Eigenoptions in the face of partial observability

by

Marcos Menon José

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

This thesis characterizes how eigenoptions behave in the face of partial observability and investigates strategies to recover their benefits. Eigenoptions—temporally extended actions (options) derived from the spectral structure of the successor representation—are known to facilitate exploration in fully observable Markov decision processes, but their behavior in partially observable settings remains poorly understood. We show that, when learned from aliased observations, the intrinsic rewards that eigenoptions maximize frequently induce cyclic behaviors that trap the agent in small regions of the state space, degrading both coverage and downstream control performance.

To address these failures, we incorporate trajectory information through multi-step credit assignment into the policy learning step and study the benefits of both memoryless and memory-based option learners. First, we introduce an unbiased offline multi-step learning algorithm for options that combines ACER and a multiple importance sampling correction tailored to option-generated data. This algorithm provides principled off-policy corrections for trajectories produced by mixtures of options and primitive actions, enabling multi-step updates for target policies. Second, we evaluate recurrent eigenoption learners based on DRQN and recurrent ACER, which maintain memory over observation histories to mitigate state aliasing.

Across multiple partially observable environments, we find that tabular eigenoption learners can be worse than acting without options, whereas a memoryless ACER agent equipped with our multi-step credit assignment update discovers eigenoptions that reliably accelerate state-space coverage and improve performance on goal-reaching tasks relative to baseline methods.

Preface

This thesis is an original work by Marcos Menon José. No part of this thesis has been previously published.

Acknowledgements

I am profoundly grateful to my supervisor, Marlos, who taught me far more than reinforcement learning. His guidance, patience, and encouragement carried me through the most difficult moment of my life and helped me push through to the end. His advice, both academic and personal, is something I will carry with me forever.

I owe my deepest gratitude to my wife, Maria Fernanda. None of this would have been possible without her unwavering support and the sacrifices she made so I could pursue my dream of studying in another country. She stood by me through every challenge, helped me navigate each step of this journey, and, together with Atena, made our home the place I most looked forward to returning to every single day.

I want to express my thanks to my brother, Luis, who has always been someone I could count on, no matter the distance. Even from another country, he was there for me in every moment I needed support. I am equally grateful to my mother, Márcia, and my father, Artur, for being with me throughout this journey, offering love, encouragement, and strength.

I want to thank my friends who made this journey. Harshil and Mohamed were by my side throughout my master's, studying together, discussing research, and supporting each other through everything. I am grateful to Prabhat for his wisdom and invaluable advice, and to Rick for always being someone I could talk to when things felt overwhelming. I also want to thank everyone in Marlos's lab for their insights, ideas, and many research provocations that shaped my work: Siddarth, Diego, Lucas, Dikshant, Alex, and Brett.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

At a high level, this thesis investigates how to learn useful forms of temporally extended behavior in reinforcement learning when the agent does not have full access to the underlying state of the environment. We focus on the challenges that arise when an agent must rely on incomplete observations and on how structured exploratory behaviors can still be learned in these settings. Along the way, we develop a new offline learning algorithm that enables effective training from data generated under temporally extended decisions.

In reinforcement learning (RL), an agent interacts with an environment to maximize cumulative rewards. The RL framework is often described through an interaction that takes place in discrete steps in which, at each time step, the agent receives an observation and a reward and then selects a primitive action. Unlike agents that operate through primitive, low-level actions, humans typically act using higher-level behaviors that span multiple time steps. The options framework (Sutton, Precup, and Singh 1999) has become one of the most widely used approaches to bridge this gap, enabling RL agents to learn and operate with temporal abstractions.

Option discovery—the automatic process of learning options— remains an open problem in RL, with a wide range of methods proposed to automatically construct useful temporal abstractions (for a complete review, see the survey by Klissarov et al. 2025). Eigenoptions (Machado, Bellemare, and Bowling 2017) is a particularly

promising class of options. Derived from the successor representation (Dayan 1993), these options capture the underlying structure of the environment by maximizing intrinsic rewards that induces policies that lead agents to distant regions of the state space. By doing so, they promote diverse exploration, improving state-space coverage (Machado et al. 2023). Eigenoptions have been successfully applied in both tabular (Machado, Bellemare, and Bowling 2017; Machado et al. 2023) and deep RL settings (Klissarov and Machado 2023; Machado et al. 2018).

While eigenoptions have shown strong potential for enhancing exploration, most studies assume environments are fully observable. Yet, in many real-world scenarios, agents must act under partial observability, where they receive only limited or noisy information about the underlying state. Formally modeled as Partially Observable Markov Decision Processes (POMDPs) (Kaelbling, Littman, and Cassandra 1998), such problems often require agents to infer *latent state representations*—internal summaries of past observations that approximate the hidden environment state—from experience.

Motivated by this gap, we investigate eigenoption discovery in the face of partial observability. We aim to characterize the specific challenges partial observability introduces and to explore how temporally-extended behaviors can help overcome them. Under partial observability, acting optimally in principle requires conditioning decisions on the entire interaction history (Kaelbling, Littman, and Cassandra 1998), which is infeasible in practice. Instead, agents often learn compact summaries of past experience that retain information relevant for long-term outcomes. To this end, we focus on two mechanisms: multi-step credit assignment and the use of memory. Multi-step methods help propagate information across time, allowing the agent to better capture delayed dependencies that may be obscured under limited observations. On the other hand, memory equips the agent with an internal state that aggregates and compresses past observations into a useful latent representation.

Most option discovery methods, including eigenoptions, require off-policy learning

as they discover multiple options at the same time while acting according a different policy. Most existing work on options, however, sidesteps the need for off-policy corrections by relying on Q-learning (Watkins and Dayan 1992) or Q-learning style updates (Klissarov and Machado 2023; Machado et al. 2023; Sutton, Precup, and Singh 1999), which do not require explicitly addressing mismatches in action distributions. However, this only allows for one-step unbiased updates.

Incorporating multi-step credit assignment within the options framework raises its own set of difficulties. Because multi-step methods rely on returns that span several time steps, they require reusing trajectories collected under different option-conditioned policies. A key challenge, then, lies in how to effectively leverage these transitions to train multiple policies, given that the probability distribution over actions depends on the specific option being followed. When learning from data generated by a different behavior policy, importance sampling provides a principled way to correct for this discrepancy. It reweights each sampled transition by the ratio between the target policy's probability of selecting an action and that probability under the behavior policy, ensuring unbiased estimates (Precup, Sutton, and Singh 2000).

Incorporating multi-step returns, such as n-step or λ-returns, within the options framework introduces additional complexity. While some prior work has explored this direction (Guo, Thomas, and Brunskill 2017; Jain and Precup 2018; Klissarov and Precup 2021), these approaches often focus on online learning or assume that all options and meta-policy maximize the same reward function. To our knowledge, offline learning from data generated by options—for instance, when optimizing alternative reward functions for option discovery—has only been investigated without the use of importance sampling corrections (Klissarov and Machado 2023; Machado et al. 2023; Ramesh, Tomar, and Ravindran 2019; Sutton, Precup, and Singh 1999).

Given that multi-step credit-assignment can help avoid some of the issues of partial observability (Allen et al. 2024), we propose a new algorithm that introduces an

importance sampling correction tailored for offline learning with data generated under options. This correction enables the use of multi-step credit assignment while learning eigenoptions, ensuring that the mismatch in action distributions is properly accounted for. Furthermore, we prove that our approach yields an unbiased estimator.

In parallel, we evaluate the role of memory in improving learning the options policies under partial observability. Specifically, we investigate recurrent neural architectures (Hopfield 1982), which allow agents to aggregate information from past observations to disambiguate hidden states. Incorporating such memory mechanisms has become one of the most widely adopted strategies in deep reinforcement learning as in the Deep Recurrent Q-Network (DRQN) (Hausknecht and Stone 2015), IMPALA (Espeholt et al. 2018), and R2D2 (Kapturowski et al. 2019).

While partial observability distorts the successor representation—because observations alias multiple latent states—this thesis does not attempt to correct these distortions. Instead, we take the observation-based successor representation as given and focus on how to learn effective eigenoption policies from the intrinsic rewards induced by its eigenvectors. Our contributions therefore concern the optimization of option policies under partial observability, rather than the design of representation-learning methods that reconstruct an undistorted eigenstructure.

1.1 Contribution

The main contributions of this thesis can be summarized as follows:

- **Characterization of the effects of partial observability on eigenoptions.** We analyze how the partial observability impacts the structure, utility, and interpretability of eigenoptions, highlighting the challenges and opportunities that arise when extending them beyond fully observable environments.

- **Development of a method for offline multi-step learning with options.** We introduce an algorithm that incorporates an unbiased importance sampling

correction, enabling the use of multi-step credit assignment when learning from data generated under options.

- **Evaluation of trajectory-based and memory-based approaches under partial observability.** We empirically investigate how trajectory-based multi-step credit assignment can mitigate the difficulties of policy learning in the face of partial observability, assessing their impact on option discovery and downstream performance.

We also make all source code publicly available at https://github.com/machado-research/eigenoptions-partial-observability.

1.2 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 reviews the foundations of sequential decision making and reinforcement learning. Chapter 3 characterizes the interaction between eigenoptions and partial observability, identifying the main challenges that arise in this setting. Chapter 4 presents our proposed algorithm for offline learning with options, including the importance sampling correction and a theoretical analysis of its unbiasedness. Chapter 5 reports our experimental results, evaluating eigenoptions under partial observability with multi-step returns and memory-based agents. Finally, Chapter 6 concludes with a discussion of the main contributions, limitations, and directions for future work.

Chapter 2

Background

In this chapter, we describe the mathematical formulation of sequential decision making problems and of reinforcement learning. Additionaly, we describe off-policy estimation, temporal abstractions via the options framework, eigenoptions, and the deep RL algorithms we used in this thesis.

2.1 Sequential decision-making problem

We formalize sequential decision making as stochastic control over time, specifying states, actions, transitions, rewards.

2.1.1 Markov decision processes

We frame sequential decision making as an agent interacting with an environment to maximize expected cumulative reward. Following Sutton and Barto (2018) notation, the interaction is modeled as a (finite) Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ the set of actions, $\mathcal{R} \subseteq \mathbb{R}$ the set of possible rewards, and γ is the discounting factor. The environment dynamics is described by the transition function $p(s', r \mid s, a) \doteq \Pr\{S_{t+1} = s',\, R_{t+1} = r \mid S_t = s, A_t = a\}$. This formulation induces trajectories $S_0, A_0, R_1, S_1, A_1, R_2, \ldots$ that satisfy the Markov property encoded by $p(\cdot, \cdot \mid s, a)$, meaning that the distribution over (S_{t+1}, R_{t+1}) depends only on the current state–action pair (S_t, A_t) and not on the past history.

2.1.2 Partially observable Markov decision process

In many real-world scenarios, the agent cannot directly observe the underlying state of the environment. Instead, it receives partial observations that provide only indirect information about the true state of the environment. Reinforcement learning under such conditions is modeled as a partially observable Markov decision process (POMDP), represented as the 8-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, p, \mathcal{T}_0, \mathcal{Z}, \mathcal{R}, \gamma)$ (Kaelbling, Littman, and Cassandra 1998), where $\mathcal{S}$ denotes the set of true states, $\mathcal{A}$ the set of actions, $\mathcal{O}$ the observation space, p the transition kernel, $\mathcal{T}_0$ the initial-state distribution, $\mathcal{Z}$ the observation model, $\mathcal{R}$ the reward function, and γ the discounting factor.

