

# Goal-Conditioned Meta-Reinforcement Learning with Replay

Ahmed Akakzia<sup>12</sup>  
ahmed.akakzia@isir.upmc.fr

## **ENSTA Tutor:**

Pierre Carpentier<sup>1</sup>

## **Internship advisors:**

Olivier Sigaud<sup>2</sup>

Mohamed Chetouani<sup>2</sup>

<sup>1</sup>Ecole Nationale Supérieure de Techniques Avancées  
Institut Polytechnique de Paris

<sup>2</sup>Institut des Systèmes Intelligents et de Robotique  
Sorbonne Université



# Contents

- 1 Introduction
- 2 Background
  - Overview on RL
  - Open-ended Learning
  - Universal Value Function Approximators
- 3 Multitask versus Meta-Reinforcement Learning
- 4 Goal-Aware Replay in Generalization-Apt Meta-Learning
- 5 Setup and Experimental Results
- 6 Conclusion and future work
- 7 Bibliography

# Introduction

- Humans are continuously-learning creatures.
- They have a strong and efficient transfer mechanism for adaptation.



What about machines ?

Our problematic: When dealing with tasks with multiple goals in an open-ended context, does meta-learning bring a plus ?

# Contents

- 1 Introduction
- 2 Background
  - Overview on RL
  - Open-ended Learning
  - Universal Value Function Approximators
- 3 Multitask versus Meta-Reinforcement Learning
- 4 Goal-Aware Replay in Generalization-Apt Meta-Learning
- 5 Setup and Experimental Results
- 6 Conclusion and future work
- 7 Bibliography

# Overview on Reinforcement Learning

- Learn how to act through experience without an explicit teacher.
- Interact with the "environment" via trial and error.
- Maximize some cumulative feedback overtime.



Figure: Overview on how reinforcement learning framework works

# Overview on Reinforcement Learning

Formally, Reinforcement Learning [12] can be used to solve **Markov Decision Processes**

## Markov Decision Process

A Markov Decision Process is a Markov Process (Chain) with reward values and control:  $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$ , where  $\mathcal{S}$  is the state space,  $\mathcal{A}$  is the action space,  $\mathcal{P}$  is the probability distribution of the transition dynamics,  $\mathcal{R}$  is the reward function and  $\gamma$  is the discount factor.

How does an agent take an action  $a_t$ ? Through a policy  $\pi$ !

**Goal:** Learn  $\pi$  that maximizes the cumulative reward.

⇒ Use Reinforcement Learning Algorithms.

# Overview on Reinforcement Learning

Deep Learning + RL  $\rightarrow$  Deep RL!

- Universality thanks to Neural Networks.
- Ability to solve complex tasks independently.
- Very task specific.
- Sample inefficient.
- Costly!

# Open-ended Learning

- Context: learn how to accomplish a continuous stream of unforeseen tasks [2].
- Problem: Inability to build the corresponding MDPs at design time

→ Task-specific rewards are not adequate in the RL scheme.

Hence, the role of the designer is to make an agent capable of leveraging its own interpretations from the states and feedback it perceives from the environment.



# Universal Value Function Approximators

## Value Function $V^\pi$

A value function is denoted  $V^\pi$ . It corresponds to the expected sum of discounted rewards received starting from state  $s$  at time  $t$  following the policy  $\pi$ :

$$V^\pi(s) = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s].$$

- A key component of RL, yet, unavailable especially in open-ended learning.
- We need to construct an estimator  $V(s; \theta)$ .
- In our case, tasks consist in reaching a goal  $g \in \mathcal{G} \subset \mathcal{S}$ .

$\Rightarrow$  UVFAs [10] aim at generalizing not only over states  $s$  but also over goals  $g$ .

# Goal Conditioned Policies

- A policy  $\pi(a|s)$  represents the probability to take action  $a \in \mathcal{A}$  when the agent is in state  $s \in \mathcal{S}$ .

