MS Thesis Defense: Non-Stationary MDPs and Continual Reinforcement Learning Algorithms

Sandesh Katakam - BS MS 2020

Thesis Supervisor: Dr. Srijith P.K.
Associate Professor, Department of AI
Indian Institute of Technology, Hyderabad
Co-Supervisor: Dr. Seshadri Chintapalli, IISER Berhampur

IISER Berhampur, 28/04/2025

Outline

Outline

A Brief History of RL

- Reinforcement Learning has over 70 years of rich academic history.
- Its origins trace back to the 1950s, rooted in early studies of **Markov Decision Processes (MDPs)**.
- MDPs formalize sequential decision-making under uncertainty.
- MDPs are discrete, stochastic analogs of optimal control problems, closely related to **Hamilton–Jacobi–Bellman (HJB)** equations.

Success of Reinforcement Learning: AlphaGo 2016

AlphaGo(2016 Seoul South Korea)

Figure: Lee Sedol against AlphaGo

MuZero 2019(Schrittwieser et al., Nov 2019)

RESEARCH

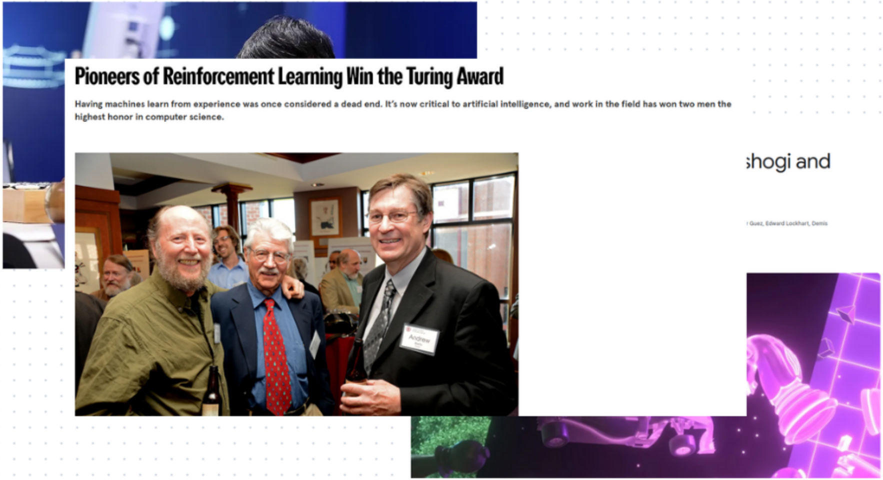
MuZero: Mastering Go, chess, shogi and Atari without rules

23 DECEMBER 2020

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, David Silver

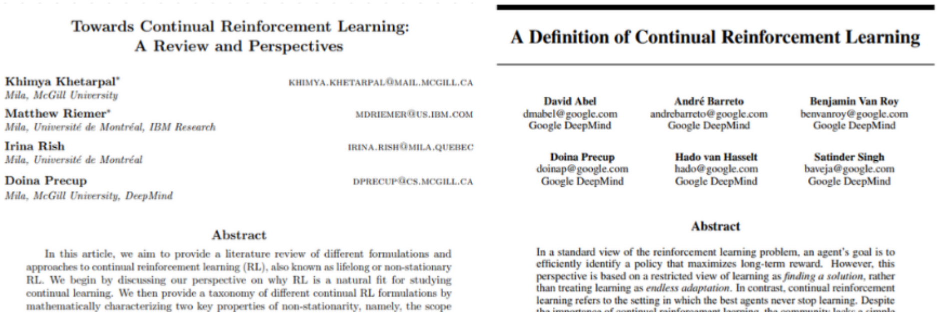
Share

Success of Reinforcement Learning: 2024 ACM Turing Award



But the Problem is...

Still a lot of problems in RL are not solved yet!! Along the direction of tasks scalability we have one such problem: Non-stationarity and Continual learning of tasks



Towards Continual Reinforcement Learning: A Review and Perspectives

Khimya Khetarpal* KHIMYA.KHETARPAL@MAIL.MCGILL.CA
Mila, McGill University
Matthew Riemer* MDRIEMER@US.IBM.COM
Mila, Université de Montréal, IBM Research
Irina Rish IRINA.RISH@MILA.QUEBEC
Mila, Université de Montréal
Doina Precup DPRECUP@CS.MCGILL.CA
Mila, McGill University, DeepMind