The environment dynamics are described by the transition function $p(s', r \mid s, a) \doteq \Pr\{S_{t+1} = s',\, R_{t+1} = r \mid S_t = s,\, A_t = a\}$, which defines the likelihood of transitioning from state s to s' under action a. At each time step, the agent receives an observation drawn from

$$\mathcal{Z}(o_t \mid s_t) : \mathcal{O} \times \mathcal{S} \to [0, 1],$$

which specifies the distribution over observations conditioned on the current unobservable state.

Because state s_t is not directly accessible, the agent usually maintains a *belief state*, $b_t \in \Delta(\mathcal{S})$, which is a probability distribution over $\mathcal{S}$ that represents its uncertainty about the current state. A policy can then be defined over belief states, $\pi(a_t \mid b_t)$, or, equivalently, over the history of interactions

$$h_t = (o_0, a_0, r_1, o_1, \ldots, a_{t-1}, r_t, o_t).$$

2.2 Reinforcement learning

Reinforcement learning tackles sequential decision-making problems by learning a *policy* that maximizes the sum of discounted rewards under the POMDP model introduced above. A (stochastic) policy $\pi(a \mid h)$ prescribes how the agent selects actions in each state, and the control objective is to find an optimal policy $\pi^\star$ that maximizes

the expected return:

$$\pi^\star \;\in\; \arg\max_{\pi}\; \mathbb{E}_\pi[G_t]\,, \qquad G_t \;\doteq\; \sum_{k=0}^{\infty} \gamma^k\, R_{t+k+1}, \tag{2.1}$$

where G_t is the (discounted) return from time t and $\gamma \in [0, 1)$ is the discount factor that trades off near- and far-term rewards.

To reason about and optimize policies, RL uses *value functions* that quantify expected return when acting according to π. The state-value function

$$V^\pi(s) \;\doteq\; \mathbb{E}_\pi[\, G_t \mid S_t = s \,] \tag{2.2}$$

measures the quality of being in state s, and the action-value function

$$Q^\pi(s, a) \;\doteq\; \mathbb{E}_\pi[\, G_t \mid S_t = s,\, A_t = a \,] \tag{2.3}$$

measures the quality of taking action a in s and following π thereafter. These functions provide a common backbone for RL algorithms: *policy evaluation* estimates V^π or Q^π for a given π, while *policy improvement* uses these estimates to refine π, iterating toward $\pi^\star$. This way, the objective, return, and value functions are linked in a closed loop that underlies both classical dynamic programming and modern model-free methods.

2.2.1 Importance sampling and off-policy corrections

Off-policy RL addresses the problem of learning about a *target policy* while using data generated by a different *behavior policy*, i.e., when the distribution under which data is collected differs from the distribution of interest. *Importance Sampling (IS)*(Rubinstein and Kroese 1981) is used to correct this mismatch and compute the expectation of a random variable x under a target distribution d using samples drawn from a different distribution d':

$$\mathbb{E}_d[x] = \mathbb{E}_{d'}\left[x\frac{d(x)}{d'(x)}\right] \approx \frac{1}{n}\sum_{i=1}^{n} x_i \frac{d(x_i)}{d'(x_i)}. \tag{2.4}$$

In the context of off-policy RL, the IS correction ratio, $\rho_{t:T-1}$, is the product of action probability ratios between the target policy, π, and the behavior policy, b, computed over a trajectory from time step t to $T-1$ (Precup, Sutton, and Singh 2000):

$$\rho_{t:T-1} = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}, \tag{2.5}$$

where $p(S_{k+1}|S_k, A_k)$ is the environment's transition probability, which cancels out under the assumption that the same dynamics govern both trajectories.

Using this correction ratio, the value function can be estimated as

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right] = \mathbb{E}_b \left[\rho_{t:T-1} G_t \mid S_t = s \right], \tag{2.6}$$

and similarly, the action-value function can be estimated as

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_b \left[\rho_{t+1:T-1} G_t \mid S_t = s, A_t = a \right], \tag{2.7}$$

where $\rho_{t+1:T-1}$ excludes the ratio at time t since action A_t is already conditioned on.

The corresponding empirical estimators over multiple trajectories are:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}, \tag{2.8}$$

$$Q(s, a) \doteq \frac{\sum_{t \in \mathcal{T}(s,a)} \rho_{t+1:T(t)-1} G_t}{|\mathcal{T}(s, a)|}, \tag{2.9}$$

where $\mathcal{T}(s)$ and $\mathcal{T}(s, a)$ denote the time steps where state s, and state-action pair (s, a), respectively, occurred in the dataset.

The update rules for incremental updates for Monte-Carlo methods using a step size α are:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[\rho_{t:T(t)-1} G_t - V(S_t) \right], \tag{2.10}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[\rho_{t+1:T(t)-1} G_t - Q(S_t, A_t) \right]. \tag{2.11}$$

Multiple Importance Sampling

While standard importance sampling corrects for the mismatch between a single behavior policy and a target policy, it assumes that all data are drawn from the same

underlying distribution. In practice, however, data may come from multiple sources—for instance, when trajectories are collected under several different behavior policies b_f. In such cases, combining samples using standard IS can lead to high-variance estimates if the individual policies differ substantially (Veach 1998).

Multiple Importance Sampling (MIS) (Veach 1998) generalizes IS to this multi-distribution setting by constructing a single unbiased estimator that accounts for all sampling distributions simultaneously. Let data be drawn from F different distributions d_f, each corresponding to a behavior policy or data source. Let $X_{k,f}$ be the k-th sample from distribution d_f, and $\eta_f(x)$ be a weighting function that determines the relative contribution of each source. The MIS estimator for the expectation under a target distribution d_π is:

$$\mathbb{E}_{d_\pi}[x] \approx \sum_{f=1}^{F} \frac{1}{N_f} \sum_{k=1}^{N_f} \eta_f(X_{k,f})\, X_{k,f} \frac{d_\pi(X_{k,f})}{d_f(X_{k,f})}, \tag{2.12}$$

where N_f is the number of samples drawn from source f. This estimator remains consistent and unbiased provided that (Veach 1998):

- For all x such that $d_\pi(x) \neq 0$, the weights satisfy $\sum_{f=1}^{F} \eta_f(x) = 1$,
- If $d_f(x) = 0$, then $\eta_f(x) = 0$.

MIS encompasses standard importance sampling as the special case where all data are drawn from a single behavior policy ($F = 1$). Moreover, by choosing appropriate weighting functions $\eta_f(x)$, MIS can significantly reduce the variance of off-policy estimates while preserving unbiasedness (Veach 1998). This makes it potentially effective in reinforcement learning settings where experience is aggregated from multiple behavior policies or replay buffers, providing a more stable and sample-efficient estimate of the target value function.

Using these properties, one can construct a MIS estimator for the value function with respect to the target policy π, using data collected from multiple behavior policies b_f. Let $G_{i,f}(s)$ denote the return from the i-th episode generated under behavior

policy b_f, starting from state s. The MIS estimator for the value function is then given by:

$$V_\pi(s) = \sum_{f=1}^{F} \frac{1}{N_f} \sum_{i=1}^{F} \eta_f(s)\, G_{i,f}(s) \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b_f(A_k \mid S_k)}. \tag{2.13}$$

2.2.2 Deep reinforcement learning

Deep RL combines reinforcement learning objectives with deep neural networks used as function approximators for value functions and policies. Instead of storing a separate parameter for every state or state-action pair, function approximation represents quantities such as $V(s)$, $Q(s,a)$, or $\pi(a \mid s)$. Deep neural networks are particularly suitable in this role because they can learn hierarchical feature representations directly from high-dimensional inputs (e.g., images or sequences of observations), allowing the agent to extract relevant structure for decision making while updating the shared parameters through gradient-based methods driven by RL objectives.

Deep Q-Network (DQN)

Q-learning (Watkins and Dayan 1992) is an algorithm that updates the value of the executed state–action pair by backing up the *greedy* estimate of the value function over next actions (i.e., toward π_*) rather than the action actually taken:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\big]. \tag{2.14}$$

Mnih et al. (2015) augmented Q-learning with a neural network $Q(s,a;w)$ as a function approximator with weights represented as w. Because sequential updates along a single trajectory in a deep network were unstable, they used two main stabilization mechanisms: experience replay (Lin 1992), which stores past transitions and trains on approximately i.i.d. minibatches sampled from a replay buffer, and a target network with parameters w' that lags behind the online network. For a minibatch $\{(s_i, a_i, r_i, s_i')\}_{i=1}^{N}$, the training target and loss are

$$y_i = r_i + \gamma \max_{a'} Q(s_i', a'; w'), \qquad \mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^{N} \big(y_i - Q(s_i, a_i; w)\big)^2. \tag{2.15}$$

These mechanisms reduce correlation in updates and prevent the target from shifting too rapidly, yielding more stable learning.

Deep Recurrent Q-Network (DRQN)

While DQN assumes access to fully observable Markov states, many environments provide only partial observations, violating the Markov property and degrading value estimates. The *Deep Recurrent Q-Network (DRQN)* proposed by Hausknecht and Stone (2015) augments the DQN architecture with a recurrent neural network—typically an LSTM (Hochreiter and Schmidhuber 1997) or GRU (Cho et al. 2014)—to maintain an internal hidden state that summarizes past observations. This hidden state acts as a belief representation, enabling the agent to potentially approximate $Q(s_t, a_t)$ using the entire observation history rather than a single frame.

Formally, DRQN receives a single observation o_t at each time step and updates a latent state:

$$h_t = f_{\text{RNN}}(h_{t-1}, \phi(o_t)), \tag{2.16}$$

where ϕ is a convolutional or MLP encoder, and f_{RNN} denotes the recurrent update function.

Training mirrors DQN but must account for sequential dependencies. Instead of sampling individual transitions, DRQN samples short sequences $\{(o_t, a_t, r_{t+1}, o_{t+1})\}_{t=k}^{k+L-1}$ from the replay buffer and unrolls the recurrent network across them. With a target network w', the bootstrap target for each time step is

$$y_t = r_{t+1} + \gamma \max_{a'} Q\big(h'_{t+1}, a'; w'\big)\,, \tag{2.17}$$

where h'_{t+1} is the hidden state computed by the target network along the same unrolled sequence. The loss over a minibatch of N sequences of length L is

$$\mathcal{L}(w) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{L}\sum_{t=1}^{L}\Big(y_t^{(i)} - Q(h_t^{(i)}, a_t^{(i)}; w)\Big)^2. \tag{2.18}$$

By using a recurrence over observations, DRQN reduces the detrimental effects of perceptual aliasing and enables more robust action-value estimation in POMDP settings, while keeping the overall training pipeline largely consistent with DQN.

Actor–Critic with Experience Replay (ACER)

ACER (Wang et al. 2017) extends the actor–critic framework (Barto, Sutton, and Anderson 1983) to multi-step off-policy learning in deep RL while maintaining stability. It enables the reuse of past trajectories through experience replay, as in DQN, while employing the Retrace(λ) method (Munos et al. 2016) to compute stable off-policy value estimates with truncated importance sampling corrections.

Retrace(λ) is a multi-step off-policy return estimator designed to balance bias and variance when learning from trajectories collected under a different behavior policy. Instead of relying on full-importance sampling, which can lead to high variance when the target and behavior policies diverge, Retrace(λ) truncates the per-step importance weights to ensure bounded updates while maintaining convergence toward the correct value function. By propagating returns backward with exponentially decaying coefficients controlled by $\lambda \in [0, 1]$, it interpolates between 1-step temporal-difference learning and Monte Carlo targets.

