Operator World Models for Reinforcement Learning

Pietro Novelli[1], Marco Pratticò[1], Massimiliano Pontil[1,2], Carlo Ciliberto[2]

pietro.novelli@iit.it, marco.prattico@iit.it, massimiliano.pontil@iit.it, c.ciliberto@ucl.ac.uk

- Operator Learning & Uncertainty quantification -

M2DS

Author : Zakaria **AKIL**

Supervisor : Lounici Karim

Ecole Polytechnique | IP Paris

August 6, 2025

Contents

1 Introduction 2
2 POWR Algorithm 3
3 Experiments 4
3.1 Paper's experiments: 4
3.2 POWR on new environment: 5
4 Critical Analysis 5
5 Conclusion 5
6 References 5

1 Introduction

Reinforcement Learning

Reinforcement Learning (RL) is a crucial subfield of Artificial Intelligence (AI) focused on training agents to make optimal decisions by interacting with their environment, aiming to maximize cumulative rewards. Unlike supervised learning, which relies on labeled datasets, or unsupervised learning, which detects patterns without predefined outcomes, RL emphasizes learning through trial and error, where agents evaluate the consequences of their actions.

The fundamental components of RL are states, actions, policies, rewards, and transition dynamics, each playing a critical role in the agent's learning process. A *state*, denoted as S_t, represents the agent's perception of the environment at a specific moment t, and it can either be discrete or continuous. *Actions*, represented by A_t, are the possible moves the agent can take within a given state, and these actions can also be either discrete or continuous.

The *policy*, denoted by π, defines how the agent chooses actions based on the current state. It can be either stochastic or deterministic. A *stochastic policy*, $\pi(a|s)$, assigns a probability distribution over possible actions for a given state, meaning the agent samples actions based on these probabilities. In contrast, a *deterministic policy*, $a = \pi(s)$, directly maps a state to a specific action, eliminating any randomness.

Rewards provide feedback on the agent's actions, helping it distinguish between positive and negative outcomes, and refining its policy accordingly. *Transition dynamics*, represented as $P(s'|s, a)$, describe the probability of transitioning from one state to another after taking a given action, capturing the behavior of the environment.

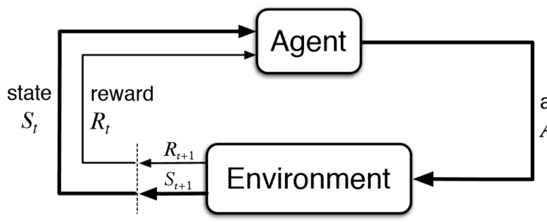


Figure 1: Action-reward feedback loop of a generic RL model [2]

RL problems are often framed as a Markov Decision Process (MDP), characterized by a finite set of states $\mathcal{S}$, actions $\mathcal{A}$, transition kernel τ, reward functions $\mathcal{R}$, and a discount factor $\gamma \in [0, 1]$, which balances immediate and future rewards. We denote the MDP :

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \tau, \mathcal{R}, \gamma)$$

The goal is then to maximize the expected return given by :

$$G_t = \sum_{k=t-1}^{T} \gamma^{k-t-1} R_k$$

So the problem can be formulated as follows :

$$\boxed{max_\pi \mathbb{E}_\pi [G_t]}$$

We define the value functions: the state-value function v_π and the action-value function q_π, given by:

$$v_\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1} \mid S_t = s], s \in \mathcal{S}$$

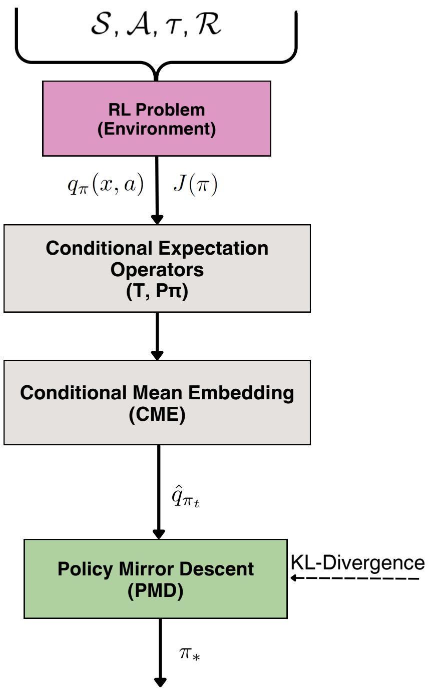
$$q_\pi(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1} \mid S_t = s, A_t = a], s \in \mathcal{S}, a \in \mathcal{A}$$