Abstract

In this article, we aim to provide a literature review of different formulations and approaches to continual reinforcement learning (RL), also known as lifelong or non-stationary RL. We begin by discussing our perspective on why RL is a natural fit for studying continual learning. We then provide a taxonomy of different continual RL formulations by mathematically characterizing two key properties of non-stationarity, namely, the scope

A Definition of Continual Reinforcement Learning

David Abel dmabel@google.com Google DeepMind
André Barreto andrebarreto@google.com Google DeepMind
Benjamin Van Roy benvanroy@google.com Google DeepMind
Doina Precup doinap@google.com Google DeepMind
Hado van Hasselt hado@google.com Google DeepMind
Satinder Singh baveja@google.com Google DeepMind

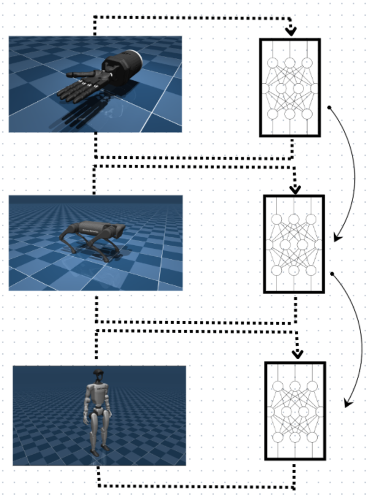
Abstract

In a standard view of the reinforcement learning problem, an agent's goal is to efficiently identify a policy that maximizes long-term reward. However, this perspective is based on a restricted view of learning as *finding a solution*, rather than treating learning as *endless adaptation*. In contrast, continual reinforcement learning refers to the setting in which the best agents never stop learning. Despite

Figure: On Left: Khetarpal et al. (2022), On Right: Abel et al. (2023)

Motivation: Why Continual RL?

- In real-world scenarios, agents face a sequence of tasks — not a fixed one.
- This leads to **non-stationarity** in dynamics, rewards, and data distribution.
- Examples:
  - A robot learning new skills across environments.
  - A recommendation system adapting to evolving user preferences.
  - An autonomous agent navigating changing traffic or weather.



Outline

Continual RL Problem Setting

(General CRL Problem $\mathcal{M}_{CRL}$):[a]

Given a state $\mathcal{S}$, action-space $\mathcal{A}$, an observation space $\mathcal{O}$, a reward function $r : \mathcal{S}x\mathcal{A} \rightarrow \mathcal{R}$, a transition function $p : \mathcal{S}x\mathcal{A} \rightarrow S$, and an observation function $x : \mathcal{S} \rightarrow \mathcal{O}$, the most general continual reinforcement learning problem problem can be expressed as

$$\mathcal{M}_{CRL} = \langle \mathcal{S}(t), \mathcal{A}(t), r(t), p(t), x(t), \mathcal{O}(t) \rangle$$

where each component of the problem formulation can be considered as a non-stationary function of form $f(i, t)$ where i is the input specific to each component.

[a] Khetarpal et.al 2022, Towards Continual Reinforcement Learning: A Review and Perspectives

Assumptions for Non-stationary Functional Form $f(i, t)$: *Lipschitz Continuity* and *Piecewise Non-stationarity*

The Non-Stationarity Problem

Definition(Non-stationary MDPs)

Non-stationary MDP as a special type of CRL Problem:[a]
where scope of non-stationarity i.e. $\alpha \subseteq \{\mathcal{S}, \mathcal{A}, r, p\}$, the observation function is an appropriate identity matrix $x = \mathcal{I}$ and the observation space is the state space $\mathcal{O} = \mathcal{S}$

$$\boxed{\mathcal{M}_{CRL} = \langle \mathcal{S}(t), \mathcal{A}(t), r(t), p(t) \rangle}$$

[a] Khetarpal et.al 2022, Towards Continual Reinforcement Learning: A Review and Perspectives

Scoping in..

Based on the scope of Non-stationary (α) which defines what elements have non-stationarity[1]

$$\alpha \subseteq \{\mathcal{S}, \mathcal{A}, r, p, x, \mathcal{O}\}$$

For our problem setting, we assume the scope includes transition function p and reward function r. So,

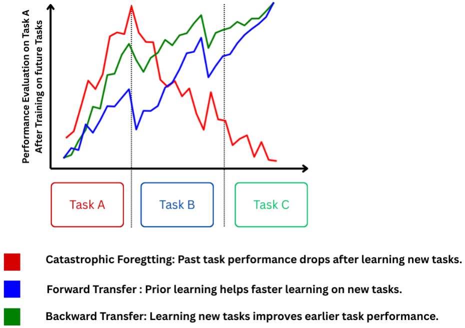
$$\boxed{\mathcal{M}_{CRL} = \langle \mathcal{S}, \mathcal{A}, r(t), p(t) \rangle}$$