Formally, the Retrace(λ) return is given by

$$Q^{\text{ret}}(S_t, A_t) = Q(S_t, A_t) + \sum_{k=t}^{T-1} \gamma^{k-t} \left(\prod_{i=t+1}^{k} c_i \right) \delta_k, \tag{2.19}$$

where $\delta_k = R_k + \gamma V(S_{k+1}) - Q(S_k, A_k)$ is the temporal-difference (TD) error, $c_i = \lambda \min\left(1, \frac{\pi(A_i|S_i)}{b(A_i|S_i)}\right)$ is the truncated importance coefficient, π is the target policy, b is the behavior policy, $\gamma \in [0, 1)$ is the discount factor, R_k is the reward at step k, and T is the length of the trajectory. This formulation yields a contraction mapping toward the correct Q-function while bounding variance by limiting the effect of large policy ratios (Munos et al. 2016).

Policy and Value Updates At each time step t, the actor network parameterized by θ outputs a stochastic policy $\pi_\theta(a_t|s_t)$, and the critic network parameterized by ϕ estimates the state-value $V_\phi(s_t)$ and action-value $Q_\phi(s_t, a_t)$. To correct for off-policy data, ACER computes the importance weight between the target and behavior policies:

$$\rho_t = \frac{\pi_\theta(a_t|s_t)}{b(a_t|s_t)}, \qquad \bar{\rho}_t = \min(c, \rho_t), \tag{2.20}$$

where $c > 0$ is a truncation constant used to limit variance.

The critic update relies on the Retrace(λ) target,

$$Q^{\text{ret}}(S_t, A_t) = r_t + \gamma V_\phi(s_{t+1}) + \gamma \bar{\rho}_{t+1}\big(Q^{\text{ret}}(S_{t+1}, A_{t+1}) - Q_\phi(s_{t+1}, a_{t+1})\big), \tag{2.21}$$

which bootstraps from the next state's value while safely truncating the off-policy correction.

The actor update combines the truncated ratio with a bias-correction term to ensure the estimate is unbiased:

$$\nabla_\theta J(\theta) = \mathbb{E}_b\Bigg[\bar{\rho}_t \nabla_\theta \log \pi_\theta(a_t|s_t)\big(Q^{\text{ret}}(S_t, A_t) - V_\phi(s_t)\big) + \mathbb{E}_{a\sim\pi_\theta}\big[(\rho_t - c)_+ \nabla_\theta \log \pi_\theta(a|s_t)\big(Q_\phi(s_t, a) - V_\phi(s_t)\big)\big]\Bigg]. \tag{2.22}$$

where $(x)_+ = \max(x, 0)$ ensures that only ratios exceeding the truncation threshold contribute to the bias-correction term. The first term performs a truncated policy-gradient update using the Retrace return, while the second term restores unbiasedness by integrating over the tail probability mass beyond c (Wang et al. 2017).

Trust-Region Regularization To further stabilize learning, ACER constrains the Kullback–Leibler (KL) divergence between the current policy π_θ and an exponentially averaged reference policy $\bar{\pi}$:

$$D_{\text{KL}}(\bar{\pi}(\cdot|s) \,\|\, \pi_\theta(\cdot|s)) < \delta, \tag{2.23}$$

where $\delta > 0$ is a small threshold controlling the trust-region size. This prevents abrupt policy shifts between updates and ensures that learning proceeds smoothly within a stable region of the policy space.

2.3 The options framework

In many problems, choosing only primitive actions at every time step can be inefficient. The options framework (Sutton, Precup, and Singh 1999) introduces *temporally-extended actions*—called options—that the agent can initiate, act according the policy for multiple steps, and terminate.

An option $\omega \in \Omega$ is defined as a tuple $\omega = \langle \mathcal{I}_\omega, \pi_\omega, \beta_\omega \rangle$, where $\mathcal{I}_\omega \subseteq \mathcal{S}$ is the initiation set, $\pi_\omega : \mathcal{S} \to \Delta(\mathcal{A})$ is the intra-option policy (the policy of each option), and $\beta_\omega : \mathcal{S} \to [0, 1]$ is the probability termination function that determines the probability the option ω ends upon arrival in a given state. The policy over options is denoted by $\mu(\omega \mid s) : \mathcal{S} \longrightarrow \Delta(\Omega)$ and note that $\mathcal{A} \subset \Omega$. The Kronecker delta is written as $\delta_{\omega,\omega'} = \begin{cases} 1, & \omega = \omega' \\ 0, & \omega \neq \omega' \end{cases}$. Finally, the upon-arrival option probability is given by $\iota_\mu(\omega'|\omega, s') = \beta(s', \omega)\, \mu(\omega' \mid s') + \big(1 - \beta(s', \omega)\big)\, \delta_{\omega,\omega'}$, representing the probability of acting according to option ω' policy when the agent arrives at state s' after following option ω.

2.3.1 Eigenoptions

While the options framework provides a principled formalism for defining temporally extended actions, it does not specify which options should be used. The problem of autonomously constructing useful options is known as *option discovery* (Klissarov et al. 2025). One successful approach for option discovery relies on the *successor representation* (SR) (Dayan 1993), which captures the long-term dynamics of the environment. In particular, *eigenoptions* (Machado, Bellemare, and Bowling 2017; Machado et al. 2023) exploit the spectral structure of the SR to define options that correspond to meaningful modes of diffusion over the state space.

Successor Representation

The SR encodes, for each state, the expected discounted visitation of all other states under a policy π. Formally, given a transition kernel $p(s' \mid s, a)$ and a discount factor $\gamma \in [0, 1)$, the SR with respect to policy π is defined as

$$\Psi_\pi(s, s') = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \, \delta_{S_t, s'} \,\middle|\, S_0 = s \right], \tag{2.24}$$

where $\delta_{S_t,s'}$ is the Kronecker delta between S_t and s'. Each row $\Psi_\pi(s, \cdot)$ represents a vector describing the expected future occupancy of states that follow s when the agent acts according to π.

Definition of Eigenoptions

Given an eigenvector $e \in \mathbb{R}^{|\mathcal{S}|}$ of the SR, one can define an intrinsic reward signal that encourages movement along the direction encoded by e:

$$r_e(s, s') = e^\top \big(\phi(s') - \phi(s) \big), \tag{2.25}$$

where $\phi(s)$ denotes the feature representation of state s, which in the tabular case can be a one-hot vector.

An *eigenoption* $\omega_e = \langle \mathcal{I}_{\omega_e}, \pi_{\omega_e}, \beta_{\omega_e} \rangle$ is then defined through the policy π_{ω_e} that maximizes the expected discounted sum of intrinsic rewards r_e. Each eigenvector yields two eigenoptions: one uses $r_e(s, s')$, and the other uses the opposite signal $-r_e(s, s')$, which drives behavior in the opposite direction. The option's termination condition $\beta_{\omega_e}(s)$ equals 1 when $Q(s, a) \leq 0$ for all actions a. Its initiation set, $\mathcal{I}_{\omega_e}$ is the complement of the termination set (Machado, Bellemare, and Bowling 2017).

The intuition behind eigenoptions is that the SR defines a reachability-based notion of proximity, and its eigenvectors expose the dominant directions of variation in this structure. The intrinsic reward then yields policies that move along these directions, reflecting connectivity rather than external reward.

Eigenoptions capture the principal directions of diffusion in the environment and correspond to temporally-extended behaviors at multiple scales. The eigenvectors associated with larger eigenvalues represent long-term, global modes of movement, while those with smaller eigenvalues correspond to more localized, short-term dynamics. As a result, eigenoptions naturally provide a hierarchy of temporal abstractions that facilitate exploration (Machado et al. 2023). Because they are derived from the SR, which depends only on the environment's transition structure and not on external rewards, eigenoptions are task-agnostic and can be reused across different tasks within the same environment.

ROD cycle

The *Representation-Driven Option Discovery* (ROD) cycle (Machado et al. 2023) formalizes option discovery as an iterative process in which representation learning and temporal abstraction continuously refine each other. In each iteration, the agent interacts with the environment to collect trajectories, learns a representation that captures the environment's dynamics, derives intrinsic reward functions from this representation, and defines new options whose policies maximize these intrinsic rewards. The newly discovered options are incorporated into the agent's behavioral repertoire.

When the ROD cycle is instantiated with *eigenoptions*, the eigenvectors of the successor representation (SR) serve as the learned state embedding from which the agent derives intrinsic rewards. These eigenvectors define intrinsic rewards that encourage movement along their respective directions, giving rise to option policies that traverse the environment's principal structural axes. Once these eigenoptions are learned and added to the option set, the agent can use them to reach previously unexplored regions of the state space.

Chapter 3

The impact of partial observability on eigenoption discovery

Partial observability complicates both representation learning and policy optimization in RL, as the same observation may correspond to different latent states that demand distinct actions.

These challenges have direct implications for eigenoptions. Because eigenoptions are learned from the structure captured by the successor representation, partial observability distorts the SR matrix and, consequently, the spectral decomposition itself. When observations alias multiple underlying states, the SR is learned over the observation, so it no longer faithfully reflects transitions between true latent states. As a result, the intrinsic rewards and option policies derived from its eigenvectors are consistent with the agent's beliefs, but they need not correspond to coherent transitions in the underlying state space, and may induce behaviors that are suboptimal or even contradictory when viewed in terms of the true MDP dynamics.

Partial observability also makes learning eigenoptions policies more difficult. The agent must estimate the sum of intrinsic rewards and termination conditions from trajectories where the true state is hidden. The temporal credit assignment needed to optimize option policies becomes noisy, as the effects of actions can only be inferred indirectly through observation sequences. Consequently, options may fail to converge to meaningful or diverse behaviors, leading to premature termination or redundancy.

In this thesis, we take the observation-based successor representation as given and focus on how to learn effective eigenoption policies from the intrinsic rewards induced by its eigenvectors. Our contributions therefore concern the optimization of option policies under partial observability, rather than the design of representation-learning methods that reconstruct an undistorted eigenstructure.

3.1 Environments

To systematically study the effects of partial observability on eigenoption discovery, we designed five grid-based environments that vary in spatial configuration, number of rooms, and distribution of aliased observations. Each environment consists of a collection of rooms connected through narrow corridors, with a fixed starting position indicated in green, depicted in Figures 3.1 and 3.2. White cells represent fully observable states while colored cells denote aliased states—distinct locations that appear visually identical to the agent. The observation at each location is represented as a one-hot encoding over these perceptual features, so all aliased cells share the same observation vector. These aliased observations introduce perceptual ambiguity, making it impossible for the agent to determine its true position from a single observation.

PO-SIX-ROOMS CENTER and PO-TWO-ROOMS environments focus on inter-room ambiguity. In the first, aliased cells occur symmetrically across multiple rooms, making distinct locations perceptually indistinguishable even though they lead to different successors. Similarly, PO-TWO-ROOMS provides a simpler setting in which partial observability induces spatial ambiguity between two large rooms.

