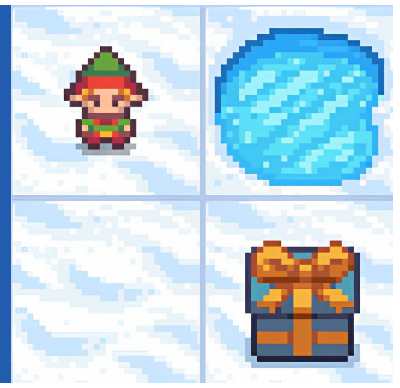
ETH zürich

Project Number: 6

Deployment-Efficient RL: A Practical Implementation



Thomas Creavin, Adam Klebus, Ke Li, Jiawei Huang (Supervisor)
Foundations of Reinforcement Learning, ETH Zurich

1 Background and Motivation

- Deploying a new policy can be costly and time-consuming.
- Frequent deployment is not suitable for robotics [1], healthcare [2], and recommender systems [3] applications due to risks and costs.
- New algorithms with theoretical guarantees have been proposed [4] to achieve good policies in a few deployments.

2 Problem Statement

An algorithm has deployment complexity K if, for an arbitrary MDP, the algorithm can:

- Return optimal policy with probability after K policy deployments;
- Collect trajectories in each deployment with a constraint that is polynomial in standard parameters.

For this setting, we're concerned with a time-dependent episodic linear MDP. Deployment complexity here depends on the dimension d of the feature mapping ϕ and time horizon H.

Our primary goal is to implement an RL agent that can learn a good policy while only using approx. 10 policy deployments based on the algorithms in Huang et. al. [4].

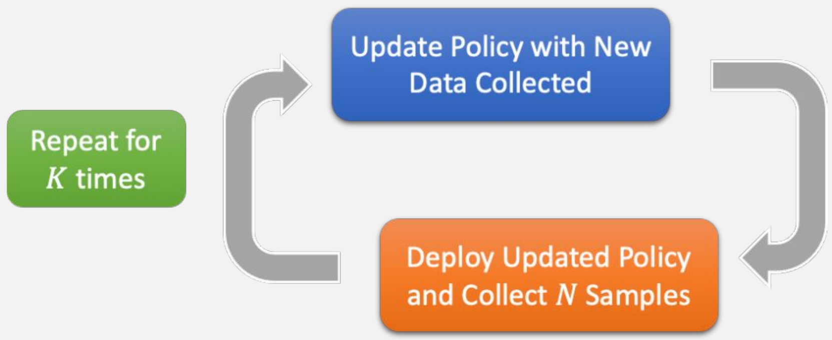


Figure 1: An Abstraction of Learning Process of Online RL Setting

3 Algorithm and Methods

Algorithm 1 from Huang et. al. [4]:

- Deterministic policy setting.
- Layer-by-layer exploration strategy.
- Theoretically optimal for setting: **O(*dH*)** deployments.

Our implementation:

- Environment: FrozenLake, OpenAI Gymnasium library.
- First-ever implementation of this algorithm, including end-to-end training and deployment of the learned policy.
- Can learn an optimal policy in as few as 10 deployments.

Algorithm 2 from Huang et. al. [4]:

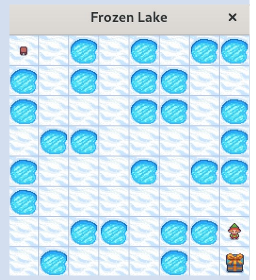
- Allows deployment of arbitrary policies, including stochastic or even non-Markovian policies.
- Improved deployment efficiency **O(*H*)**.
- More complicated implementation.

4 Initial Insights and Challenges

Insights for practical setting:

- Hyperparameters are not in line with theoretical guarantees.
- Encouraging exploration heuristic: set probability to start exploring the next layer.
- Fewer deployments than theoretically needed for guarantees.
- Since dynamics models in our environments are stationary, we can make some simplifications in the algorithm.
- Success in a simple 10x10 FrozenLake environment after reward function refinement and hyperparameter tuning.

Challenges



- Sparse rewards: Agent struggles to find good policies in a sparse reward environment (e.g. 1 for reaching goal, 0 otherwise).
- Environment exploration: Agent struggles to explore due to only deterministic policies.
- Sensitivity to hyperparameters: highly sensitive to certain hyperparameters in practice, require fine tuning.

5 Future Investigation

- Further refinement and evaluation of the deterministic agent on the FrozenLake environment.
- Investigate the possible impact of hyperparameter tuning.
- Compare the performance of our implementation to the theoretical limit.
- Benchmark the performance against standard algorithms such as Q-learning etc.
- Set up additional test environments like GridWorld.
- Implement a stochastic DE-RL Agent using the Deployment-Efficient RL with Covariance Matrix Estimation algorithm (Algorithm 2).

References

[1] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, 2016.
[2] Mehdi Fatemi, Taylor W Killian, Jayakumar Subramanian, and Marzyeh Ghassemi. Medical dead-ends and learning to identify high-risk states and treatments. Advances in Neural Information Processing Systems, 34:4856–4870, 2021.
[3] Yuanguo Lin, Yong Liu, Fan Lin, Lixin Zou, Pengcheng Wu, Wenhua Zeng, Huanhuan Chen, and Chunyan Miao. A survey on reinforcement learning for recommender systems, 2022.
[4] Jiawei Huang, Jinglin Chen, Li Zhao, Tao Qin, Nan Jiang, and Tie-Yan Liu. Towards deployment-efficient reinforcement learning: Lower bound and optimality. arXiv preprint arXiv:2202.06450, 2022.