[1] Khetarpal et.al 2022, Towards Continual Reinforcement Learning: A Review and Perspectives

Core Challenges in Continual RL

- **Forward Transfer:** how pre-training on an earlier task $\mathcal{T}_i$ speeds up convergence on a later task $\mathcal{T}_j$
- **Backward Transfer:** how learning task $\mathcal{T}_j$ improves performance on a previous task $\mathcal{T}_i$
- **Catastrophic Forgetting:** drop in performance on earlier tasks $\mathcal{T}_i$ after training sequentially up to $\mathcal{T}_t$

2



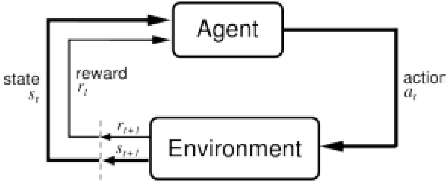
[2]Wang et.al 2023, A Comprehensive Survey of Continual Learning: Theory, Method and Application

Outline

Markov Decision Process

An MDP is defined by:

- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



Goal:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) \mid \pi\right]$$

Optimal Control: Given an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{H})$, Find an optimal policy $\pi*$ [3]

[3] Deep RL Bootcamp 2017 by Pieter Abbeel , UC Berkeley

Bellman Equations and Related Terms

Value function for a state s

$$\mathcal{V}^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1}) \;\middle|\; \pi, s_0 = s\right]$$

$=$ sum of discounted rewards when starting from state s and acting optimally

But knowing the value of a state is not enough if we also need to know which action to take

Instead of just states, what if we assign values to (state,action) pairs?

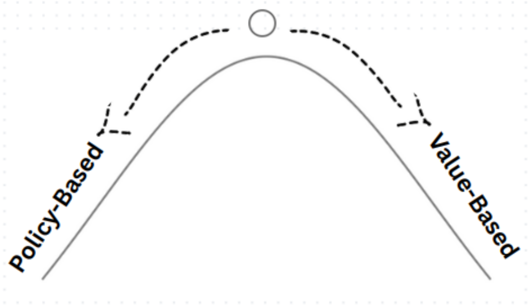
$$\mathcal{Q}^*(s, a) = \max_{\pi} \mathbb{E}\left[\textstyle\sum_{t=0}^{H} \gamma^t \mathcal{R}(s_t, a_t, s_{t-1}) \mid s_0 = s, a_0 = a, \pi\right]$$

tell us how good it is to take action a at state s and then act optimally

Broad Taxonomy of RL Algorithms

Depending on the quantity we choose to optimize, reinforcement-learning algorithms fall into two main classes:

- **Value-based methods**, which learn an action-value function $Q(s, a)$.
- **Policy-based methods**, which directly optimize a parameterized policy π_θ to maximize the expected return.



Introducing Q-Learning

Bellman Equations for $\mathcal{Q}^*$: *Optimal action-values must satisfy a recursive relationship...*

$$\mathcal{Q}^* = \mathbb{E}[\mathcal{R}(s, a, s') + \gamma \max_{a'} \mathcal{Q}^*(s', a')]$$

We don't know $\mathcal{Q}$ exactly, but we can learn iteratively by updating estimates based on this recursive formula.
The Q-Learning update rule

$$\mathcal{Q}(s, a) \leftarrow \mathcal{Q}(s, a) + \alpha(r + \gamma \max_{a'} \mathcal{Q}(s', a') - \mathcal{Q}(s, a))$$

This gives us the direct way to estimate Q-values without knowing the model of the environment
Limitations: We cannot store all Q-values in a table for every state-action pair in large state-action spaces

Deep Q-Learning(Mnih et al., 2013)

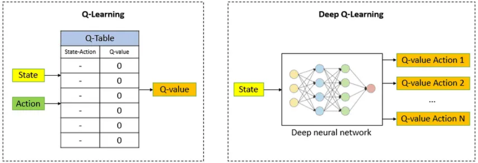
Motivation: We need a way to **generalize** Q-values across similar states. Neural networks are a good choice of function approximations for Q-values.

In Deep Q-Learning, given a state s, a neural network outputs Q-values for all actions.