The PO-SIX-ROOMS BOTTLENECK environment was constructed to examine the effect of aliasing at structural bottlenecks—critical states that connect otherwise separated regions. Here, all bottleneck states share the same perceptual representation, forcing the agent to act under ambiguity when deciding which direction to move. Because multiple possible continuations exist from visually identical states, any deterministic policy will fail to consistently reach distant goals. Thus, successful behav-

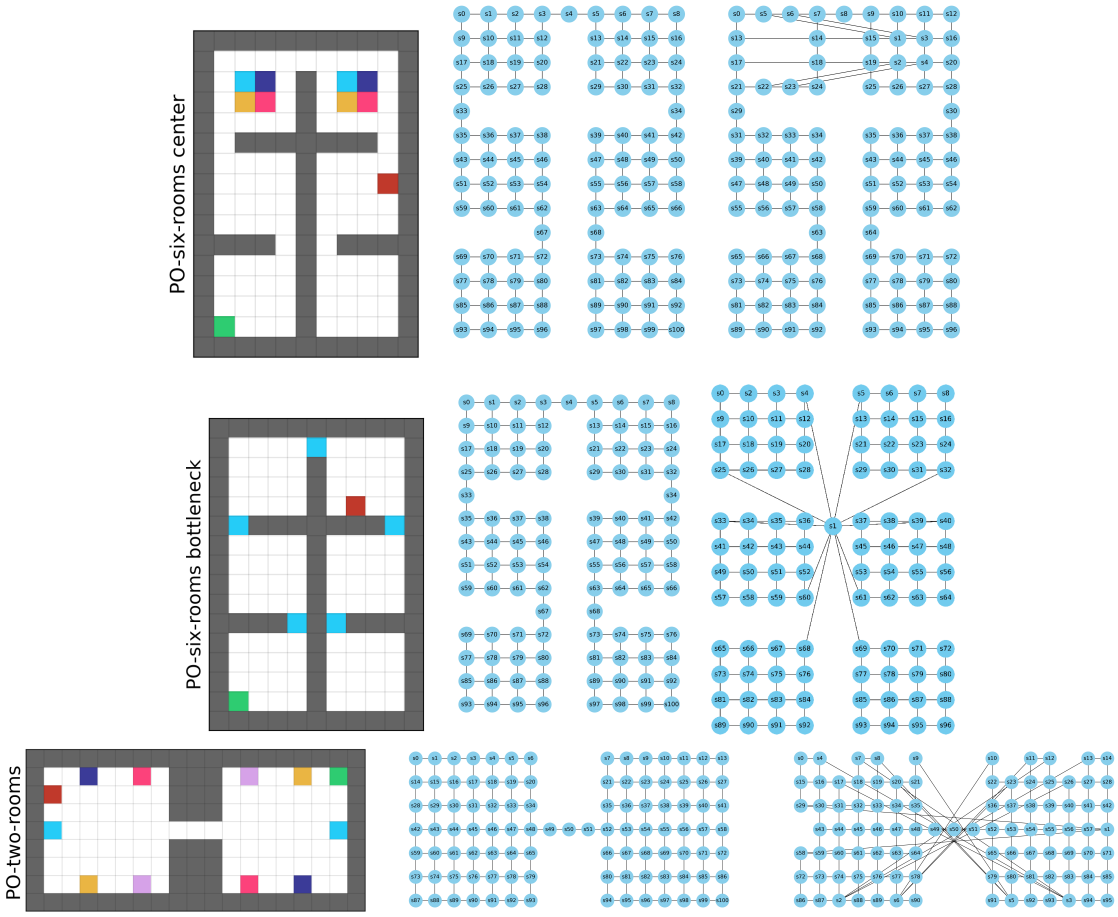


Figure 3.1: Grid layouts and corresponding graph representations for the five environments. For each environment, we display (left) the rendered grid with aliased observation classes indicated by color, (middle) the true MDP transition graph over underlying states, and (right) the perceived POMDP graph derived from the agent's observation model. Aliased states collapse into the same observation and therefore induce a distorted graph structure compared to the true MDP. The light-green cell marks the starting state. The red cell denotes the goal position used only in the reward-maximization experiments in Section 5.4.

ior requires *stochastic policies* that randomize between actions in proportion to the correct long-term outcomes.

Finally, the PO-NINE-ROOMS and PO-SIXTEEN-ROOMS environments in Figure 3.2 are designed to isolate intra-room ambiguity. In these environments, the agent can uniquely determine its position whenever it is adjacent to a wall. Away from the walls, however, it only knows which room it is in, not its exact location within that room.

Figure 3.2: Grid layouts and corresponding graph representations for the second set of environments. Like before, we display (left) the rendered grid with aliased observation classes indicated by color, (middle) the true MDP transition graph over underlying states, and (right) the perceived POMDP graph derived from the agent's observation model. Aliased states collapse into the same observation and therefore induce a distorted graph structure compared to the true MDP. The light-green cell marks the starting state. The red cell denotes the goal position used only in the reward-maximization experiments in Section 5.4.

3.2 Eigenoption Learning

To analyze how partial observability influences the structure of eigenoptions, we visualize in Figure 3.3 the first eigenvector and the first eigenoption—computed with value iteration— for each environment, comparing results obtained with and without partial observability.

Under full observability, the eigenvectors in Figure 3.3 are aligned with the spatial topology of each environment, and their corresponding eigenoptions generate coherent navigation behaviors connecting distant regions. This behavior is characteristic of the diffusion processes captured by the eigenvectors of the successor representation.

Under partial observability, however, the learned eigenvectors become irregular

and locally oscillatory, particularly around aliased or bottleneck regions. The corresponding eigenoptions show cyclic motion patterns or *loops*, repeatedly revisiting some states rather than progressing toward distinct subgoals. These loops are a direct consequence of the *bootstrapping* inherent to value iteration: when multiple aliased states share the same observation, their value estimates become mutually reinforcing, producing feedback cycles in the induced policy field.

This phenomenon is most evident in the environments PO-SIX-ROOMS CENTER and PO-SIX-ROOMS BOTTLENECK, where perceptual ambiguity leads to options with loops. In the latter, oscillatory behavior emerges between the two rooms, while in PO-SIXTEEN-ROOMS and PO-NINE-ROOMS environments, there are loops in multiple rooms. In all cases, partial observability disrupts the global structure that normally aligns eigenoptions with the geometry of the environment.

Further experimental details—including how the transition matrices, successor representations, eigenvectors, and options were computed, as well as the hyperparameters used for value iteration and visualization—are provided in Appendix A.

3.3 Exploration with eigenoptions under partial observability

To evaluate the exploratory value of the discovered eigenoptions, we follow the experimental protocol proposed by Machado et al. (2023), measuring the percentage of unique *states* (rather than observations) visited by the agent over time. The objective is to quantify how effectively the learned options promote coverage of the true state space, independent of external rewards or task-specific objectives.

3.3.1 Eigenoption construction

As detailed in Appendix A, eigenoptions are computed from the successor representation (SR) of a uniform random policy. We obtain the SR analytically using

$$\Psi = (I - \gamma P_\pi)^{-1}, \qquad \gamma = 0.975,$$

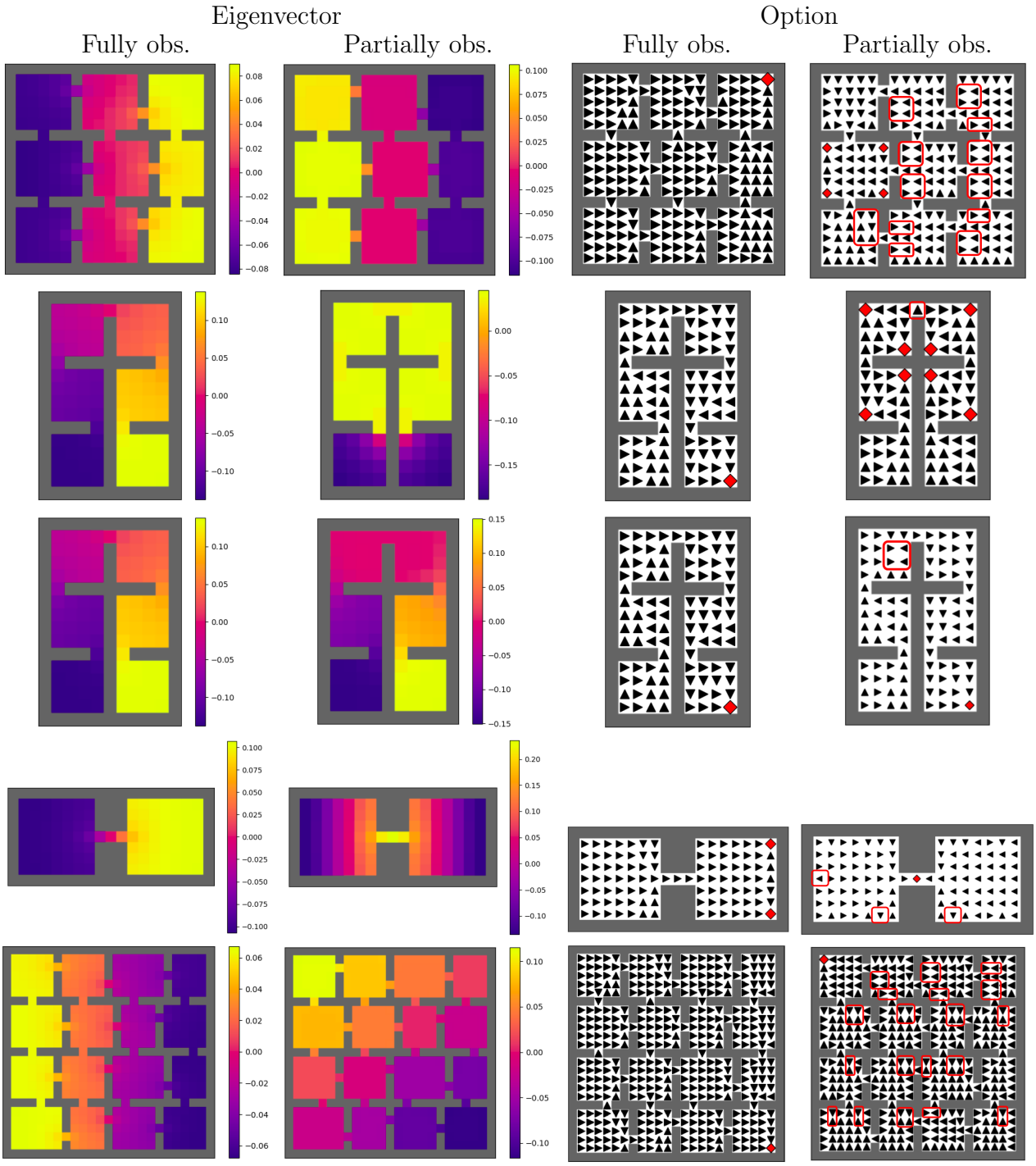


Figure 3.3: First eigenvector and corresponding eigenoption for each environment under full and partial observability. Each row corresponds to one environment; Colors show the value of the first eigenvector component assigned to each underlying grid cell/state, and induced eigenoptions under the two observability settings. The infinite loops are highlighted in red.

where the transition matrix P_π is defined either over states (fully observable) or over observations (partially observable). From each eigenvector we derive intrinsic rewards and compute the corresponding option policy using tabular value iteration. We evaluate sets of 0, 8, and 32 eigenoptions.