What if we are dealing with goals  $g \in \mathcal{G} \subset \mathcal{S}$  to be reached?

⇒ Use Goal Conditioned Policies (GCPs) [6]!

The idea is to use a policy conditioned not only on the state but also on the goal:  $\pi(a|s, g)$ .

Thanks to the universality of neural networks and to UVFAs, one can generalize not only over states but also over goals.

⇒ Transfer among states and goals assured.

# Hindsight Experience Replay

Issue:

- 1 Sparse rewards.
- 2 Growing complexity of  $\mathcal{S}$  either because of
  - ▶ continuous state space or
  - ▶ huge number of reachable states

→ When aiming at a goal  $g$ , reaching it via exploration becomes less and less unlikely.

Solution: Use Hindsight Experience Replay (HER) [1].

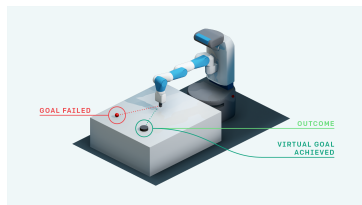


Figure: HER in a Fetch example [9].

# Contents

- 1 Introduction
- 2 Background
  - Overview on RL
  - Open-ended Learning
  - Universal Value Function Approximators
- 3 Multitask versus Meta-Reinforcement Learning**
- 4 Goal-Aware Replay in Generalization-Apt Meta-Learning
- 5 Setup and Experimental Results
- 6 Conclusion and future work
- 7 Bibliography

Back to initial problem: deal with multiple tasks in an open-ended context

## Multitask Reinforcement Learning

- Takes as input multiple training tasks.
- Trains a same policy on these tasks.
- Aims at performing efficiently on these tasks.

## Meta Reinforcement Learning

- Takes as input multiple training tasks.
- Trains a policy able to solve these tasks in few shots.
- Aims at learning a latent structure crucial for fast adaptation to unforeseen testing tasks.

→ In both cases, all the tasks need to share a certain structure.

# Meta Reinforcement Learning

## Task $\mathcal{T}$

A task  $\mathcal{T}$  to be solved with reinforcement learning is defined as a set of initial probability distribution over states, a transition distribution, a parametric loss function and a time horizon.

$$\mathcal{T} = \{p(s_0), p(s_{t+1}|s_t, a_t), \mathcal{L}(\theta, \mathcal{D}), H\},$$

where  $\mathcal{D} = \{(s_0, a_0, \dots, s_H, a_H)^k\}$  is a set of trajectories.

# Meta Reinforcement Learning

## Mechanistic view

- 1 Trains a policy using training tasks.
- 2 Takes a new unforeseen task, perform one/few roll-outs.
- 3 Adapt to the new task.

- A distribution over tasks  $p(\mathcal{T})$  needs to be defined.
- All considered tasks need to be drawn from this distribution.
- Many ways of doing meta-learning: hyper-parameters [13], loss functions [11], exploration strategies [7], initialization weights [4].

## Probabilistic view

- 1 Extract prior information using training tasks.
- 2 Takes a new unforeseen task, infer posterior parameters.
- 3 Adapt to the new task.

# Model-Agnostic Meta-learning (MAML)

MAML [4] is a gradient-based meta-learning algorithm

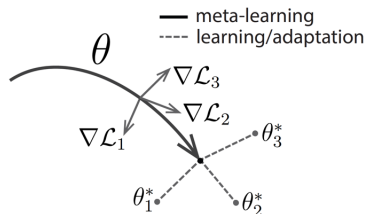


Figure: MAML optimization path [4]

Two nested updates:

- Inner update (one or many):  $\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_i)$ .
- Outer update:  $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{T_i \sim p(T)} \mathcal{L}(\theta'_i, \mathcal{D}'_i)$ .