$\mathcal{Q}$ is parameterized by a neural network with weights θ

Training using this objective:

$$\mathcal{L}(\theta) = (r + \gamma \max_{a'} \mathcal{Q}(s', a'; \theta^-) - \mathcal{Q}(s, a; \theta))^2$$

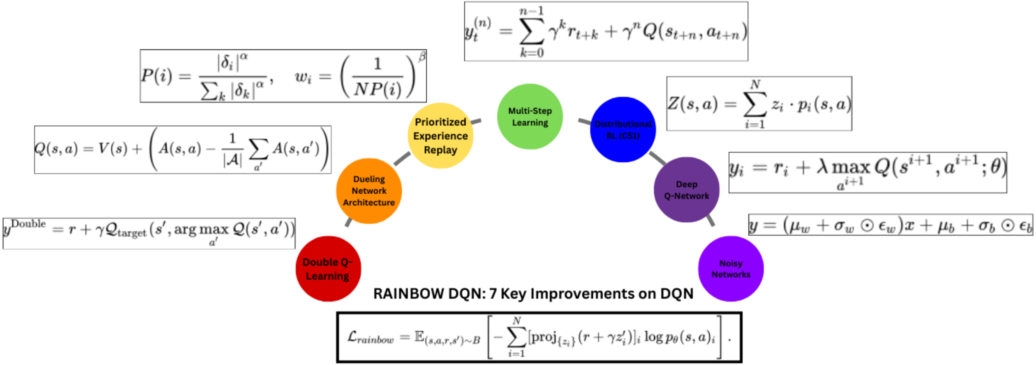


Important Tricks:

- Experience Replay
- Target networks

After each training step we use $\mathcal{Q}*$ we implicitly derive the corresponding π^* and use it to sample new actions

Improving DQN: Rainbow DQN



Hessel et.al 2017, Rainbow: Combining improvements in Deep Reinforcement Learning

Policy Optimization via Likelihood Ratio Gradient

Policy Optimization Approach:

Rather than computing Q^* first, we directly optimize:

$$\max_{\theta} \mathbb{E}\left[\sum_{t=0}^{H} \mathcal{R}(s_t) \mid \pi_\theta\right] \tag{1}$$

where θ parameterizes policy π_θ

Likelihood Ratio Method:

For trajectory $\tau = (s_0, a_0, \ldots)$ with return $R(\tau)$:

$$U(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] \tag{2}$$

$$= \sum_{\tau} P(\tau; \theta) R(\tau) \tag{3}$$

Goal: $\max_\theta U(\theta)$

This gradient-based approach directly optimizes policy parameters instead of deriving policy from value functions. [4]

[4] Deep RL Bootcamp 2017 by Pieter Abbeel , UC Berkeley

Likelihood Ratio Policy Gradient (Sutton et al., 1999)

$$U(\theta) = \sum_{\tau} P(\tau;\theta)R(\tau) \tag{4}$$

Taking gradient w.r.t θ:

$$\nabla_\theta U(\theta) = \sum_{\tau} \nabla_\theta P(\tau;\theta)R(\tau) \tag{5}$$

Using the identity:

$$\nabla_\theta P(\tau;\theta) = P(\tau;\theta)\nabla_\theta \log P(\tau;\theta) \tag{6}$$

we get: $\nabla_\theta U(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log P(\tau;\theta)R(\tau)]$

Limitations

- High variance in gradient estimates
- Sample inefficient
- Sensitive to step size

Motivation for Natural Policy Gradient

Problem: The standard policy gradient uses the Euclidean gradient, which is *not invariant* to the parameterization of the policy.

Idea: Use the **Natural Gradient**, which accounts for the geometry of the policy space.

$$\nabla_\theta^{\text{Natural}} U(\theta) = F(\theta)^{-1} \nabla_\theta U(\theta)$$

where $F(\theta)$ is the Fisher Information Matrix.

Interpretation: Move in the steepest ascent direction *measured under KL-divergence* rather than Euclidean distance.

Limitations: Still sensitive to step-sizes, No Guarantee of Monotonic improvement towards optimal policy, No Explicit trust region constraints [5]

[5] Kakade et.al 2001, A Natural Policy Gradient

Trust Region Policy Optimization (TRPO) and PPO

TRPO: Solve a constrained optimization:

$$\max_{\theta} \quad \mathbb{E}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}}(s,a)\right]$$

subject to:

$$\mathbb{E}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s) \| \pi_\theta(\cdot|s))\right] \leq \delta$$

PPO: Simplifies TRPO by using a **clipped surrogate objective**:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min\left(r(\theta)A,\ \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A\right)\right]$$

where $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}$. [6] [7]

[6] Schulman et.al 2015, Trust Region Policy Optimization

[7] Schulman et.al 2017, Proximal Policy Optimization Algorithms

Maximum Entropy RL Framework: SAC (Haarnoja et al., 2018)

Motivation: Previous methods (TRPO, PPO) focus on **constrained maximization of expected return**. SAC(Haarnoja et al., 2018) instead maximizes a **soft, entropy-augmented objective** for better exploration and robustness.

