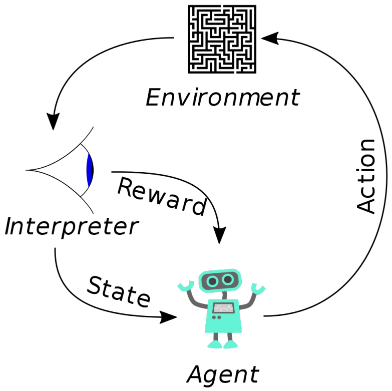
Empirical Analysis of Policy Gradient Algorithms where Starting States are Sampled accordingly to Most Frequently Visited States

Samy Aittahar, Raphaël Fonteneau, Damien Ernst
University of Liège

IFAC 2020 World Congress

Outline

Sequential Tasks: A Visual Overview



In reinforcement learning,
the agent learns how to take actions that maximise rewards based on its observations and its past experience.

Reinforcement Learning: Achievements

Recent developments have shown impressive results in complex games.

Further research effort is required to handle industrial applications, such as autonomous driving and power system related tasks, though.

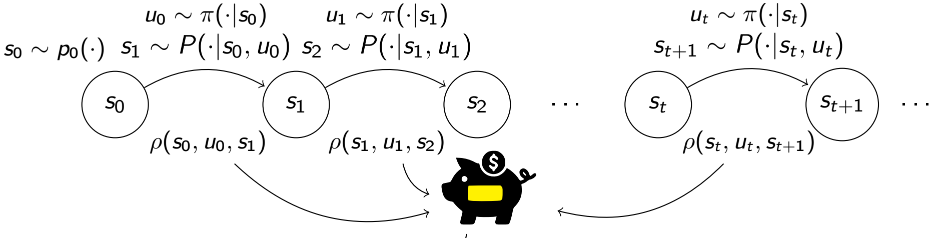
Reinforcement Learning: MDP and Policy

A Markov Decision Process (MDP) is a sequential task defined by:

- A state space $\mathcal{S}$.
    - We assume that what the agent observes the environment state.
- An action space $\mathcal{U}$.
- An initial state distribution $p_0 : \mathcal{S} \rightarrow [0, 1]$.
- A conditional probability distribution - called the *transition function* - which specifies the probability of reaching the next state from a state-action pair $P : \mathcal{S} \times \mathcal{U} \times \mathcal{S}$ denoted as $P(\cdot|\cdot, \cdot)$.
- A real-valued function $\rho : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ which specifies the reward collected from reaching a given state from a state-action pair.
- A discount factor γ which specifies the importance of future rewards.

A policy is defined as a density-based probabilistic mapping $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^+$.

Reinforcement Learning: Policy Performance


$u_0 \sim \pi(\cdot|s_0)$
$u_1 \sim \pi(\cdot|s_1)$
$u_t \sim \pi(\cdot|s_t)$
$s_0 \sim p_0(\cdot)$
$s_1 \sim P(\cdot|s_0, u_0)$
$s_2 \sim P(\cdot|s_1, u_1)$
$s_{t+1} \sim P(\cdot|s_t, u_t)$
s_0
s_1
s_2
$\cdots$
s_t
s_{t+1}
$\cdots$
$\rho(s_0, u_0, s_1)$
$\rho(s_1, u_1, s_2)$
$\rho(s_t, u_t, s_{t+1})$

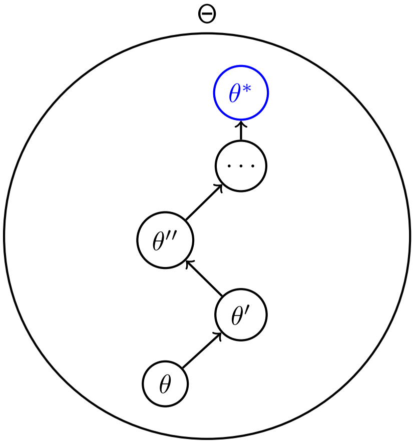
$$\sum_{t=0}^{\infty} \gamma^t \rho(s_t, u_t, s_{t+1}) = R^{\pi}_{s_0} \longrightarrow$$

$$\boxed{\begin{array}{c}\text{Policy performance:}\\ \underset{\substack{s_0 \sim p_0(\cdot), u_t \sim \pi(\cdot|s_t), \forall t \in \mathbb{N}\\ s_{t+1} \sim P(\cdot|s_t, u_t), \forall t \in \mathbb{N}}}{\mathbb{E}} \left\{ R^{\pi}_{s_0} \right\} = J(\pi)\end{array}}$$

Direct Policy Search Techniques for Finding an Optimal Parametrised Policy

The goal is to find an optimal policy $\arg\max_{\pi} J(\pi)$.