3.3.2 Exploration procedure

During exploration, the agent begins at the fixed start state and follows a stochastic mixture policy: with probability 0.95 it executes a random primitive action, and with probability 0.05 it selects one eigenoption uniformly at random and follows it until termination. Each run lasts 50 000 steps with episodes capped at 1 000. Let $\mathcal{S}$ denote the set of all MDP states, and let

$$\mathcal{C}_t = \{\, s \in \mathcal{S} : \exists\, \tau \leq t \text{ such that } S_\tau = s \,\}$$

be the set of distinct states that have been visited at least once up to time t. We define the coverage at time t as

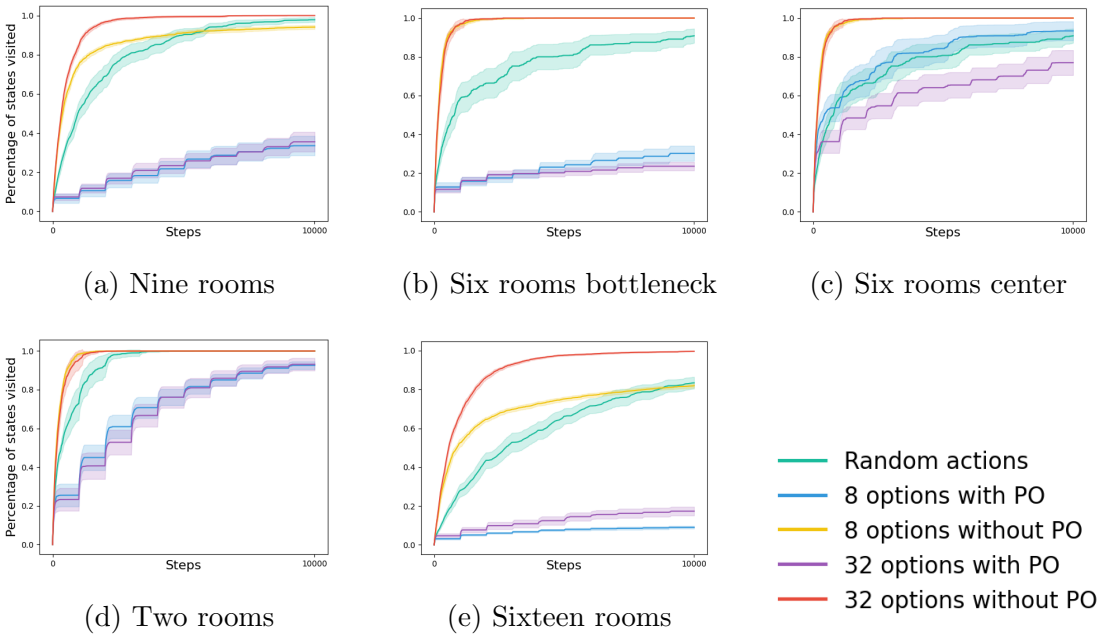
$$\text{Coverage}(t) = \frac{|\mathcal{C}_t|}{|\mathcal{S}|}. \tag{3.1}$$

Each configuration is repeated across 30 seeds, and we report the mean and 95% confidence intervals.

3.3.3 Results

Figure 3.4 shows the diffusion curves for all environments under both full observability and partial observability. Under full observability, eigenoptions significantly accelerate coverage: both 8 and 32-option sets quickly approach complete exploration, clearly outperforming the random baseline.

Under partial observability, however, eigenoptions often *reduce* coverage, even when compared to a uniform random policy over primitive actions. Across all POMDP environments, the curves exhibit a characteristic ladder-like pattern: long flat segments where no new states are visited, followed by sudden increases at episode boundaries.



(a) Nine rooms (b) Six rooms bottleneck (c) Six rooms center

(d) Two rooms (e) Sixteen rooms

Figure 3.4: Exploration performance measured as coverage over 10,000 steps. Shaded regions denote 95% confidence intervals over 30 runs.

This behavior arises because many eigenoptions learned from aliased observations induce cyclic attractors; when such an option is selected (5% of the time), the agent becomes stuck in a loop for the remainder of the episode. As a result, exploration proceeds in discrete bursts rather than continuously.

These findings demonstrate that eigenoptions, while highly effective in fully observable domains, can be detrimental under partial observability when learned directly from aliased observations. This reinforces the need for either memory-based disambiguation mechanisms or modified option-learning algorithms that avoid self-reinforcing cycles.

Chapter 4

Learning options offline with multi-step methods

In this chapter, we study offline multi-step learning from data generated by a hierarchy of options. Our goal is to reuse option-generated trajectories to perform offline, multi-step policy evaluation (and later control) for a *flat* target policy $\pi(a \mid s)$ defined over primitive actions. This setting arises naturally in option discovery objectives, such as in the representation-driven option discovery (ROD) cycle: trajectories are collected by executing existing options, while learning is performed for a different target policy, often under a modified reward signal (e.g., intrinsic rewards for eigenoptions).

We consider an agent executing options $\omega \in \Omega$ with intra-option policies $\{\pi_\omega\}_{\omega\in\Omega}$ and termination functions $\{\beta_\omega\}_{\omega\in\Omega}$, under a policy over options $\mu(\omega \mid s)$. Trajectories in the offline dataset include the executed option identity at each time step, i.e., $(s_t, \omega_t, a_t, r_{t+1}, s_{t+1}, \omega_{t+1}, a_{t+1}, \dots)$. Although the dataset records option identities, our objective is to evaluate or improve a *flat* target policy $\pi(a \mid s)$ without prescribing any target policy over options. To obtain a finite set of distributions suitable for Multiple Importance Sampling (MIS), we define each distribution by conditioning only on the option active at the start of a trajectory suffix:

$$d_\omega(\tau_{t:T}) \doteq \Pr_b(\tau_{t:T} \mid S_t = s,\ \omega_t = \omega),$$

which conditions on ω_t but marginalizes over future options $\omega_{t+1:T-1}$. Under this definition, the option process is treated as latent for $k > t$ when computing $d_\omega(\tau)$: the

primitive behavior becomes a history-dependent mixture over intra-option policies.

Unbiased multi-step off-policy estimators for flat target policies are well known in general (e.g., per-decision importance sampling and its multi-step extensions). The challenge in our setting is that the offline data are generated by the options framework, so the induced primitive behavior is not a single stationary policy $b(a \mid s)$ but a history-dependent mixture that depends on the starting option and on the observed trajectory suffix. To the best of our knowledge, there is no prior derivation that makes this induced primitive behavior likelihood explicit—by marginalizing over future (latent) options while conditioning only on the starting option—in a way that enables applying standard multi-step flat-policy corrections without defining any target policy over options. Existing methods either (i) evaluate or improve a hierarchy (policy over options plus intra-option policies), or (ii) construct targets that bootstrap on option-valued quantities such as $Q(s, \omega)$. In contrast, we seek multi-step corrections yielding targets for $V^\pi(s)$ or $Q^\pi(s, a)$ under a flat target policy π, without introducing any target distribution over future options.

4.1 Related work on off-policy correction for options

Options instantiate distinct policies, π_ω, with their own termination rules, β_ω. When data are collected under the options framework but we wish to evaluate or improve a different policy, *off-policy* corrections are needed to account for the mismatch between the behavior and target option-conditioned policies.

Guo, Thomas, and Brunskill (2017) extended importance sampling to the options framework by adapting per-decision IS to call-and-return trajectories. Their estimator decomposes the mismatch between a behavior hierarchy and an evaluation hierarchy into (i) differences in the policy over options $\mu(\omega \mid s)$, and (ii) differences in the intra-option policies $\pi_\omega(a \mid s)$ versus the intra-option behavior policies $b_\omega(a \mid s)$.

Given a dataset $\mathcal{D} = \{\tau^{(i)}\}_{i=1}^n$ of option-annotated trajectories, let each trajectory

i contain $k^{(i)}$ executed options, and let $j_t^{(i)}$ denote the number of primitive steps taken while option $\omega_t^{(i)}$ is active. Following Guo, Thomas, and Brunskill (2017), we partition the options into

$$\Omega = \{\omega : \pi_\omega = b_\omega\}, \qquad \bar{\Omega} = \Omega^{\text{all}} \setminus \Omega,$$

so that options in Ω require only option-level corrections, whereas options in $\bar{\Omega}$ require both option- and action-level corrections.

The PDIS estimator is

$$\text{PDIS}(\mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{k^{(i)}} W_t^{(i)} \, y_t^{(i)}, \tag{4.1}$$

where the option-level weight is

$$W_t^{(i)} = \prod_{u=1}^{t} \frac{\mu\big(\omega_u^{(i)} \mid S_u^{(i)}\big)}{\mu_b\big(\omega_u^{(i)} \mid S_u^{(i)}\big)}, \tag{4.2}$$

and the per-option contribution is

$$y_t^{(i)} = \begin{cases} v_t^{(i)}, & \omega_t^{(i)} \in \Omega, \\ \displaystyle\sum_{b=1}^{j_t^{(i)}} \rho_{t,b}^{(i)} \, r_{t,b}^{(i)}, & \omega_t^{(i)} \in \bar{\Omega}, \end{cases} \tag{4.3}$$

with the action-level IS ratio

$$\rho_{t,b}^{(i)} = \prod_{c=1}^{b} \frac{\pi_{\omega_t^{(i)}}\big(A_{t,c}^{(i)} \mid S_{t,c}^{(i)}\big)}{b_{\omega_t^{(i)}}\big(A_{t,c}^{(i)} \mid S_{t,c}^{(i)}\big)}. \tag{4.4}$$

This decomposition yields an unbiased off-policy estimator for the evaluation hierarchy when the option identities and boundaries are available in the data.

Jain and Precup (2018) derive option-level importance sampling corrections that permit the use of eligibility traces. For any candidate option $\omega \in \Omega$ that could have been active at s_t, they correct the backup with the ratio of upon-arrival option probabilities:

$$\rho_{t+1}(\omega \leftarrow \omega_t) \;=\; \frac{\iota(\omega_{t+1} \mid \omega, s_{t+1})}{\iota(\omega_{t+1} \mid \omega_t, s_{t+1})}, \tag{4.5}$$

where we use the notation $o \leftarrow \omega_t$ to emphasize that we re-interpret the same trajectory as if option o had been active at time t. Because a_{t+1} is sampled from

$\pi_{\omega_{t+1}}(\cdot \mid s_{t+1})$ in both numerator and denominator, intra-option action terms cancel. Chaining these ratios yields n-step off-policy returns for $Q(s, \omega)$, enabling consistent intra-option learning for multiple options from a single call-and-return trajectory while respecting the option terminations β_ω and the policy-over-options μ. In contrast to PDIS, which focuses on evaluating a particular hierarchy, this line of work explicitly leverages the option structure to share data across different options.

Alternatively, Klissarov and Precup (2021) propose targets that correct for discrepancies between the executing option ω_t and an arbitrary option ω being evaluated. Using standard importance sampling over the intra-option policies, their one-step target for $Q(s_t, \omega)$ is

$$U_t^\rho(s_t, \omega) \;=\; \underbrace{\frac{\pi_\omega(a_t \mid s_t)}{\pi_{\omega_t}(a_t \mid s_t)}}_{\text{action IS}} r_{t+1} + \gamma \; \underbrace{\frac{\iota(\omega_{t+1} \mid \omega, s_{t+1})}{\iota(\omega_{t+1} \mid \omega_t, s_{t+1})}}_{\text{option upon-arrival IS}} \hat{Q}(s_{t+1}, \omega_{t+1}), \tag{4.6}$$

where $\hat{Q}$ denotes the critic. This target simultaneously corrects (i) the intra-option action mismatch between π_{ω_t} and π_ω, and (ii) the hierarchical option-selection mismatch induced by β_ω and μ. Their analysis shows how to construct flexible off-policy targets for options while still relying on call-and-return trajectories annotated with the executing option.

These methods are designed for option-annotated trajectories, and they provide principled estimators for option-valued quantities such as $Q(s, \omega)$ or for updating a policy over options from call-and-return data. They all share a similar structure: for each target option ω, they construct importance ratios that compare the probability of the observed trajectory under the behavior hierarchy with the probability of a counterfactual trajectory in which the same primitive sequence is interpreted as having been generated by ω.

