and the utility of next state $s_{t+1}$, the critic updates the action value $Q(s_t, a_t)$
    Actor Update
    Given the action-value function, the actor updates the importance weights
    if the weights have a high variance then
        the set $\mathcal{A}(s_t)$ is resampled
    end if
end for
```

Fig 1: SMC Learning Algorithm [2]

A. Action Selection

In SMC-learning, the actor executes the action selection by randomly selecting one action from those accessible in the current state to select the best possible action in the current state, given the estimated action-value function [2].

B. Critic Update

Among the variety of function approximation approaches that may be used for estimating action-value function, the Lazaric et al. [2] use a straightforward approach, for each action accessible in state s, the critic maintains an action value, $\mathcal{Q}$(s, a_i), and adjusts it according to Temporal Difference (TD) update rules [9]:

$$Q(s_t, a_t) \leftarrow (1-\alpha) Q(s_t, a_t) + \alpha u(r_t, a_{t+1}, s_{t+1}),$$

where $\alpha \in$ [0,1] represents learning rate and $u(r_t, a_{t+1}, s_{t+1}) = r_t + \gamma \mathcal{Q}(s_{t+1}, a_{t+1})$ is the target utility. TD algorithms helps in computing $\mathcal{Q}^*(s, a)$ by interacting directly with the environment.

C. Actor Update

This is one of the most important steps in SMC-learning in which the actor updates its policy. The actor updates the weights W_i using the importance sampling principle, thereby conducting a policy improvement step based on the action values estimated by the critic hence, the actions with higher estimates are given more weight in this way. The Lazaric et al. [2] have adopted the Boltzmann exploration strategy to update the weights for the possible actions [9]. The Boltzmann exploration strategy provides the mechanism that favors activities that have greater projected utility values.

The agent's policy has changed since the weights have been updated. However, because the optimal action may not be available, it may not be feasible to solve continuous action MDPs optimally by exploring only a finite collection of actions taken from a prior distribution. As a result, to improve the distribution of the samples on the action domain, a resampling step may be required in the SMC approach after the importance sampling phase. The purpose of doing resampling is to replace small-weight samples with new samples that are near in weight to large-weight samples, reducing the weight gap [2].

IV. Experiment using SMC Learning vs Other Algorithms

Lazaric et al. [2] have analyzed the boat problem presented in [7] to examine the properties of SMC-learning and compare its performance to that of other RL techniques. The challenge is to learn how to maneuver a boat from a river's left bank to its right bank wharf while dealing with a strong non-linear stream. The coordinates of the boat, x, and y, are specified in the range [0,200] while the desired direction U is set in the range [-90°,90°] [2].

As shown in Fig 2, the performance of SMC-learning in terms of boat problems has been compared with three other RL algorithms: State–action–reward–state–action (SARSA) with Boltzmann exploration with different discretizations of the action space, SARSA with tile coding [9] and Continuous Q-learning [10]. It is visible from the

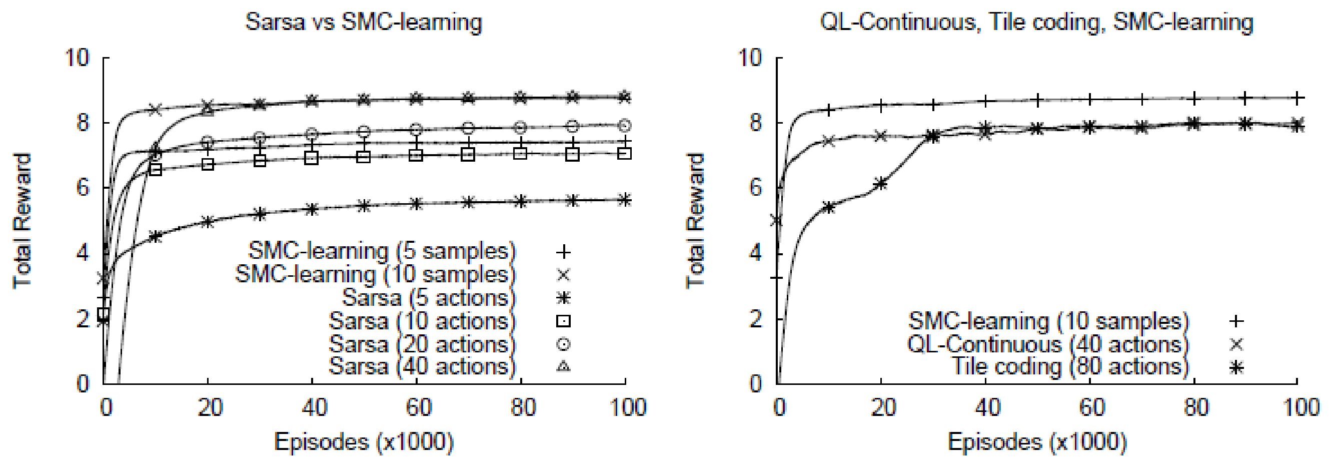

Fig 2: SMC-learning vs. SARSA (left), SMC-learning vs. tile coding (right), and Continuous Q-learning vs. SMC-learning (right) [2]