# Contents

- 1 Introduction
- 2 Background
  - Overview on RL
  - Open-ended Learning
  - Universal Value Function Approximators
- 3 Multitask versus Meta-Reinforcement Learning
- 4 Goal-Aware Replay in Generalization-Apt Meta-Learning
- 5 Setup and Experimental Results
- 6 Conclusion and future work
- 7 Bibliography



Figure: Gargamel image from The Smurfs show.

A multigoal meta-reinforcement algorithm based on:

- HER,
- Soft-Actor-Critic RL algorithm [8],
- MAML meta-optimization mechanism.

# Soft-Actor-Critic (SAC) algorithm I

- A state-of-the-art RL algorithm.
- Mainly defined for RL tasks involving continuous actions.
- Trains two networks, the actor and the critic:
  - ▶ The actor takes as input the state and goal and outputs a proba over actions.
  - ▶ The critic takes as input the state, goal and action and outputs an action value function (Q-function).
  - ▶ The actor and the critic networks communicate together.

## Action value function: $Q(s, a)$

An action value function is denoted  $Q^\pi$ . It corresponds to the expected sum of discounted rewards received starting from a state  $s$  and taking an action  $a$  at time  $t$ , following the policy  $\pi$ .

$$Q^\pi(s, a) = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s, a_t = a].$$

# Soft-Actor-Critic (SAC) algorithm II

## Value function and Action value function

$$V^\pi(s) = \int_{a \in \mathcal{A}} Q^\pi(s, a)$$

- SAC is an off-policy algorithm.
- It stores trajectories in a replay buffer.
- The objective function it optimizes is:

$$J(\pi) = \sum_{t=0}^T \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t))],$$

where  $\mathcal{H}$  defines the entropy function.

- It uses two different critic networks to mitigate the approximation errors [5].

In our case, we use HER in the SAC algorithm: SAC+HER.

# SAC+HER for Multitask Learning

- Uses trajectories from different training tasks instead of a single one.
- Shares a replay buffer among all the training tasks.
- Learns a policy capable of performing effectively on all the training tasks.

The different training tasks need to share a same structure.

Using GCPs permits the use of a shared replay buffer as rewards become task-independent.

# Goal-Aware Replay in Generalization-Apt Meta-Learning (GARGAML) I

Given  $N$  training tasks and an off-policy RL algorithm, GARGAML works as follows:

- 1 Initialize the meta-parameter  $\theta$
- 2 Perform  $k$  inner updates for each task:
  - 1 Create a copy from the meta-parameters.
  - 2 Perform roll-outs using the pre-update parameters.
  - 3 Evaluate the loss function using created trajectories.
  - 4 Perform  $k$  inner updates
  - 5 Perform roll-outs using post-update parameters.
- 3 Evaluate the meta-loss using post-update trajectories and post-update parameters.
- 4 Perform the meta-update of  $\theta$
- GARGAML uses two multitask replay buffer: one for pre-updated parameters and one for post-updated parameters.

# Goal-Aware Replay in Generalization-Apt Meta-Learning (GARGAML) II

- the post-update replay buffer is used in the meta-optimization.
- The used RL algorithm is SAC, which optimizes the actor and the critic loss.
- GARGAML performs the meta-optimization only on the critic loss function.

## Constraint on the loss function

To ensure the universality of the few-shot meta-learning [3], the loss function needs to have a gradient that is linear to the output of the last layer of the neural network when evaluated in 0. Besides, the coefficient matrix needs to be invertible:

$$\nabla_y \mathcal{L}(y, 0) = Ay,$$

where  $A$  is invertible.

# Contents

- 1 Introduction
- 2 Background
  - Overview on RL
  - Open-ended Learning
  - Universal Value Function Approximators
- 3 Multitask versus Meta-Reinforcement Learning
- 4 Goal-Aware Replay in Generalization-Apt Meta-Learning
- 5 Setup and Experimental Results**
- 6 Conclusion and future work
- 7 Bibliography





# Setup I

- 3D Navigation in a cube.
- Continuous state and action spaces.
- 0/1 Sparse reward signals.
- Goal Conditioned Policies.
- The agent always starts from the center of the cube.
- **Evaluation:** Success rate over  $N$  episodes.