Taking the construction of Jain and Precup (2018) at a single time step t as an example, the call-and-return process induces two trajectory distributions: one for the actual trajectory, $(s_t, \omega_t, a_t, r_{t+1}, s_{t+1}, \omega_{t+1}, a_{t+1}, r_{t+2}, \ldots, s_T)$, and one for the coun-

terfactual trajectory in which the same primitive sequence is reinterpreted as having started from option ω, $(s_t, \omega, a_t, r_{t+1}, s_{t+1}, \omega_{t+1}, a_{t+1}, r_{t+2}, \ldots, s_T)$. Writing $\Pr_{\mu,\beta}$ for the trajectory probability induced by the hierarchy (μ, Ω, β), the option-level ratio can be expressed as

$$\rho_{t+1}(\omega \leftarrow \omega_t) = \frac{\Pr_{\mu,\beta}\big(r_{t+1}, s_{t+1}, \omega_{t+1}, a_{t+1}, r_{t+2}, \ldots, s_T | s_t, \omega, a_t\big)}{\Pr_{\mu,\beta}\big(r_{t+1}, s_{t+1}, \omega_{t+1}, a_{t+1}, r_{t+2}, \ldots, s_T | s_t, \omega_t, a_t\big)}$$

$$= \frac{Pr_{\mu,\beta}(s_{t+1} \mid s_t, a_t)\, \iota(\omega_{t+1} \mid \omega, s_{t+1}) \prod_{k=t+1}^{T-1} \pi_{\omega_k}(a_k \mid s_k)\, Pr_{\mu,\beta}(s_{k+1} \mid s_k, a_k)\, \iota(\omega_{k+1} \mid \omega_k, s_{k+1})}{Pr_{\mu,\beta}(s_{t+1} \mid s_t, a_t)\, \iota(\omega_{t+1} \mid \omega_t, s_{t+1}) \prod_{k=t+1}^{T-1} \pi_{\omega_k}(a_k \mid s_k)\, Pr_{\mu,\beta}(s_{k+1} \mid s_k, a_k)\, \iota(\omega_{k+1} \mid \omega_k, s_{k+1})}$$

$$= \frac{\iota(\omega_{t+1} \mid \omega, s_{t+1})}{\iota(\omega_{t+1} \mid \omega_t, s_{t+1})},$$

and following the same derivation for two-steps

$$\rho_{t+1}(\omega \leftarrow \omega_t) = \frac{\iota(\omega_{t+1} \mid \omega, s_{t+1})\iota(\omega_{t+2} \mid \omega, s_{t+2})}{\iota(\omega_{t+1} \mid \omega_t, s_{t+1})\iota(\omega_{t+2} \mid \omega_{t+1}, s_{t+2})}.$$

These methods provide principled estimators for option-valued quantities (e.g., $Q(s, \omega)$) or for evaluating a specified hierarchy. They do not directly address the offline option-discovery setting in which the target policy is flat over primitive actions and we explicitly avoid committing to any target policy over options. In particular, we seek targets that do not bootstrap on values of future options and do not depend on target option-selection probabilities.

Alternatively, if one tries to apply standard IS directly, two issues arise. First, there is no single stationary primitive behavior policy $b(a \mid s)$: conditioned only on the starting option, the primitive behavior at time k is history-dependent:

$$\Pr_b(A_k = a \mid H_k^{\text{pre}},\ \omega_t) = b_{\omega_t}(a \mid H_k^{\text{pre}}) = \sum_{\omega \in \Omega} p_{\mu,\beta}^{(\omega_t)}(\omega \mid H_k^{\text{pre}})\, \pi_\omega(a \mid S_k),$$

where the mixing weights $p_{\mu,\beta}^{(\omega_t)}(\omega \mid H_k^{\text{pre}})$ depend on the observed history H_k^{pre} through the latent option process. Second, existing option-IS methods typically define corrections relative to the options framework and produce targets that involve option-valued bootstrapping; in our setting, the target does not select options at all.

These limitations motivate the algorithm developed in this chapter. Our key idea is to treat each starting option as a separate behavior component and view option-generated data as samples from a finite family of distributions $\{d_\omega\}_{\omega\in\Omega}$. We then apply MIS to combine them into unbiased estimators for flat-policy evaluation, and later extend the approach with Retrace-style truncation for stability.

4.2 Multiple importance sampling for options

Throughout, $\tau_{t:T}$ denotes a primitive trajectory suffix from time t until a terminal index T. In episodic tasks, T is the end of the episode. More generally, if offline logs are truncated, T denotes the truncation point and the return functional $G_t(\tau_{t:T})$ is defined accordingly (e.g., a truncated Monte Carlo return, or a bootstrapped return). Importantly, we do not assume the starting option ω_t remains active until T; options may terminate and be resampled under call-and-return dynamics.

4.2.1 Option-induced primitive behavior as a history-dependent mixture

Consider a hierarchy (μ, Ω, β) and a time step t at which option ω_t is active. Conditioning on (S_t, ω_t) induces primitive trajectory suffixes

$$\tau_{t:T} = (S_t, A_t, S_{t+1}, A_{t+1}, \ldots, S_T).$$

To keep track of the information available at each decision, we distinguish between two types of history at time $k \geq t$:

$$H_k^{\text{pre}} = (S_t, A_t, \ldots, S_{k-1}, A_{k-1}, S_k), \qquad \text{(history } \textit{before} \text{ drawing } A_k\text{)}, \tag{4.7}$$

$$H_k^{\text{post}} = (S_t, A_t, \ldots, S_{k-1}, A_{k-1}, S_k, A_k), \qquad \text{(history } \textit{after} \text{ observing } A_k\text{)}. \tag{4.8}$$

Because we condition only on the starting option, the currently active option at time $k > t$ is a latent variable. Let

$$p_k^{\text{pre}}(\omega) \doteq p_{\mu,\beta}^{(\omega_t)}(\omega_k = \omega \mid H_k^{\text{pre}})$$

denote the belief over the active option upon reaching S_k. Then the induced primitive behavior at step k is the history-dependent mixture

$$b_{\omega_t}(A_k \mid H_k^{\text{pre}}) = \sum_{\omega \in \Omega} p_k^{\text{pre}}(\omega)\, \pi_\omega(A_k \mid S_k). \tag{4.9}$$

After observing A_k, we update the belief using Bayes' rule:

$$p_k^{\text{post}}(\omega) \;\doteq\; p_{\mu,\beta}^{(\omega_t)}(\omega_k = \omega \mid H_k^{\text{post}}) = \frac{p_k^{\text{pre}}(\omega)\, \pi_\omega(A_k \mid S_k)}{\sum_{\bar{\omega} \in \Omega} p_k^{\text{pre}}(\bar{\omega})\, \pi_{\bar{\omega}}(A_k \mid S_k)} = \frac{p_k^{\text{pre}}(\omega)\, \pi_\omega(A_k \mid S_k)}{b_{\omega_t}(A_k \mid H_k^{\text{pre}})}. \tag{4.10}$$

Under the standard options execution semantics (SMDP model), the option process evolves upon arrival in S_{k+1} according to the induced kernel

$$\iota(\omega_{k+1} \mid \omega_k, S_{k+1}) = \beta_{\omega_k}(S_{k+1})\, \mu(\omega_{k+1} \mid S_{k+1}) + \big(1 - \beta_{\omega_k}(S_{k+1})\big)\, \delta_{\omega_{k+1},\omega_k}. \tag{4.11}$$

Therefore, the next pre-action belief is obtained by marginalizing over ω_k:

$$p_{k+1}^{\text{pre}}(\omega') = \sum_{\omega \in \Omega} p_k^{\text{post}}(\omega)\, \iota(\omega' \mid \omega, S_{k+1}). \tag{4.12}$$

The initialization at time t is $p_t^{\text{pre}}(\omega) = \delta_{\omega,\omega_t}$.

Assumption 1. *[Per-option coverage] For every option $\omega \in \Omega$ and every state s in which ω can be executed, the intra-option policy assigns non-zero probability to every action:*

$$\pi_\omega(a \mid s) > 0 \quad \textit{for all } a \in \mathcal{A}.$$

4.2.2 Trajectory-wise likelihood ratio with posterior over options

We first identify the likelihood ratio between the option-induced behavior and the flat target policy at the level of entire trajectory suffixes, explicitly accounting for the latent option process via the posterior $p_{\mu,\beta}^{(\omega_t)}(\omega_k \mid H_k^{\text{post } A_k})$.

Lemma 1 (Option-induced likelihood ratio) *Fix a starting state $S_t = s$ and starting option ω_t. Let $\Pr_\pi(\tau_{t:T} \mid S_t = s)$ denote the probability of a primitive trajectory suffix $\tau_{t:T}$ under the flat target policy $\pi(a \mid s)$. Let $\Pr_b(\tau_{t:T} \mid S_t = s, \omega_t)$*

denote the marginal *probability of the same primitive suffix under the options behavior, conditioned only on the starting option ω_t and marginalizing over future options $\omega_{t+1:T-1}$. Equivalently, $\Pr_b$ is induced by the history-dependent primitive behavior policy $b_{\omega_t}(\cdot \mid H_k^{\text{pre}})$ defined in* (4.9)*. Assuming identical environment dynamics, define*

$$\rho_{t:T-1} \doteq \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b_{\omega_t}(A_k \mid H_k^{\text{pre}})}. \tag{4.13}$$

Then

$$\rho_{t:T-1} = \frac{\Pr_\pi(\tau_{t:T} \mid S_t = s)}{\Pr_b(\tau_{t:T} \mid S_t = s, \omega_t)}.$$

Proof. Fix a trajectory suffix $\tau_{t:T} = (S_t, A_t, S_{t+1}, A_{t+1}, \ldots, S_T)$. Under the flat target policy π, the environment is Markov and the dynamics do not depend on the policy once actions are fixed. Therefore, by the chain rule,

$$\Pr_\pi(\tau_{t:T} \mid S_t) = \prod_{k=t}^{T-1} \Pr_\pi(A_k, S_{k+1} \mid H_k^{\text{pre A}_k}) = \prod_{k=t}^{T-1} \Pr_\pi(A_k \mid H_k^{\text{pre A}_k}) \Pr(S_{k+1} \mid S_k, A_k), \tag{4.14}$$

where $H_k^{\text{pre A}_k} = (S_t, A_t, \ldots, S_{k-1}, A_{k-1}, S_k)$. Since under π actions depend only on the current state, $\Pr_\pi(A_k \mid H_k^{\text{pre A}_k}) = \pi(A_k \mid S_k)$, and thus

$$\Pr_\pi(\tau_{t:T} \mid S_t) = \prod_{k=t}^{T-1} \pi(A_k \mid S_k) \Pr(S_{k+1} \mid S_k, A_k). \tag{4.15}$$

Under the option-induced behavior, we condition on (S_t, ω_t) and again apply the chain rule:

$$\Pr_b(\tau_{t:T} \mid S_t, \omega_t) = \prod_{k=t}^{T-1} \Pr_b(A_k, S_{k+1} \mid H_k^{\text{pre A}_k}, \omega_t). \tag{4.16}$$

By definition of $H_k^{\text{pre A}_k}$ and the environment dynamics,

$$\Pr_b(A_k, S_{k+1} \mid H_k^{\text{pre A}_k}, \omega_t) = \Pr_b(A_k \mid H_k^{\text{pre A}_k}, \omega_t) \Pr(S_{k+1} \mid S_k, A_k). \tag{4.17}$$

We now identify the behavior policy at step k with the mixture in (4.9), that is,

$$\Pr_b(A_k \mid H_k^{\text{pre A}_k}, \omega_t) = b_{\omega_t}(A_k \mid H_k^{\text{pre A}_k}). \tag{4.18}$$