SAC Objective:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{H} \gamma^t \left(R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right)\right]$$

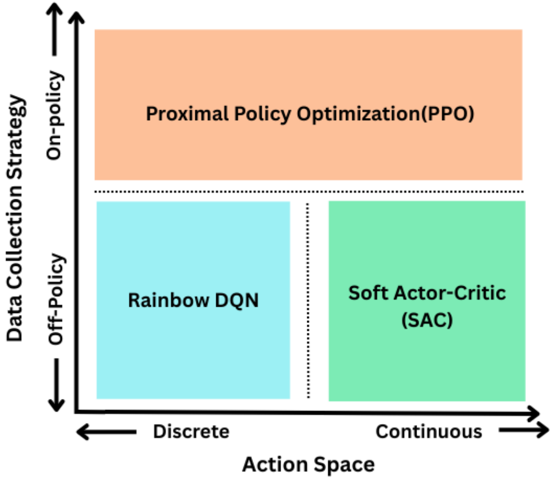
where $\mathcal{H}(\pi(\cdot|s)) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[-\log \pi(a|s)\right]$ is the policy entropy.

Key Differences:

- **Entropy regularization:** encourages *stochastic* policies for exploration.
- **Off-policy learning:** reuses experience efficiently.
- **Energy-based policies:** policies are learned implicitly via Q-functions.

Result: SAC achieves better **sample efficiency** and **stability** in practice.

The Three Representative Algorithms



Outline

Existing Methods in Continual RL

Continual learning in RL is an relatively less explored than continual learning in other settings (supervised and unsupervised setting).

Examples of some approaches: CLEAR (He and Sick, 2021), Modular Lifelong learning with neural composition(Mendez et al., 2022), Lifelong Reinforcement Learning with Modulating Masks (Ben-Iwhiwhu et al., 2023)

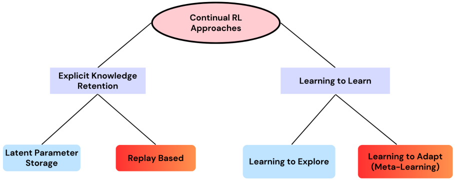


Figure: Taxonomy of Continual RL approaches

Drawbacks

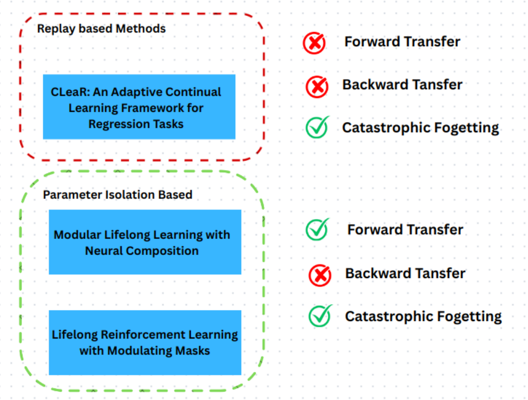


Figure: Strengths and Weaknesses of Existing Methods in Continual RL

Gradient Interference and Alignment in Continual RL

Gradient Interference: When two task gradients point in conflicting directions, updating on one degrades performance on the other:

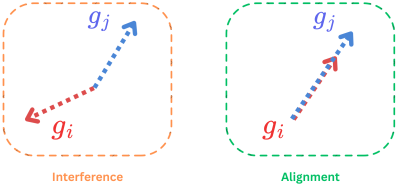
$$\nabla_\theta \mathcal{L}_i \cdot \nabla_\theta \mathcal{L}_j < 0$$

Impact:

- Interference $\Rightarrow$ catastrophic forgetting
- Alignment $\Rightarrow$ continual improvement across tasks

Gradient Alignment: When gradients for different tasks point similarly, updates yield positive transfer:

$$\nabla_\theta \mathcal{L}_i \cdot \nabla_\theta \mathcal{L}_j > 0$$



Here, $g_i = \nabla_\theta \mathcal{L}_i$ and $g_j = \nabla_\theta \mathcal{L}_j$

Outline

MAML: Model-Agnostic Meta-Learning

Bi-level Optimization: Inner and Outer Loop Updates

MAML (Finn et al., 2017)

Meta-objective:

$$\min_{\theta} \sum_{\tau \in \mathcal{T}} \mathcal{L}_{\tau}(U^k(\theta)), \quad U(\theta) = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\tau}(\theta)$$

Learn a common initialization θ such that k inner-loop gradient steps on task τ minimize its loss.