Our goal is to see whether the proposed meta-RL GARGAML algorithm generalizes better than classic multitask RL SAC.

For this, we use the mujoco fetch environments [9].



## Setup II

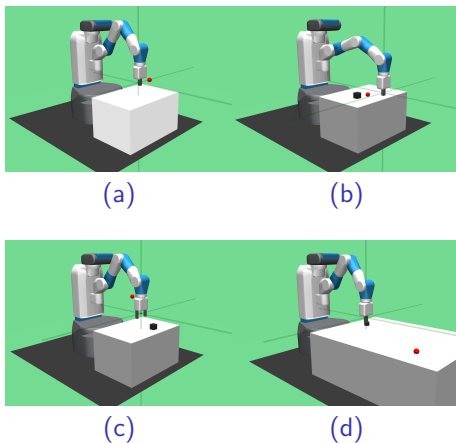


Figure: Mujoco Fetch (a) reach (b) push (c) pick and place and (d) slide

# Evaluation protocol I

We perform two batch of experiments with different protocols:

## One task, multiple goals

- At each run, we use one single task: Reach/Push...
- We decompose the 3D space into a training and a testing zone.
- During training, we generate goals in the training zone.
- At test time, the agent must reach goals in the unexplored testing zone.

## Unforeseen tasks, multiple goals

- Decompose the tasks in training and testing tasks.
- During training, all the training tasks are given to the agent.
- At test time, the agent must adapt to unforeseen testing tasks.

# Evaluation protocol II

## One task, multiple goals

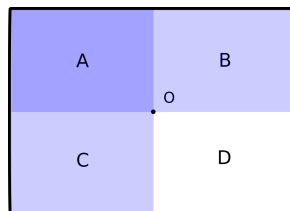


Figure: Multigoal evaluation protocol in the first experiment.

## Unforeseen tasks, multiple goals

- Training tasks: FetchReach, FetchPush, FetchPickAndPlace.
- Testing task: FetchSlide

- Training zone: ABC.
- Testing zone: D.

# Experimental results: One task, multiple goals

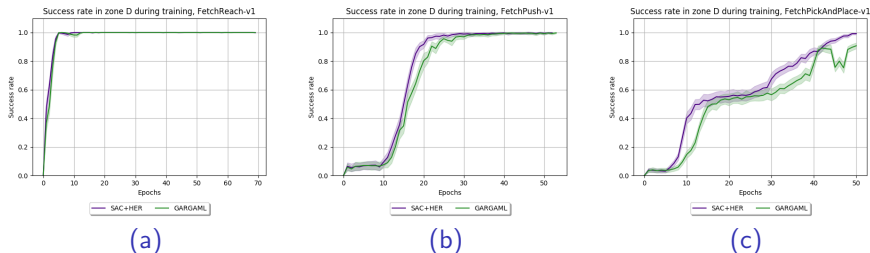


Figure: Success rate progress in testing zone D during training (500 epoch) in Fetch (a) reach (b) push (c) pickandplace.

- Both agents manage to perform optimally on goals generated in the testing zone and fine tuning is not necessary.
- Meta-RL mechanism in GARGAML doesn't bring too much. Actually, we believe this is because multigoal + GCPs  $\neq$  multitask.

⇒ We are doing meta-learning and adaptation on same task.