Paper Overview & Problem Statement

This paper addresses the optimization of the policy using Policy Mirror Descent (PMD) which is a powerful optimization algorithm that extends the classic proximal gradient descent to a non-euclidean spaces. Many other papers have already proposed PMD in RL but their solutions missed efficiency and exact convergence rates guaranties. The paper focuses on how to efficiently deploy it in Reinforcement Learning (RL) settings, particularly under conditions where only inexact estimators of the action-value function are available. Traditional PMD methods require explicit knowledge of action-value functions for all policies, which is often impractical in RL due to the need for extensive sampling and interactions with the environment. This results in computational inefficiency and convergence rates that depend on unrealistic assumptions such as access to perfect simulators. Using operator learning theory, this paper seeks to overcome these limitations by proposing a novel approach to estimate the action-value function through Conditional Mean Embeddings (CMEs), thus reducing the dependence on extensive sampling and providing a more computationally efficient alternative.

2 POWR Algorithm

The introduced algorithm, POWR, is composed of two primary components: Mirror Descent utilized in policy optimization and the operator-based estimation of the action-value function. The figure below represents an overview of the whole algorithm.



$\mathcal{S}, \mathcal{A}, \tau, \mathcal{R}$
RL Problem (Environment)
$q_\pi(x,a)$
$J(\pi)$
Conditional Expectation Operators (T, Pπ)
Conditional Mean Embedding (CME)
$\hat{q}_{\pi_t}$
Policy Mirror Descent (PMD)
KL-Divergence
π_*

Figure 2: POWR Architecture

Policy Mirror Descent

Mirror descent is an optimization problem originally used for convex optimazation but it turns that it enjoyed global convergence with sublinear or even sublinear rates at the cost of dimension dependent constants even when applied to maximizing the expected return.

Policy Mirror Descent is a method for optimizing the policy π in Markov Decision Processes (MDP) using the action-value function q_π

Let's consider the MDP :

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, \tau : \Omega \to \mathcal{P}(\mathcal{X}), \nu \in \mathcal{P}(\mathcal{X}), r : \Omega \to \mathbb{R}, \gamma)$$

where $\Omega = \mathcal{X} x \mathcal{A}$, ν is initial state distribution, r is the reward function, and $\gamma > 0$ is a discount coefficient.

We recall that a policy $\pi : \mathcal{X} \to \mathcal{P}(\mathcal{A})$ is a Borel measurable function mapping states to probability distributions over actions.

The objective is to maximize the expected return:

$$J(\pi) = \mathbb{E}_{\nu,\pi,\tau}\left[\sum_{t=0}^{\infty} \gamma^t r(X_t, A_t)\right], \tag{1}$$

where $(X_t, A_t)_{t\in\mathbb{N}}$ denotes the Markov process induced by π and τ, with $X_0 \sim \nu$, $A_t \sim \pi(\cdot|X_t)$, and $X_{t+1} \sim \tau(\cdot|X_t, A_t)$.

PMD update is given by :

$$\pi_{t+1}(\cdot|x) = \underset{p\in\Delta(\mathcal{A})}{\arg\min}\left[-\eta\langle q_{\pi_t}(\cdot, x), p\rangle + D(p, \pi_t(\cdot|x))\right], \tag{2}$$

where $\eta > 0$ is a step size, D is a Bregman divergence, and $q_\pi : \Omega \to \mathbb{R}$ is the action-value function defined as:

$$q_\pi(x, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(X_t, A_t) \mid X_0 = x, A_0 = a\right]. \tag{3}$$

In this algorithm, Bregman Divergence is taken as Kullback-Leibler divergence:

$$D = D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}$$

Efficient Estimation of action-value function

A significant contribution of the paper is the extension of PMD to settings where exact knowledge of q_{π_t} is not available. Instead, the authors propose learning estimators $\hat{r}$ and $\hat{\tau}$ for the reward and transition distributions, respectively, and using these to estimate q_π.