Inner loop: Task-specific adaptation via SGD.

Outer loop: Meta-optimization over many tasks.
[8]

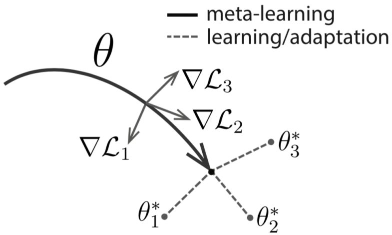


Figure: MAML update scheme showing fast adaptation and meta-update.

[8] Chelsea Finn et.al 2017 Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks(ICML 2017)

MAML vs Look-Ahead MAML

Look-Ahead MAML (Gupta et al., 2020)

We seek parameters θ and per-parameter step-sizes α to minimize over tasks $\{\mathcal{T}\}_{i=1}^{t}$

$$\min_{\theta,\alpha} \mathbb{E}_{\tau_t}\left[\mathcal{L}_{\text{meta}}\left(U^k(\theta,\alpha;\tau_t)\right)\right]$$

$$U(\theta,\alpha;\tau) = \theta - \alpha \odot \nabla_\theta \ell_{\text{inner}}(\theta;\tau).$$

First-order hypergradient:

$$g_\alpha = \frac{\partial L_{\text{meta}}(\theta_k)}{\partial \alpha} = \nabla_{\theta_k} L_{\text{meta}}(\theta_k) \cdot \left(-\sum_{j=0}^{k-1} \nabla_{\theta_j} \ell_{\text{inner}}(\theta_j)\right)$$

$$\alpha \leftarrow \max(0,\ \alpha - \eta\, g_\alpha), \quad \theta \leftarrow \theta - \alpha \odot \nabla_\theta L_{\text{meta}}(\theta_k).$$

to mitigate gradient interference

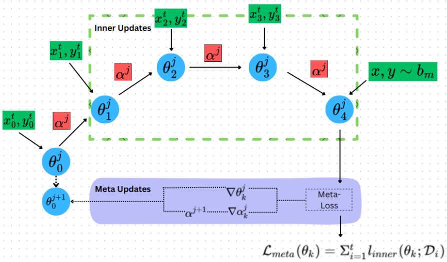


Figure: LookAhead MAML Approach

Deriving the Hypergradient g_α (Part 1)

We derive the meta-gradient w.r.t. per-parameter step-size α:

$$
\begin{aligned}
g_\alpha = \frac{\partial}{\partial \alpha} L_{\mathrm{meta}}(\theta_k) &= \frac{\partial L_{\mathrm{meta}}(\theta_k)}{\partial \theta_k} \cdot \frac{\partial \theta_k}{\partial \alpha} \\
&= \nabla_\theta L_{\mathrm{meta}}(\theta_k) \cdot \frac{\partial}{\partial \alpha} \left(\theta_{k-1} - \alpha \circ \nabla_\theta \ell_{\mathrm{inner}}(\theta_{k-1})\right) \\
&= \nabla_\theta L_{\mathrm{meta}}(\theta_k) \cdot \left(\frac{\partial \theta_{k-1}}{\partial \alpha} - \frac{\partial}{\partial \alpha} \left(\alpha \circ \nabla_\theta \ell_{\mathrm{inner}}(\theta_{k-1})\right) \right)
\end{aligned}
$$

We now recursively expand $\frac{\partial \theta_{k-1}}{\partial \alpha}$ using the update rule:

$$
\theta_j = \theta_{j-1} - \alpha \circ \nabla_\theta \ell_{\mathrm{inner}}(\theta_{j-1})
$$

Deriving the Hypergradient g_α (Part 2)

Unrolling the recursion:

$$\frac{\partial \theta_k}{\partial \alpha} = -\nabla_\theta \ell_{\text{inner}}(\theta_{k-1}) + \left(\frac{\partial \theta_{k-1}}{\partial \alpha} \cdot \frac{\partial}{\partial \theta_{k-1}} \left(-\alpha \circ \nabla_\theta \ell_{\text{inner}}(\theta_{k-1})\right) \right)$$

$$\approx -\sum_{j=0}^{k-1} \nabla_\theta \ell_{\text{inner}}(\theta_j) \quad \text{(First-order approximation: ignore higher-order } \alpha\text{-dependence)}$$