Substituting this into the expression above yields

$$\Pr_b(\tau_{t:T} \mid S_t, \omega_t) = \prod_{k=t}^{T-1} b_{\omega_t}(A_k \mid H_k^{\text{pre A}_k}) \Pr(S_{k+1} \mid S_k, A_k). \tag{4.19}$$

Taking the ratio of (4.15) and (4.19) gives

$$\frac{\Pr_\pi(\tau_{t:T} \mid S_t)}{\Pr_b(\tau_{t:T} \mid S_t, \omega_t)} = \frac{\prod_{k=t}^{T-1} \pi(A_k \mid S_k) \Pr(S_{k+1} \mid S_k, A_k)}{\prod_{k=t}^{T-1} b_{\omega_t}(A_k \mid H_k^{\text{pre A}_k}) \Pr(S_{k+1} \mid S_k, A_k)} \tag{4.20}$$

$$= \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b_{\omega_t}(A_k \mid H_k^{\text{pre A}_k})}. \tag{4.21}$$

All environment dynamics $\Pr(S_{k+1} \mid S_k, A_k)$ cancel, as they are identical under π and under the option-induced behavior. The remaining product is precisely the definition of $\rho_{t:T-1}$ in (4.13), so

$$\frac{\Pr_\pi(\tau_{t:T} \mid S_t)}{\Pr_b(\tau_{t:T} \mid S_t, \omega_t)} = \rho_{t:T-1}. \tag{4.22}$$

Assumption 1 ensures that $b_{\omega_t}(A_k \mid H_k^{\text{pre A}_k}) > 0$ whenever $\pi(A_k \mid S_k) > 0$, so each ratio is well defined. ■

If one conditions on the full logged option sequence $\omega_{t:T-1}$, the primitive behavior likelihood becomes $\prod_{k=t}^{T-1} \pi_{\omega_k}(A_k \mid S_k)$, yielding the ratio $\prod_{k=t}^{T-1} \pi(A_k \mid S_k)/\pi_{\omega_k}(A_k \mid S_k)$. We do not adopt this conditional likelihood because it would index distributions by option paths rather than by just the starting option.

4.2.3 From single options to MIS over options

Lemma 1 gives, for a *fixed* starting option ω_t, a trajectory-wise likelihood ratio between the option-induced primitive behavior and a flat target policy $\pi(a \mid s)$. In an offline dataset, however, trajectory suffixes starting from the same state s may have been generated under different starting options. This fits the Multiple Importance Sampling (MIS) viewpoint: each starting option defines a distinct *behavior component* over trajectory suffixes, and we combine these components into an unbiased estimator for a target expectation under π.

Trajectory distributions. Fix a time index t and a state s, and consider the space of primitive trajectory suffixes

$$\tau \equiv \tau_{t:T} = (S_t, A_t, S_{t+1}, A_{t+1}, \ldots, S_T) \qquad \text{with } S_t = s.$$

Let the *target* distribution over suffixes be

$$d_\pi(\tau) \doteq \Pr_\pi(\tau_{t:T} = \tau \mid S_t = s),$$

i.e., the probability of observing τ when primitive actions are sampled from the flat target policy $\pi(\cdot \mid \cdot)$.

For each starting option $\omega \in \Omega$, define the corresponding *option-conditioned behavior distribution*

$$d_\omega(\tau) \doteq \Pr_b(\tau_{t:T} = \tau \mid S_t = s,\, \omega_t = \omega),$$

i.e., the probability of observing τ when the active option at time t is ω and subsequent behavior follows the standard options execution semantics induced by (μ, Ω, β).

Our goal is to estimate the target value

$$V^\pi(s) \doteq \mathbb{E}_{\tau \sim d_\pi}[G_t(\tau)]\,,$$

where $G_t(\tau)$ is any return functional of the suffix τ.

Likelihood ratios for each behavior component. Under Assumption 1, the support of each d_ω covers the support of d_π, so the likelihood ratio $d_\pi(\tau)/d_\omega(\tau)$ is well-defined whenever $d_\pi(\tau) > 0$.

Moreover, by Lemma 1, for any suffix τ generated from the behavior component indexed by ω we can write

$$\frac{d_\pi(\tau)}{d_\omega(\tau)} = \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b_\omega(A_k \mid H_k^{\text{pre}})}, \tag{4.23}$$

where $b_\omega(\cdot \mid H_k^{\text{pre}})$ is the option-induced primitive behavior mixture at step k, computed conditional on $\omega_t = \omega$ as in (4.9).

MIS estimator across starting options. Suppose the offline dataset contains, for each starting option ω, a collection of N_ω i.i.d. suffixes drawn from d_ω:

$$\tau^{(\omega,1)}, \ldots, \tau^{(\omega,N_\omega)} \overset{\text{i.i.d.}}{\sim} d_\omega(\cdot).$$

Let $\eta_\omega(\tau)$ be MIS weights satisfying

$$\sum_{\omega\in\Omega} \eta_\omega(\tau) = 1 \quad \text{whenever } d_\pi(\tau) > 0, \qquad d_\omega(\tau) = 0 \Rightarrow \eta_\omega(\tau) = 0.$$

Define the MIS estimator

$$\widehat{V}(s) = \sum_{\omega\in\Omega} \frac{1}{N_\omega} \sum_{i=1}^{N_\omega} \eta_\omega\big(\tau^{(\omega,i)}\big)\, G_t\big(\tau^{(\omega,i)}\big)\, \frac{d_\pi(\tau^{(\omega,i)})}{d_\omega(\tau^{(\omega,i)})}. \tag{4.24}$$

Theorem 1 (Unbiased MIS estimator over option-conditioned behavior components) *Under Assumption 1 and the MIS conditions above,*

$$\mathbb{E}_{\mathcal{D}}\Big[\widehat{V}(s)\Big] = \mathbb{E}_{\tau\sim d_\pi}[G_t(\tau)] = V^\pi(s),$$

where the outer expectation is taken over the random dataset $\mathcal{D} = \{\tau^{(\omega,i)}\}$, with $\tau^{(\omega,i)}$ i.i.d. from d_ω for each $\omega \in \Omega$ (and independent across ω).

Proof. For each ω, define

$$f_\omega(\tau) \;\doteq\; \eta_\omega(\tau)\, G_t(\tau)\, \frac{d_\pi(\tau)}{d_\omega(\tau)}.$$

By linearity of expectation and i.i.d. sampling within each option-indexed subset,

$$\mathbb{E}_{\mathcal{D}}\Big[\widehat{V}(s)\Big] = \sum_{\omega\in\Omega} \mathbb{E}\left[\frac{1}{N_\omega}\sum_{i=1}^{N_\omega} f_\omega(\tau^{(\omega,i)})\right] = \sum_{\omega\in\Omega} \mathbb{E}_{\tau\sim d_\omega}[f_\omega(\tau)]\,.$$

Expanding each expectation (discrete finite-horizon case),

$$\mathbb{E}_{\tau\sim d_\omega}[f_\omega(\tau)] = \sum_\tau d_\omega(\tau)\, f_\omega(\tau) = \sum_\tau \eta_\omega(\tau)\, G_t(\tau)\, d_\pi(\tau),$$

where Assumption 1 ensures the ratio is well-defined on the support of d_π. Summing over ω yields

$$\mathbb{E}_{\mathcal{D}}\Big[\widehat{V}(s)\Big] = \sum_\tau \left(\sum_{\omega\in\Omega} \eta_\omega(\tau)\right) G_t(\tau)\, d_\pi(\tau) = \sum_\tau G_t(\tau)\, d_\pi(\tau) = \mathbb{E}_{\tau\sim d_\pi}[G_t(\tau)].$$

■

Practical choice of MIS weights. Theorem 1 holds for any $\{\eta_\omega\}$ satisfying the MIS conditions. In this thesis we adopt the simple, option-independent choice

$$\eta_\omega(\tau) \equiv \eta_\omega(s) = \frac{1}{|\Omega|} \qquad \text{for all } \omega \in \Omega,$$

avoiding estimating option-occupancy probabilities while satisfying $\sum_\omega \eta_\omega(\tau) = 1$.

Resulting stochastic-approximation update. Using the likelihood ratio from Lemma 1, we obtain the practical update

$$V(S_t) \leftarrow V(S_t) + \alpha\, \rho_{t:T-1} \big(G_t - V(S_t)\big), \tag{4.25}$$

where $\rho_{t:T-1} = d_\pi(\tau_{t:T})/d_{\omega_t}(\tau_{t:T})$ is evaluated on the observed suffix as in (4.23). The constant factor $1/|\Omega|$, is absorbed into the step-size α. This update is the core building block for Algorithm 1, which we state next.

4.3 MIS-Retrace: Multiple-Importance Retrace under option-induced behavior

Theorem 1 and Lemma 1 provide a trajectory-wise likelihood-ratio correction from option-induced primitive behavior to a flat target policy $\pi(a \mid s)$. In practice, using the full product of importance ratios over long horizons can lead to high variance and unstable learning when combined with bootstrapping. To obtain a stable multi-step target for critic learning, we adapt the truncated multi-step scheme Retrace(λ) (Munos et al. 2016) to the option-induced behavior setting.

We consider offline learning of an action-value function $Q(s, a)$ using data generated by executing options. The target policy π is a *primitive* policy (it does not select options). MIS-Retrace is used to construct a low-variance multi-step target for Q under π that can be employed inside an actor–critic method (e.g., ACER), where stability is prioritized over exact unbiasedness.

Algorithm 1 Offline value function prediction using MIS over options

Require: Dataset $\mathcal{D} = \{\tau^{(1)}, \ldots, \tau^{(N)}\}$ of N episodes
Require: Target policy π
Require: Intra-option policies $\{\pi_\omega\}_{\omega \in \Omega}$
Require: Upon-arrival kernel $\iota(\omega' \mid \omega, s')$
Require: Step-size $\alpha > 0$, discount factor $\gamma \in [0, 1)$
Initialize value estimates $V(\cdot)$ arbitrarily
for each trajectory $\tau = (s_0, \omega_0, a_0, r_1, s_1, \omega_1, a_1, r_2, \ldots, s_T)$ in $\mathcal{D}$ **do**
    for $t = 0$ to $T - 1$ **do**
        $G \leftarrow 0$, $\rho \leftarrow 1$
        $p^{\mathrm{pre}}(\omega) \leftarrow \delta_{\omega, \omega_t} \quad \forall\, \omega \in \Omega$
        for $k = t$ to $T - 1$ **do**
            $b \leftarrow \sum_{\omega \in \Omega} p^{\mathrm{pre}}(\omega)\, \pi_\omega(a_k \mid s_k)$
            $\rho \leftarrow \rho \cdot \dfrac{\pi(a_k \mid s_k)}{b}$
            $G \leftarrow G + \gamma^{k-t}\, r_{k+1}$
            if $k < T - 1$ **then**
                $p^{\mathrm{post}}(\omega) \leftarrow \dfrac{p^{\mathrm{pre}}(\omega)\, \pi_\omega(a_k \mid s_k)}{b} \quad \forall\, \omega \in \Omega$
                $p^{\mathrm{pre}}(\cdot) \leftarrow 0$
                for $\omega' \in \Omega$ **do**
                    for $\omega \in \Omega$ **do**
                        $p^{\mathrm{pre}}(\omega') \mathrel{+}= p^{\mathrm{post}}(\omega)\, \iota(\omega' \mid \omega, s_{k+1})$
                    end for
                end for
            end if
        end for
        $V(s_t) \leftarrow V(s_t) + \alpha\, \rho\, (G - V(s_t))$
    end for
end for

4.3.1 Exact ratios versus approximate ratios

For a fixed starting option label ω_t (observed in the data at time t), the exact per-step importance ratio implied by Lemma 1 is

$$\rho_k \doteq \frac{\pi(A_k \mid S_k)}{b_{\omega_t}(A_k \mid H_k^{\text{pre}})}, \qquad b_{\omega_t}(A_k \mid H_k^{\text{pre}}) = \sum_{\omega \in \Omega} p_k^{\text{pre}}(\omega)\, \pi_\omega(A_k \mid S_k),$$

where the belief $p_k^{\text{pre}}(\omega) = \Pr(\omega_k = \omega \mid H_k^{\text{pre}}, \omega_t)$ is obtained by filtering via (4.10)–(4.12).