This reduces the need for extensive interaction with the MDP, making the approach more practical in reinforcement learning scenarios.

The relevance of this method lies in its ability to handle the approximation errors in q_{π_t} while retaining convergence guarantees.

If the approximation error $\|\hat{q}_{\pi_t} - q_{\pi_t}\|_\infty \le \epsilon$ for all $t = 1, \ldots, T$, then:

$$\max_\pi J(\pi) - J(\pi_T) \le O(\epsilon + \frac{1}{T}). \tag{4}$$

This result underscores the practical utility of PMD in reinforcement learning by providing a robust framework for policy optimization under uncertainty.

Operator World Models

The POWR algorithm innovatively incorporates Conditional Mean Embeddings (CME) to model the transition operator of the Markov Decision Process (MDP). This approach facilitates the estimation of the action-value function in closed form, leveraging matrix operations to handle both finite and infinite state settings efficiently.

Conditional Expectation Operators

The key to this methodology is defining a transfer operator T associated with the MDP transition distribution τ. This operator enables the estimation of the expected next state given the current state and action pair. In addition, we consider another Markov Operator P_π that encodes the policy's action distribution given a state. Formally, $\forall f \in \mathcal{B}(\mathcal{X})$ and $\forall g \in \mathcal{B}(\Omega)$:

$$(Tf)(x,a) = \int_X f(x')\tau(dx'|x,a) = \mathbb{E}[f(X')|x,a].$$

$$(P_\pi f)(x) = \int_X g(x,a)\pi(da|x) = \mathbb{E}[g(X,A)|X=x].$$

This operatorial framework provides a robust foundation for expressing the action-value function and the expected return in a RL problems:

$$q_\pi(x,a) = \sum_{t=0}^{\infty}(\gamma T P_\pi)^t r = (Id - \gamma T P_\pi)^{-1} r$$

$$J(\pi) = P_\pi (I - \gamma T P_\pi)^{-1} r, \quad \nu = \langle P_\pi q_\pi, \nu \rangle$$

Conditional Mean Embeddings

The paper introduces the use of CMEs to estimate the transfer operator and the reward function. By learning these components as operators between suitably chosen Sobolev spaces, the action-value function q_π for any policy π can be computed efficiently by:

$$\boxed{q_\pi(x,a) = (Id - \gamma \hat{T}_n P_\pi)^{-1} \hat{r}_n},$$

Where T_n, and r_n are learned by minimizing the squared loss.

$$T_n = \underset{T \in \mathcal{HS}(F,G)}{\arg\min} \left(\frac{1}{n} \sum_{i=1}^{n} \left\| \varphi(x_i') - T^* \psi(x_i, a_i) \right\|_F^2 + \lambda \left\| T \right\|_{\mathcal{HS}}^2 \right)$$

$$\boxed{T_n = S_n^* K_\lambda^{-1} Z_n}$$

$$\boxed{r_n = S_n^* b = \sum_{i=0}^{n} b_i \phi(x_i, a_i)}$$

Where $b = K_\lambda^{-1} y$, S_n and Z_n are operators from the restricted Sobolev spaces to $\mathbb{R}^n$, $K_\lambda = S_n S_n^* + n\lambda Id_n$ the regularized Gram matrix, and data entries $y_i = r(x_i, a_i)$.

POWR Policy Update

Combining these operator-based estimations with PMD results in the POWR algorithm, which iteratively updates the policy using the estimated action-value function. The policy update can be written in close form and expressed using SOFTMAX operator as :

$$\boxed{\pi_{t+1}(\cdot|x) = \text{SOFTMAX}\left(\log \pi_0(\cdot|x) + \eta \sum_{s=0}^{t} \hat{q}_{\pi_s}(x,\cdot)\right)}$$

where η is the step size.

3 Experiments

3.1 Paper's experiments:

The experiments were conducted on classical Gym environments. The method alternates between policy execution for sample collection and running POWR to update the policy.

