DOP: Deep Optimistic Planning with Approximate Value Function Evaluation

Francesco Riccio
Sapienza University of Rome
Via Ariosto 25, Rome, Italy
riccio@diag.uniroma1.it

Roberto Capobianco
Sapienza University of Rome
Via Ariosto 25, Rome, Italy
capobianco@diag.uniroma1.it

Daniele Nardi
Sapienza University of Rome
Via Ariosto 25, Rome, Italy
nardi@diag.uniroma1.it

ABSTRACT

Research on reinforcement learning has demonstrated promising results in manifold applications and domains. Still, efficiently learning effective robot behaviors is very difficult, due to unstructured scenarios, high uncertainties, and large state dimensionality (e.g. multi-agent systems or hyper-redundant robots). To alleviate this problem, we present DOP, a deep model-based reinforcement learning algorithm, that attacks the curse of dimensionality and reduces the computational demand of the planning process while achieving good performance.

KEYWORDS

Robot Learning; Reinforcement Learning; Deep Reinforcement Learning; Planning

1 INTRODUCTION

Action planning in robotics is a complex task due to unpredictabilities of the physical world, uncertainties in the observations, and rapid explosions of the state dimensionality. For example, hyper-redundant manipulators are typically affected by the curse of dimensionality problem when planning in large state spaces. Similarly, in multi-robot collaborative tasks, each robot has to account for both the state of the environment and other robots’ states. Due to the curse of dimensionality, generalization and policy generation are time consuming and resource intensive. While deep learning approaches led to improved generalization capabilities and major successes in reinforcement learning [7–9] and robot planning [5, 6], most techniques require huge amounts of data collected through agent’s experience. In robotics, this is often achieved by spawning multiple simulations [14] in parallel to feed a single neural network. During simulation, however, the robot explores its huge search space, with little or no prior knowledge. While encoding prior information in robot behaviors is often desirable, this is difficult when using neural networks and it is mostly achieved through imitation learning [2, 16] with little performance guarantees. To overcome this issue, planning techniques, such as Monte-Carlo tree search [1], have been applied in literature. Unfortunately, these methods fail in generalizing and show limitations in relating similar states [15]. As in prior work [11], we attack the generalization problem in policy generation by enhancing the Upper Confidence Tree (UCT) algorithm [4] with an external action-value function approximator, that selects admissible actions and consequently drives the node-expansion phase during episode simulation. In this paper, we extend the algorithm in [11] to use a more powerful representation, based on deep learning, that enables better generalization and supports higher dimensional problems. The extended version of this paper is available in [10]1. In fact, DOP (Deep Optimistic Planning), is based on Q-learning and allows agents to plan complex behaviors in scenarios characterized by discrete action spaces and large state spaces. To model action values, we use a convolutional neural network (CNN) that is iteratively refined by aggregating [13] samples collected at every timestep. DOP generates action policies by running a Monte Carlo tree search [17] and incrementally collecting new samples that are used to improve a deep Q-network approximating action values. We aim at demonstrating that DOP can efficiently be used to generalize policies and restrict the search space to support learning in high-dimensional state spaces. Our contribution consists in an extension of prior work [11] to use deep learning and improve both the focused exploration and generalization capabilities.

2 DOP

As in previous literature [8, 9], we choose to change the representation adopted in [11] and approximate the action values using a deep neural network. The input of the network is an image capturing the state of the environment, and its output is the Q value for each action. Differently from DQN, we perform a data aggregation [13] procedure, where all the transitions are iteratively collected, aggregated and used at training time. Specifically, at each iteration i we collect a dataset $D^i = \{x\}$ of transitions experienced by the agent, and we aggregate it into $D^{0:i} = \bigcup_{d=0}^i D^d$ that is used for learning. The aggregated dataset is used to minimize the $\ell_2$-loss:

$$\ell_2(r_{t+1} + \gamma \max_{a'} Q^i(s_{t+1}, a'), Q^i(s_t, a_t)),$$

(1)

where $s$, $a$, and $r$ represent the state, action and reward signal at timestep $t$ respectively. $\gamma$ is the discount factor and $\theta$ is the set of parameters of the network. The optimization is performed using an Adam [3] optimizer. DOP is an iterative algorithm (see Algorithm 1, for major details [10]) that, at each iteration $i = 1 \ldots I$, (1) generates

