DOP: Deep Optimistic Planning with Approximate Value Function Evaluation

Robotics Track

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

ACM Reference Format:
Francesco Riccio, Roberto Capobianco, and Daniele Nardi. 2018. DOP: Deep Optimistic Planning with Approximate Value Function Evaluation. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

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 $\mathrm{D}^i = \{x\}$ of transitions experienced by the agent, and we aggregate it into $\mathrm{D}^{0:i} = \{\cup \mathrm{D}^d | d = 0 \ldots i\}$, that is used for learning. The aggregated dataset is used to minimize the ℓ_2-loss:

$$\ell_2(r_{t+1} + \gamma \max_{a'} Q_\theta(s_{t+1}, a'), Q_\theta(s_t, a_t)), \tag{1}$$

where s, a and r represent the state, action and reward signal at timestep t respectively. γ is the discount factor and θ 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

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

[1] https://arxiv.org/abs/1803.08501

Algorithm 1: DOP

```
begin
    for $i = 1$ to $I$ do
        $s_0 \leftarrow$ random state from $\Delta$.
        for $t = 1$ to $T$ do
1)          Get state $s_t$ by executing $\pi^{i-1}(s_{t-1})$.
2)          $\mathrm{D}^i \leftarrow \mathrm{UCT_{DOP}}(s_t, \lambda_0, \epsilon_{\tilde{A}})$.
3)          $\mathrm{D}^{0:i} \leftarrow \mathrm{D}^i \cup \mathrm{D}^{0:i-1}$.
            $Q_\theta$.UPDATE($\mathrm{D}^{0:i}$, $\alpha$, $\gamma$).
4)          $\pi^i(s) \leftarrow \arg\max_a Q_\theta(s, a)$.
    return $\pi^I$
```

a new policy π_i which improves π_{i-1} and (2) learns action values that are used at planning time to reduce the search space. The agent executes an UCT search, where admissible actions are selected through Q value estimates, and incrementally collects new samples that are used to improve Q value estimates. More in detail, at each iteration, DOP performs the following algorithminc steps. First, the agent follows its policy π^{i-1} for T timesteps, generating a set of T states $\{s_t\}$. Then, for each generated state s_t, the agent runs a modified UCT ($\mathrm{UCT_{DOP}}$) search [4] with depth H. Specifically, at each $h = 1 \dots H$, the $\mathrm{UCT_{DOP}}$ algorithm

(1) evaluates a subset of "admissible" actions $\tilde{\mathrm{A}} \subseteq \mathrm{A}$ in $s_{t+(h-1)}$, that are determined according to $Q_\theta(s_{t+(h-1)}, a)$ such that

$$Q_\theta(s_{t+(h-1)}, a) >= \lambda \max_a Q_\theta(s_{t+(h-1)}, a) + \epsilon_{\tilde{A}} \tag{2}$$

where λ is typically initialized to 0.5. Through $\epsilon_{\tilde{A}}$, a certain amount of exploration is guaranteed;

(2) selects and executes the best action $a^*_h \in \tilde{\mathrm{A}}$ according to

$$a^*_h = \arg\max_a Q_\theta(s_{t+(h-1)}, a) + C \cdot \sqrt{\frac{\log(\sum_a \eta(s_{t+(h-1)}, a))}{\eta(s_{t+(h-1)}, a)}}, \tag{3}$$

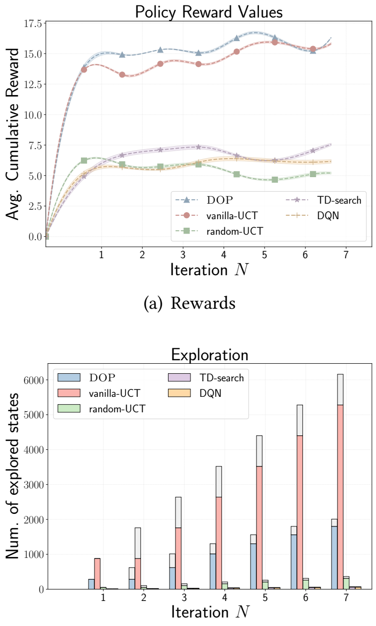
where C is a constant that multiplies and controls the exploration term e, and $\eta(s_{t+(h-1)}, a)$ is the number of occurrences of a in $s_{t+(h-1)}$.

(3) runs M roll-outs by executing an ϵ-greedy policy based on π^{i-1} until a terminal state is reached.

DOP uses $\mathrm{UCT_{DOP}}$ as an expert and collects, a dataset D^i of H transitions experienced in simulation. After each UCT run, D^i is aggregated into $\mathrm{D}^{0:i} = \mathrm{D}^i \cup \mathrm{D}^{0:i-1}$ [12, 13], and the dataset is used to perform updates of the Q-network, as illustrated in previous section. Finally, once Q_θ is updated, the policy is generated as to maximize the action values: $\pi^i(s) = \arg\max_a Q_\theta(s, a)$.

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* whit 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.



(a) Rewards

(b) States

Figure 1: The fetching task. (a) reports the avg. cumulative reward, while (b) number of explored states obtained by DOP, DQN, TD-search, *random-UCT* and *vanilla-UCT*.

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

[1] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (2012), 1–43.

[2] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard. 2010. Learning and Reproduction of Gestures by Imitation. *IEEE Robotics Automation Magazine* 17, 2 (June 2010), 44–54. https://doi.org/10.1109/MRA.2010.936947

[3] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[4] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. *Machine learning: ECML 2006* (2006), 282–293.

[5] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17, 39 (2016), 1–40.

[6] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2016. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research* (2016), 0278364917710318.

[7] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[8] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. 1928–1937.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (26 02 2015), 529–533. http://dx.doi.org/10.1038/nature14236

[10] F. Riccio, R. Capobianco, and D. Nardi. 2018. DOP: Deep Optimistic Planning with Approximate Value Function Evaluation. *ArXiv e-prints* (mar 2018). arXiv:cs.RO/1803.08501

[11] Francesco Riccio, Roberto Capobianco, and Daniele Nardi. 2018. Q-CP: Learning Action Values for Cooperative Planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018-01-30). –.

[12] Stephane Ross and J Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979* (2014).

[13] Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics*. 627–635.

[14] Andrei A Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. 2016. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286* (2016).

[15] David Silver, Richard S Sutton, and Martin Müller. 2012. Temporal-difference search in computer Go. *Machine learning* 87, 2 (2012), 183–219.

[16] Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817* (2017).

[17] Mark H. Winands, Yngvi Björnsson, and Jahn-Takeshi Saito. 2008. Monte-Carlo Tree Search Solver. In *Proceedings of the 6th International Conference on Computers and Games (CG '08)*. Springer-Verlag, Berlin, Heidelberg, 25–36. https://doi.org/10.1007/978-3-540-87608-3_3