Approach - Search optimal policies that are parametrised by vectors $\theta \in \Theta$.



Policy search techniques -
Navigate in the space Θ towards θ^*.

Gradient-free policy search techniques: Monte-Carlo Tree Search techniques [3], Covariance Matrix Adaptation ([2]), Cross-Entropy Methods [4].

Gradient-based policy search techniques (our scope): Policy Gradient algorithms [6]

Policy Gradient Theorem

Under mild assumptions on the environment components,
the Policy Gradient Theorem [6] states that the gradient of $J(\pi_\theta)$ is

$$\nabla_\theta J(\pi_\theta) = \int_{\mathcal{S}} \int_{\mathcal{U}} p^{\pi_\theta}(s) Q^{\pi_\theta}(s,u) \nabla_\theta \pi_\theta(u|s) du ds,$$

Discounted State Visitation Measure
$\int_{\mathcal{S}} p_0(s') \sum_{k=0}^{\infty} \gamma^k Pr(s' \to s, k, \pi_\theta) ds'$.
Pr is a k-step state-to-state transition density-based distribution

Cost-to-go from applying action u to state s following π_θ afterwards

$$\underset{\substack{u \sim \pi(\cdot|s), \forall t \in \mathbb{N}_+ \\ s' \sim P(\cdot|s,u), \forall t \in \mathbb{N}_+ \\ R^{\pi}_{s'}}}{\mathbb{E}} \{\rho(s,u,s') + \gamma R^{\pi}_{s'}\}$$

Iterate over Policy Gradient Estimates

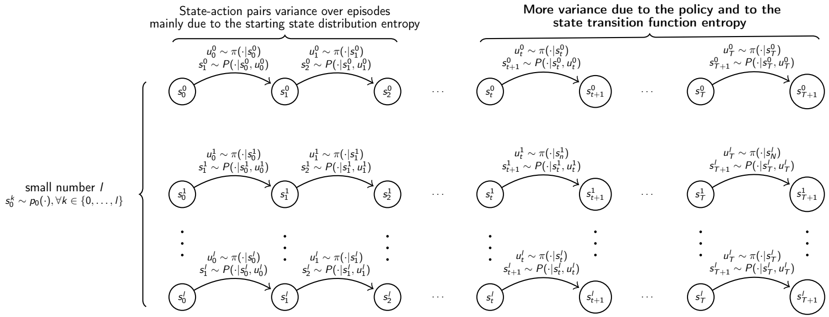
Since the environment is unknown, $\nabla_\theta J(\pi_\theta)$ cannot be computed.
Approach - Estimate the gradient of the policy π_θ from the trajectories generated by this policy using the REINFORCE [8] estimator:

$$\hat{\nabla}_\theta J(\pi_\theta) = \frac{1}{T} \sum_{t=0}^{T-1} \nabla_\theta \pi_\theta(u_t|s_t) \sum_{k=0}^{T} \gamma^k \rho_k$$

.
Use the REINFORCE estimator to compute a well-performing θ as follows:

1. Define a learning rate α and a prior policy parametrised with θ.
2. Collect trajectories by sampling actions with the policy π_θ from initial states sampled with p_0.
3. Compute $\hat{\nabla}_\theta J(\pi_\theta)$ by aggregating REINFORCE estimates across the trajectories.
4. Set $\theta \leftarrow \theta + \alpha \hat{\nabla}_\theta J(\pi_\theta)$ and go to Step 2 or stop here.