So the hypergradient becomes:

$$g_\alpha = \nabla_\theta L_{\text{meta}}(\theta_k) \cdot \left(-\sum_{j=0}^{k-1} \nabla_\theta \ell_{\text{inner}}(\theta_j) \right)$$

Update Rule:

$$\alpha \leftarrow \max(0, \alpha - \eta g_\alpha) \quad \text{(projected gradient descent)}$$

LookAhead MAML for Rainbow DQN, PPO and SAC

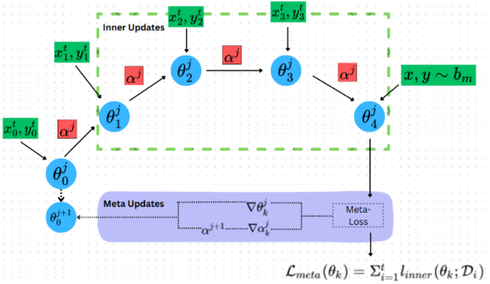
For Rainbow Algorithm

$$\ell_{\text{inner}}^{\text{Rainbow}}(\theta; B) = \mathbb{E}_{(s,a,r,s')\sim B}\left[-\sum_{i=1}^{N}\left[\text{proj}_{\{z_i\}}(r+\gamma z_i')\right]_i \log p_\theta(s,a)_i\right].$$

For PPO Algorithm

$$\ell_{\text{inner}}^{\text{PPO}}(\theta; D) = -\mathbb{E}_{(s,a)\sim D}\left[\min\left(r_\theta \hat{A}, \text{clip}(r_\theta, 1-\epsilon, 1+\epsilon)\hat{A}\right)\right]$$

$$\boxed{r_\theta = \frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}}$$



For SAC Algorithm

$$\boxed{J_\pi(\theta) = \mathbb{E}_{s\sim D, a\sim\pi_\theta}\left[\tau \log \pi_\theta(a|s) - Q_\phi(s,a)\right]}$$

$$\ell_{\text{inner}}^{\text{SAC}}(\Psi; D) = J_Q(\phi; D) + J_\pi(\theta; \phi, D) + J_\tau(\tau; \theta, D)$$

$$\boxed{J_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim D}\left[(Q_\phi(s,a) - y)^2\right]}$$

$$\boxed{J_\tau(\tau) = \mathbb{E}_{s\sim D, a\sim\pi_\theta}\left[-\tau(\log \pi_\theta(a|s) + \bar{H})\right]}$$

Proposed Modifications for Look-Ahead MAML in RL Setup

- **Experience Replay inside Inner Loop**
  We now do K gradient steps using fresh mini-batches from the replay buffer:

$$\theta_j = \theta_{j-1} - \alpha \circ \nabla_\theta[\ell_{\text{inner}}(\theta_{j-1}; \mathcal{B}_j)], \quad \mathcal{B}_j \sim \mathcal{D},\ j = 1, \ldots, K.$$

- **Correcting for Off-Policy Bias**
  When we compute the inner-loop loss on replayed transitions $(s, a, r, s') \sim \mathcal{D}$, we weight by the importance ratio:

$$\ell_{\text{inner}}(\theta) = -\mathbb{E}_{\substack{s\sim\mathcal{D} \\ a\sim\pi_{\theta_{\text{old}}}}}\Big[w(s,a)\log\pi_\theta(a \mid s)\, Q_\phi(s,a)\Big], \quad w(s,a) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)}.$$

- **Variance Control in Policy Updates**
  To stabilize the meta-gradient, we add a clipping or trust-region term to each inner step:

$$\theta_j = \theta_{j-1} - \alpha \circ \mathrm{clip}(\nabla_\theta \ell_{\text{inner}}(\theta_{j-1}), -\delta, +\delta), \text{or equivalently constrain the KL:}$$

$$\min_{\theta_j}\ \ell_{\text{inner}}(\theta_j) \quad \text{s.t.} \quad \mathrm{KL}[\pi_{\theta_j} \,\|\, \pi_{\theta_{j-1}}] \le \epsilon.$$

Proposed Modifications for Look-Ahead MAML in RL Setup

Control Prior Definition

A fixed, well-tuned policy $\pi_{\text{prior}}(a \mid s)$ (e.g. LQR, H-∞, PID) used to stabilize learning. For $\lambda \in [0, 1]$