graph representing SMC-learning vs SARSA(left), that the performance and convergence time of SMC-learning even with only 5 samples surpasses SARSA with 5 and 10 actions [2].

SMC-learning, SARSA with tile coding utilizing two tilings and a resolution of 2.25° (corresponding to 80 actions), and Continuous Q-learning with 40 actions are shown in Fig 2-right. We can notice from the figure that SMC-learning surpasses the other two algorithms. Because of the non-linearity of the system's dynamics, tile coding's generalization over the action space harms learning performance. The agent is prevented from learning which of the various acts is the best by generalizing over them. Correspondingly Continuous Q-learning is strictly related to the actions provided by the designer and to the implicit assumption of linearity of the action-value function [2].

V. My opinion and Conclusion

Lazaric et al. [2] have proposed a novel actor-critic algorithm for solving continuous action problems in this study. The algorithm is built on a Sequential Monte Carlo approach, which allows the actor to express the current policy using a finite set of potential actions coupled with weights that are updated using the critic's utility values. We have discussed in the paper that It is often impossible to do a full search in a continuous action space to identify the best action. Several ways to overcome this challenge limit their search to a finite number of points. Algorithms such as tile coding and interpolation-based must make (sometimes implicit) assumptions about the shape of the value function to keep this number low. On the other hand, in SMC-learning, the algorithm does not require any assumptions about the shape of the value function, it is model-free, and it can learn to obey stochastic policies as well (needed in multi-agent problems). Furthermore, based on the experimental results of the boat problem, we have seen that SMC-learning takes less than one-sixth of the trails needed by the SARSA algorithm for convergence, thanks to initial resampling done on the available actions.

References

[1] Alex M. Andrew. Reinforcement learning: An introduction by richard s. sutton and andrew g. barto, adaptive computation and machine learning series, mit press (bradford book), cambridge, mass., 1998, xviii 322 pp, isbn 0-262-19398-1, (hardback, £31.95). *Robotica*, 17(2):229–235, 1999.

[2] Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Reinforcement learning in continuous action spaces through sequential monte carlo methods. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[3] Vijay R. Konda and John N. Tsitsiklis. Onactor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[4] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225, 2006.

[5] Hajime Kimura and Shigenobu Kobayashi. Reinforcement learning for continuous action using stochastic gradient ascent. *Intelligent Autonomous Systems (IAS-5)*, pages 288–295, 1998.

[6] Hado van Hasselt and Marco A. Wiering. Reinforcement learning in continuous action spaces. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 272–279, 2007.

[7] L. Jouffe. Fuzzy inference system learning by reinforcement methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(3):338–355, 1998.

[8] Jun S. Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[10] José Del R Millán, Daniele Posenato, and Eric Dedieu. Continuous-action q-learning. *Machine Learning*, 49(2):247–265, 2002.