The Large Variance Issue over Finite Trajectories


State-action pairs variance over episodes mainly due to the starting state distribution entropy
More variance due to the policy and to the state transition function entropy
small number l
$s_0^k \sim p_0(\cdot), \forall k \in \{0, \dots, l\}$
$u_0^0 \sim \pi(\cdot|s_0^0)$ $s_1^0 \sim P(\cdot|s_0^0, u_0^0)$
$u_1^0 \sim \pi(\cdot|s_1^0)$ $s_2^0 \sim P(\cdot|s_1^0, u_1^0)$
$u_t^0 \sim \pi(\cdot|s_t^0)$ $s_{t+1}^0 \sim P(\cdot|s_t^0, u_t^0)$
$u_T^0 \sim \pi(\cdot|s_T^0)$ $s_{T+1}^0 \sim P(\cdot|s_T^0, u_T^0)$
$u_0^1 \sim \pi(\cdot|s_0^1)$ $s_1^1 \sim P(\cdot|s_0^1, u_0^1)$
$u_1^1 \sim \pi(\cdot|s_1^1)$ $s_2^1 \sim P(\cdot|s_1^1, u_1^1)$
$u_t^1 \sim \pi(\cdot|s_n^1)$ $s_{t+1}^1 \sim P(\cdot|s_t^1, u_t^1)$
$u_T^l \sim \pi(\cdot|s_N^l)$ $s_{T+1}^l \sim P(\cdot|s_T^l, u_T^l)$
$u_0^l \sim \pi(\cdot|s_0^l)$ $s_1^l \sim P(\cdot|s_0^l, u_0^l)$
$u_1^l \sim \pi(\cdot|s_1^l)$ $s_2^l \sim P(\cdot|s_1^l, u_1^l)$
$u_t^l \sim \pi(\cdot|s_t^l)$ $s_{t+1}^l \sim P(\cdot|s_t^l, u_t^l)$
$u_T^l \sim \pi(\cdot|s_T^l)$ $s_{T+1}^l \sim P(\cdot|s_T^l, u_T^l)$
s_0^0 s_1^0 s_2^0 s_t^0 s_{t+1}^0 s_T^0 s_{T+1}^0
s_0^1 s_1^1 s_2^1 s_t^1 s_{t+1}^1 s_T^1 s_{T+1}^1
s_0^l s_1^l s_2^l s_t^l s_{t+1}^l s_T^l s_{T+1}^l

Consequence - Mismatch between the state distribution of the finite trajectories and the discounted state visitation measure $p^{\pi_\theta} \rightarrow$ Unstable policy updates.

Assumption - The agent is able to sample starting states with a probability distribution that differs from p_0.

Our proposition - Sample starting states from p^{π_θ}.

Policy Gradient Estimates by Sampling over Frequently Visited Starting States

Unfortunately, the agent has not access to p^{π_θ}.

Proposition - Approximate p^{π_θ} by iteratively updating the parametric probability density function to fit the state distribution of the episodes generated by the current policy.

1. Define a learning rate α and a prior policy parametrised with θ.
2. Define the prior starting state distribution $p_0' \leftarrow p_0$.
3. Trajectory sampling sub-routine:
   1. Sample the initial state from p_0'.
   2. Generate a trajectory by sampling actions from the policy π_θ
   3. Compute the parametric distribution $f(\cdot)$ to fit the trajectories generated by π_θ so far.
   4. Set $p_0' \leftarrow f$ and go back to Step 3.1 or stop here.
4. Compute $\hat{\nabla}_\theta J(\pi_\theta)$ by aggregating REINFORCE estimates across the trajectories.
5. Set $\theta \leftarrow \theta + \alpha \hat{\nabla}_\theta J(\pi_\theta)$ and go to Step 3 or stop here.

Related work

- Sampling starting states from a surrogate initial state distribution has been considered in the literature
- Uniform sampling of initial state from the goal state(s) neighborhood with progressive widening [1] .
- Sampling starting states from expert trajectories to solve hard-to-explore games [5].
- To the best of our knowledge, there is no existing approach where starting states are sampled accordingly to their future discounted visitation measures.

Experimental protocol

To assess the benefits of our approach, we have designed three policy search cases:

1. The starting state is always sampled from the original initial state distribution.
2. The starting state is always sampled from an uniform distribution over the state space (bounded support).
3. The very first starting state is sampled from the original initial state distribution. Then, at the end of each episode, the parameters of a multivariate Gaussian mixture are computed to fit the state distribution of the trajectories generated by the policy, as explained in our modified policy gradient algorithm.

For each case, at each policy update, the new policy is tested by simulating trajectories for which starting states are sampled from the original initial state distribution.

Result samples (1/3)

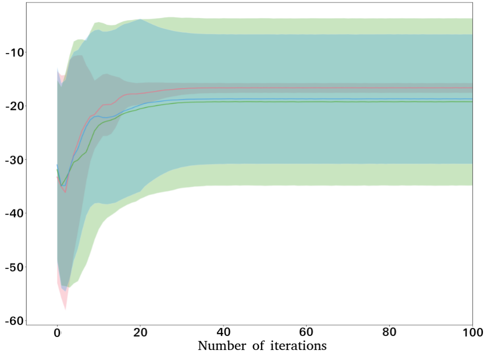


Figure: Mass Spring Damper System - Performance report in terms of mean cumulative reward at each policy update (shaded area is std). Green, blue and red curve correspond to the first, the second and the third case, respectively.