- **Mixture Policy**

$$\pi_{\text{mix}}(a \mid s) = (1 - \lambda)\, \pi_\theta(a \mid s) + \lambda\, \pi_{\text{prior}}(a \mid s),$$

- **Gradient Estimate**

$$\nabla_\theta J_{\text{mix}} = \mathbb{E}_{s,a \sim \pi_{\text{mix}}}[\nabla_\theta \log \pi_{\text{mix}}(a \mid s)\, Q(s, a)].$$

- **Variance Reduction**

$$[\nabla_\theta J_{\text{mix}}] \ \leq\ (1 - \lambda)^2\, [\nabla_\theta J],$$

with bias $O(\lambda)$.

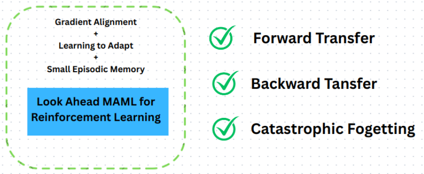
1. Choose a stabilizing prior π_{prior}.
2. Set mixing coef. λ (e.g. 0.1).
3. Collect rollouts under π_{mix}.
4. Compute updates via $\nabla_\theta \log \pi_{\text{mix}}(a \mid s)$.
5. Optionally anneal λ over time.

Outline

Conclusions

Look-Ahead MAML tackles gradient misalignment, using per-parameter learning rates and meta-objective with implementation trick that includes replay memory through Reservoir Sampling and populating a Replay Buffer ($\mathcal{R}$)

Overall, we provide a mathematical derivation of the objective functions for PPO, SAC, and the Rainbow algorithm in LookAhead MAML framework. We also provide modifications for the existing LookAhead MAML framework to RL Setup.



Future Work and Empirical Experiments

Atari benchmark:(diverse set of games) widely used tested in RL.
57 games with different transition function and reward functions

We plan to test and empirically demonstrate our proposed method **(La-MAML with PPO, SAC, and Rainbow DQN)** on a sequence of games(tasks) from this benchmark.
[9]

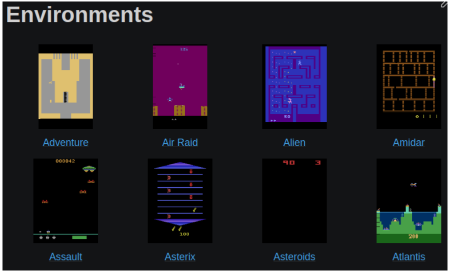


Figure: Arcade Learning Environment (ALE)

[9] Marc G. Bellemare et.al 2012, The Arcade Learning Environment: An Evaluation Platform for General Agents

Acknowledgements

Sincere thanks to my advisor Dr. Srijith P.K, Committee Members, Department of Mathematical Sciences IISER Berhampur, IIT Hyderabad

Dr. Srijith P.K
(Associate Professor),
Department of Computer Science and Department of AI,
IIT Hyderabad

BRAIN

Bayesian Reasoning And INtelligence Lab

IISER
BERHAMPUR

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Questions?

"The only stupid question is the one you were afraid to ask but never did"
- Richard Sutton

References

- Reinforcement Leraning: An Introduction, Richard Sutton and Andrew Barto
- Deep RL Bootcamp 2017 by Pieter Abbeel , UC Berkeley
- Khetarpal et.al 2022, Towards Continual Reinforcement Learning: A Review and Perspectives
- Gupta et.al 2020, La-MAML: Look-Ahead Meta Learning for Continual Learning
- Chelsea Finn et.al 2017 Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks(ICML 2017)
- David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In Advances in Neural Information Processing Systems
- Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research (JAIR)
- Kakade et.al 2001, A Natural Policy Gradient
- Schulman et.al 2015, Trust Region Policy Optimization
- Schulman et.al 2017, Proximal Policy Optimization Algorithms
- Abel et.al 2023, A Definition of Continual Reinforcement Learning
- Hessel et.al 2017, Rainbow: Combining improvements in Deep Reinforcement Learning
- Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor
- Wang et.al 2023, A Comprehensive Survey of Continual Learning: Theory, Method and Application

Abel, D., Barreto, A., Roy, B. V., Precup, D., van Hasselt, H., and Singh, S. (2023). A definition of continual reinforcement learning.

Ben-Iwhiwhu, E., Nath, S., Pilly, P. K., Kolouri, S., and Soltoggio, A. (2023). Lifelong reinforcement learning with modulating masks.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.