1https://arxiv.org/abs/1803.08501
Robotics Track Extended Abstract

Algorithm 1: DOP

begin
for $i = 1$ to $I$ do
   $s_0 \leftarrow$ random state from $A$.
for $t = 1$ to $T$ do
   1) Get state $s_t$ by executing $\pi^{t-1}(s_{t-1})$.
   2) $D^t \leftarrow$ UCTDOP$(s_t, \lambda_0, \epsilon, \theta)$. 
   3) $D^{t+1} \leftarrow D^t \cup D^{t+1}$.
   4) $Q_0$.UPDATE($D^{t+1}, \alpha, \gamma$).
   5) $\pi^t(s) \leftarrow \arg \max_a Q_0(s, a)$.
return $\pi^T$.

1) evaluates a subset of “admissible” actions $\hat{A} \subseteq A$ in $s_{t+1(0)}-1$, that are determined according to $Q_0(s_{t+1(0)}, a)$ such that

$$Q_0(s_{t+1(0)}, a) = \lambda \max Q_0(s_{t+1(0)}, a) + \epsilon_i$$

where $\lambda$ is typically initialized to 0.5. Through $\epsilon_i$, a certain amount of exploration is guaranteed;

2) selects and executes the best action $a^*_{t+1(0)} \in \hat{A}$ according to

$$a^*_{t+1(0)} = \arg \max_a Q_0(s_{t+1(0)}, a) + C \sqrt{\frac{\log \sum_a \eta(s_{t+1(0)}, a)}{\eta(s_{t+1(0)}, a)},}$$

where $C$ is a constant that multiplies and controls the exploration term $\epsilon_i$, and $\eta(s_{t+1(0)}, a)$ is the number of occurrences of $a$ in $s_{t+1(0)}$;

3) runs $M$ roll-outs by executing an $\epsilon$-greedy policy based on $\pi^{t-1}$ until a terminal state is reached.

DOP uses UCTDOP as an expert and collects, a dataset $D^t$ of $H$ transitions experienced in simulation. After each UCT run, $D^t$ is aggregated into $D^{t+1} = D^t \cup D^{t+1}$ [12, 13], and the dataset is used to perform updates of the $Q$-network, as illustrated in previous section. Finally, once $Q_0$ is updated, the policy is generated as to maximize the action values: $\pi^t(s) = \arg \max_a Q_0(s, a)$.

3 SELECTED EXPERIMENT

We evaluate DOP in different robotic applications, however, due to lack of space, we restrict here to the evaluation of a fetching task whitt a KUKA 7-DOF robotic arm (a complete set of the experiments and discussion is in [10]). We compare against DQN [9], TD-search [15] and both a vanilla-UCT and random-UCT implementations. We refer to vanilla-UCT as the standard UCT algorithm that always expands every possible action in $A_j$, for every agent $j$. Random-UCT, instead, is a naive algorithm where at each step of the Monte-Carlo search one action is randomly expanded. We evaluate the cumulative reward obtained during different executions of DOP against the number of explored states and iterations of the algorithms. In this scenario, the state space is represented through an image collected by an overlooking camera. The robot can perform 10 actions to translate and rotate its end-effector in the environment. The reward function is shaped and it is computed as a weighted sum of four components: the first is inversely proportional to the Euclidean distance of the end-effector to the target, the second it proportional to the distance to the virtual center of the obstacle, the third and the fourth are inversely proportional to the pitch and yaw angle respectively. In this way the reward function promotes states that are near the target, far from the obstacle, and with the end-effector oriented upwards – to fetch objects with a preferred orientation, e.g. glass full of water. It is important to notice that, since first iterations and with a reduced set of training samples, DOP is able to outperform other algorithms that need a huge training set to learn competitive policies, e.g. DQN. Still, vanilla-UCT shows comparable rewards, but the number of explored states for this algorithm is –65% larger than DOP.

4 CONCLUSION

DOP is an iterative algorithm that uses action values learned through a deep Q-network to guide and reduce the exploration of the state space in high-dimensional scenarios. Our key contribution consists in an extension of Q-CP [11] to use deep learning and improve both the focused exploration and the generalization of the algorithm. Our future work points towards applications for continuous and online learning, where focused exploration is key to further improve the performance of the system.
REFERENCES