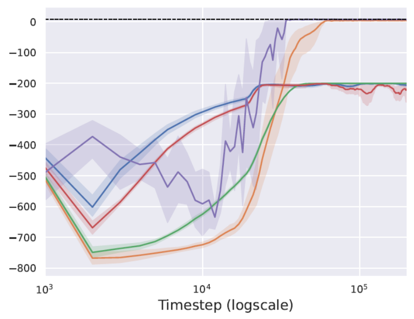


Figure 3: Performance on Taxi environment [1]

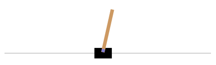
Environment	POWR	Performance Metric
FrozenLake-v1	Outperforms	Higher sample efficiency
Taxi-v3	Avoids local optima	Higher reward with fewer steps
MountainCar-v0	Outperforms	Faster convergence

Table 1: Performance Summary Across Different Environments

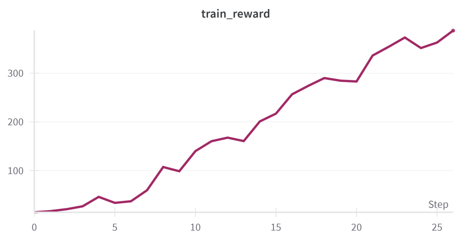
According to the the authors experiments and results, the method seems consistently outperforms baselines in sample efficiency, achieving higher rewards in fewer timesteps.

3.2 POWR on new environment:

I used the POWR algorithm for several tasks, such as CartPole, LunarLander, and others. The learning process takes a long time, but in some cases, such as CartPole, the algorithm tends to lose stability after a few steps within the episodes. However, the algorithm shows signs of convergence.



((a)) CartPole env



((b)) Train cumulative reward

4 Critical Analysis

The proposed method demonstrates several key advantages. Firstly, it significantly outperforms well-established baselines such as A2C, DQN, TRPO, and PPO in terms of sample complexity, achieving higher cumulative rewards with fewer interactions. This efficiency is particularly notable in environments like Taxi-v3, where the method avoids converging to local optima, a common issue faced by other algorithms. Furthermore, the efficient integration of policy mirror descent (PMD) offers a fresh approach to reinforcement learning, with the potential for global optimality and convergence at a polynomial rate, making it a promising contribution to the field. In fact, Previous applications of Mirror Descent in Reinforcement Learning often overlook the estimation errors of the action-value function, failing to distinguish them from the objective estimation error. Additionally, they do not thoroughly examine the impact of PMD approximation on overall convergence. While this paper acknowledges these issues, it falls short of providing a comprehensive analysis of the associated trade-offs. The reliance on approximation methods such as Monte Carlo for PMD updates in infinite action spaces further raises concerns about their effect on convergence rates.

One notable drawback is its instability during the initial stages of training, likely stemming from a suboptimal balance between exploration and exploitation. This instability could hinder early performance and requires further refinement. Additionally, the exploration-exploitation trade-off, a critical aspect of reinforcement learning, needs to be better managed to enhance the method's stability. Regarding the Operator Learning Theory, the approach heavily relies on Sobolev spaces to ensure the well-posedness of the Conditional Mean Embeddings (CMEs). This could limit the generalization to other function spaces, potentially restricting the applicability of the method across diverse environments.

5 Conclusion

To summarize, the POWR algorithm improves reinforcement learning by combining Policy Mirror Descent with operator-based action-value function estimation using Conditional Mean Embeddings. It enhances sample efficiency and convergence speed, outperforming traditional RL methods in various environments. While it shows promising results in avoiding local optima and achieving higher rewards with fewer interactions, further work is needed to refine exploration-exploitation strategies, address stability issues, and maybe work on problems where the action space is not finite which is a still a big question to tackle in this field. Overall, POWR offers a promising approach for efficient policy optimization in RL tasks.

6 References

1. [Reviewed Paper]"Operator World Models for Reinforcement Learning", by Pietro Novelli and Marco Pratticò and Massimiliano Pontil and Carlo Ciliberto, 2024, arXiv, eprint=2406.19861, Paper link.