# Experimental results: Multiple tasks, multiple goals

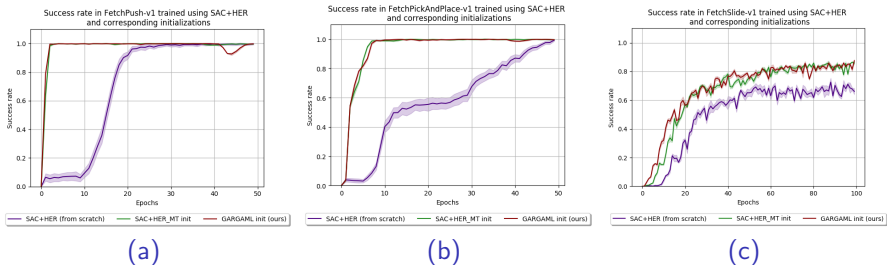


Figure: Success rate progress using SAC+HER in fine tuning in fetch (a) push (b) pickandplace (c) slide

- From the learned initializations, both algorithms manage to fastly solve training tasks ((a) and (b)).
- GARGAML is slightly better in adapting to the testing task:
  - ▶ The training tasks are not different enough, or
  - ▶ The testing task is very close to the training tasks.

# Contents

- 1 Introduction
- 2 Background
  - Overview on RL
  - Open-ended Learning
  - Universal Value Function Approximators
- 3 Multitask versus Meta-Reinforcement Learning
- 4 Goal-Aware Replay in Generalization-Apt Meta-Learning
- 5 Setup and Experimental Results
- 6 Conclusion and future work
- 7 Bibliography

# Conclusion

- GARGAML is slightly better than classic multitask approaches when adapting with unforeseen tasks.
- The slight improvement may be due to the fact that the tasks we considered are not rich enough.
- Our next steps will be:
  - 1 Encode contextual latent variables relative to the structure of tasks.
  - 2 Use more and varied tasks.



Thanks!

Thank you for your attention!  
Any question?








IP PARIS


# Contents

- 1 Introduction
- 2 Background
  - Overview on RL
  - Open-ended Learning
  - Universal Value Function Approximators
- 3 Multitask versus Meta-Reinforcement Learning
- 4 Goal-Aware Replay in Generalization-Apt Meta-Learning
- 5 Setup and Experimental Results
- 6 Conclusion and future work
- 7 Bibliography**

# Bibliography I

-  Marcin Andrychowicz et al. “Hindsight Experience Replay”. In: *arXiv preprint arXiv:1707.01495* (2017).
-  Stephane Doncieux et al. “Open-ended Learning: a Conceptual Framework based on Representational Redescription”. In: *Frontiers in Robotics and AI* 12 (2018). DOI: [10.3389/fnbot.2018.00059](https://doi.org/10.3389/fnbot.2018.00059).
-  Chelsea Finn. “Learning to Learn with Gradients”. PhD thesis. UC Berkeley, 2018.
-  Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1126–1135.
-  Scott Fujimoto, Herke van Hoof, and David Meger. “Addressing function approximation error in actor-critic methods”. In: *arXiv preprint arXiv:1802.09477* (2018).

## Bibliography II

-  Dibya Ghosh, Abhishek Gupta, and Sergey Levine. “Learning Actionable Representations with Goal-Conditioned Policies”. In: *arXiv preprint arXiv:1811.07819* (2018).
-  Abhishek Gupta et al. “Meta-reinforcement learning of structured exploration strategies”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 5302–5311.
-  Tuomas Haarnoja et al. “Soft actor-critic algorithms and applications”. In: *arXiv preprint arXiv:1812.05905* (2018).
-  Matthias Plappert et al. “Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research”. In: *arXiv preprint arXiv:1802.09464* (2018).
-  Tom Schaul et al. “Universal Value Function Approximators”. In: *International Conference on Machine Learning*. 2015, pp. 1314–1320.

# Bibliography III



John Schulman et al. “High-dimensional continuous control using generalized advantage estimation”. In: *arXiv preprint arXiv:1506.02438* (2015).



Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. Vol. 2. 4. MIT press Cambridge, 1998.



Zhongwen Xu, Hado P van Hasselt, and David Silver. “Meta-gradient reinforcement learning”. In: *Advances in neural information processing systems*. 2018, pp. 2396–2407.