Result samples (2/3)

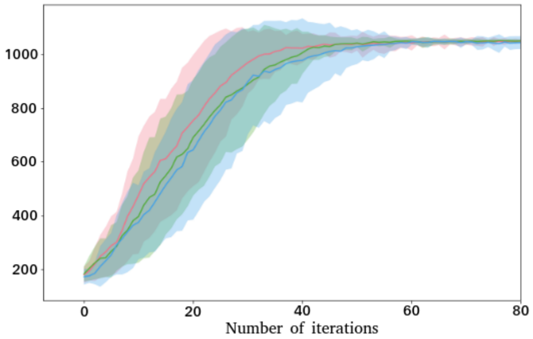


Figure: Catcher - Performance report in terms of mean cumulative reward at each policy update (shaded area is std). Green, blue and red curve correspond to the first, the second and the third case, respectively.

Result samples (3/3)

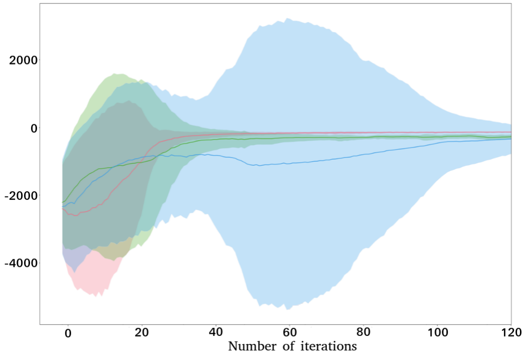


Figure: Ship Steering - Performance report in terms of mean cumulative reward at each policy update (shaded area is std). Green, blue and red curve correspond to the first, the second and the third case, respectively.

Observations

- Mean cumulative rewards are slightly higher in the third case compared to others. The policy updates are also more stable for the third case.
    - Though, during the first policy updates in Ship steering, performances are better in the first and the second case.
- Sensitivity analyses (reported in the paper) over the hyper-parameters have shown that our approach is rather sensitive to the choice of both the learning rate and the number of components of the multivariate Gaussian mixture, with no clear pattern.
    - This might explains the performance issue of the third case for the first policy updates in the Ship steering environment.

Limitations and Future Research Directions

- While improvements have been observed in our approach, a theoretical bias-variance analysis should be carried out as future work.
- Further empirical analysis suggests that the hyperparameters specific to the third case should be optimized over the policy updates course.
    - Suggestion: Multi-armed bandits where each arm is associated to an hyper-parameter value and its reward to the associated policy performance.
- Finally, our approach could be applied and studied in other gradient-free reinforcement learning algorithms.

Limitations and Future Research Directions (cont'd)

- Our approach is limited to simulator-based environments where the agent has access to the state.
- What if the agent has only access to partial information about the state?
- Suggestion #1: Introduce the assumption that the agent can restart from a state it has already encountered in the past. Fit parametric distribution $f(\cdot)$ from the partial information and use it to restart from a given state in the past trajectories.
    - Quite simple approach, but may be memory consuming in practice (e.g., when dealing with complex video game).
    - Neural network-based density functions could be exploited for complex signals [7].
- Suggestion #2: Train in parallel an additional policy to reject actions, based on the information collected throughout the trajectory, that might lead to state regions that are likely to be less visited by the policy in expectation.

References I

C. Florensa, D. Held, M. Wulfmeier, and P. Abbeel.
Reverse curriculum generation for reinforcement learning.
CoRR, abs/1707.05300, 2017.

Nikolaus Hansen and Andreas Ostermeier.
Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation.
In *Proceedings of IEEE international conference on evolutionary computation*, pages 312–317. IEEE, 1996.

Xiaobai Ma, Katherine Driggs-Campbell, Zongzhang Zhang, and Mykel J. Kochenderfer.
Monte carlo tree search for policy optimization.
In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3116–3122. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

References II

S. Mannor, R. Y. Rubinstein, and Y. Gat.
The cross entropy method for fast policy search.
In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 512–519, 2003.

T. Salimans and R. Chen.
Learning montezuma's revenge from a single demonstration.
CoRR, abs/1812.03381, 2018.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour.
Policy gradient methods for reinforcement learning with function approximation.
In *Advances in neural information processing systems*, pages 1057–1063, 2000.

References III

Antoine Wehenkel and Gilles Louppe.
Unconstrained monotonic neural networks.
In *Advances in Neural Information Processing Systems*, pages 1543–1553, 2019.

R. J. Williams.
Simple statistical gradient-following algorithms for connectionist reinforcement learning.
Machine Learning, 8(3):229–256, May 1992.