While this yields an exact likelihood ratio, conditioning the belief on every observed action (full filtering) can produce highly variable $b_{\omega_t}(A_k \mid H_k^{\text{pre}})$ and therefore highly variable ratios. We therefore adopt a lower-variance approximation that propagates the belief using only the induced option-transition kernel ι, without conditioning on actions.

Concretely, we define an approximate pre-action belief $\tilde{p}_k^{\text{pre}}$ by

$$\tilde{p}_{k+1}^{\text{pre}}(\omega') = \sum_{\omega \in \Omega} \tilde{p}_k^{\text{pre}}(\omega)\, \iota(\omega' \mid \omega, S_{k+1}), \qquad \tilde{p}_t^{\text{pre}}(\omega) = \delta_{\omega,\omega_t}. \tag{4.26}$$

This induces an approximate mixture behavior

$$\tilde{b}_{\omega_t}(A_k \mid \tilde{H}_k) \doteq \sum_{\omega \in \Omega} \tilde{p}_k^{\text{pre}}(\omega)\, \pi_\omega(A_k \mid S_k),$$

and approximate ratios

$$\tilde{\rho}_k \doteq \frac{\pi(A_k \mid S_k)}{\tilde{b}_{\omega_t}(A_k \mid \tilde{H}_k)}. \tag{4.27}$$

Because (4.26) ignores action conditioning, in general $\tilde{\rho}_k \neq \rho_k$, and the resulting correction is no longer an exact likelihood ratio. MIS-Retrace should therefore be viewed as a practical bias–variance trade-off, analogous to other truncated/approximate IS methods used for stable multi-step bootstrapping.

4.3.2 Retrace truncation with approximate MIS ratios

Following Retrace(λ) (Munos et al. 2016), we define truncation coefficients

$$c_k \doteq \lambda \min\big(1, \tilde{\rho}_k\big), \qquad \lambda \in [0, 1].$$

Truncation prevents large products of ratios from dominating the update, substantially reducing variance at the cost of bias.

Given a trajectory segment $(S_t, A_t, R_{t+1}, S_{t+1}, \ldots, S_T)$, define

$$V(S_{k+1}) \doteq \sum_a \pi(a \mid S_{k+1})\, Q(S_{k+1}, a),$$

with the convention $V(S_T) = 0$ at terminal states (episodic tasks).

The MIS-Retrace target for $Q(S_t, A_t)$ is

$$Q_t^{\text{MIS-Ret}} = Q(S_t, A_t) + \sum_{k=t}^{T-1} \gamma^{k-t} \left(\prod_{i=t+1}^{k} c_i \right) \delta_k^\pi, \tag{4.28}$$

where the TD-error under the target policy is

$$\delta_k^\pi \;=\; \tilde{\rho}_k \big(R_{k+1} + \gamma V(S_{k+1}) - Q(S_k, A_k) \big).$$

Equation (4.28) admits the standard backward recursion:

$$Q_k^{\text{ret}} = Q(S_k, A_k) + \delta_k^\pi + \gamma\, c_{k+1} \Big(Q_{k+1}^{\text{ret}} - Q(S_{k+1}, A_{k+1}) \Big),$$

with terminal condition $Q_T^{\text{ret}} = Q(S_T, A_T)$ when applicable. We denote by $Q_t^{\text{MIS-Ret}}$ the value returned by this recursion at time t. Algorithm 2 implements this approximation and is evaluated in the next chapter when learning eigenoptions using ACER.

4.4 Results

In this section we empirically evaluate the estimator in Equation 4.25 and Algorithm 1. We compare it against on-policy Monte Carlo estimates and against naive wrong estimators to evaluate how it behaves in practice. We then study, in a K-armed bandit, how different coverage profiles impact our algorithm's performance and fixed point. We evaluate MIS-retrace on the next chapter.

4.4.1 Comparison against naive estimators

We study a small MDP summarized in Figure 4.1 to compare our estimator against naive importance sampling baselines. The environment has eight non-terminal states

Algorithm 2 Offline Q-function estimation using MIS-Retrace over options

Require: Dataset $\mathcal{D}$
Require: Target policy π (define $V(s) = \sum_a \pi(a \mid s)\, Q(s,a)$)
Require: Intra-option policies $\{\pi_\omega\}_{\omega \in \Omega}$
Require: Induced option-transition kernel $\iota(\omega' \mid \omega, s')$
Require: Step-size α, discount γ, trace parameter λ

for each trajectory $\tau = (s_0, \omega_0, a_0, r_1, s_1, \omega_1, a_1, r_2, \ldots, s_T)$ in $\mathcal{D}$ **do**
  $L \leftarrow T$ ▷ *number of transitions*
  Initialize arrays $\tilde{b}[0{:}L-1]$, $\tilde{\rho}[0{:}L-1]$, $c[0{:}L-1]$
  Initialize belief $\tilde{p}_0^{\mathrm{pre}}(\omega) \leftarrow \delta_{\omega,\omega_0}$
  ▷ *Forward pass: compute $\tilde{b}_k, \tilde{\rho}_k, c_k$ using prior-only propagation*
  for $k = 0$ to $L-1$ **do**
    $\tilde{b}[k] \leftarrow \sum_{\omega \in \Omega} \tilde{p}_k^{\mathrm{pre}}(\omega)\, \pi_\omega(a_k \mid s_k)$
    $\tilde{\rho}[k] \leftarrow \dfrac{\pi(a_k \mid s_k)}{\tilde{b}[k]}$
    $c[k] \leftarrow \lambda \min\big(1, \tilde{\rho}[k]\big)$
    if $k < L-1$ **then**
      $\tilde{p}_{k+1}^{\mathrm{pre}}(\cdot) \leftarrow 0$
      for $\omega' \in \Omega$ **do**
        for $\omega \in \Omega$ **do**
          $\tilde{p}_{k+1}^{\mathrm{pre}}(\omega') \mathrel{+}= \tilde{p}_k^{\mathrm{pre}}(\omega)\, \iota(\omega' \mid \omega, s_{k+1})$
        end for
      end for
    end if
  end for
  $Q^{\mathrm{ret}} \leftarrow 0$
  ▷ *Backward pass: Retrace recursion with $\tilde{\rho}$ and* c
  for $k = L-1$ down to 0 **do**
    $s \leftarrow s_k$, $a \leftarrow a_k$, $r \leftarrow r_{k+1}$
    if $k = L-1$ **then**
      $V_{k+1} \leftarrow 0$ ▷ *terminal bootstrap (episodic)*
    else
      $V_{k+1} \leftarrow \sum_{a'} \pi(a' \mid s_{k+1})\, Q(s_{k+1}, a')$
    end if
    $\delta_k^\pi \leftarrow \tilde{\rho}[k]\big(r + \gamma V_{k+1} - Q(s,a)\big)$
    if $k = L-1$ **then**
      $Q^{\mathrm{ret}} \leftarrow Q(s,a) + \delta_k^\pi$
    else
      $Q^{\mathrm{ret}} \leftarrow Q(s,a) + \delta_k^\pi + \gamma\, c[k+1]\big(Q^{\mathrm{ret}} - Q(s_{k+1}, a_{k+1})\big)$
    end if
    $Q(s,a) \leftarrow Q(s,a) + \alpha\big(Q^{\mathrm{ret}} - Q(s,a)\big)$
  end for
end for

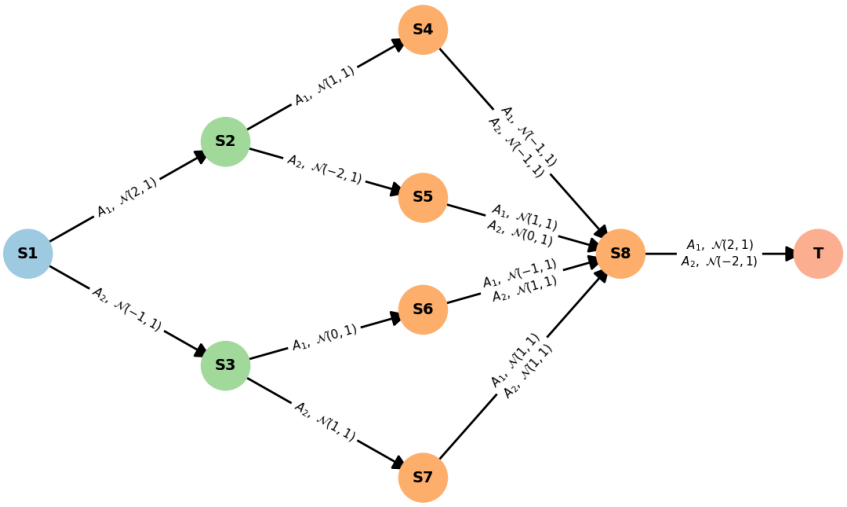


Figure 4.1: Simple MDP with two actions. Each state-action pair yields a reward sampled from a Gaussian distribution with distinct means and unit variance.

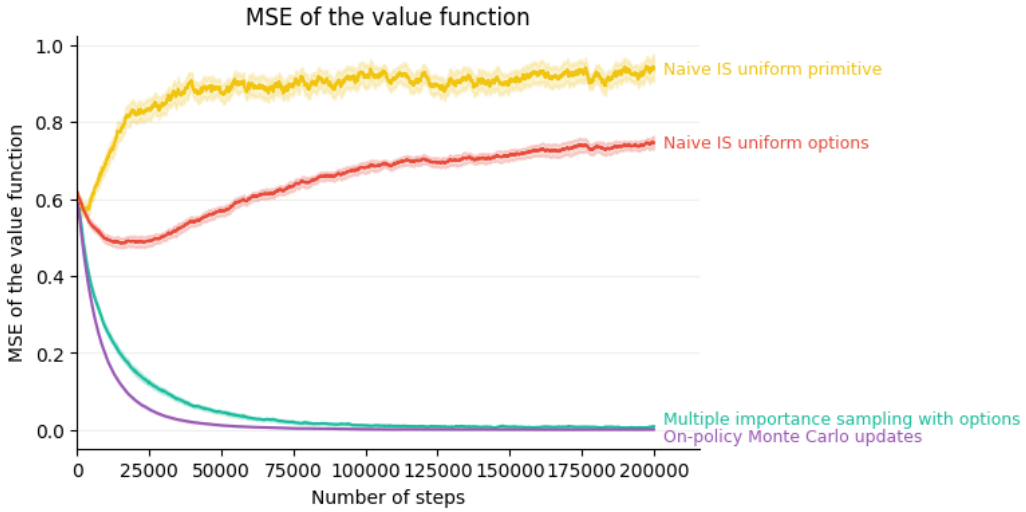


Figure 4.2: Mean squared error (MSE) of value estimates on the small tabular MDP from Figure 4.1 as a function of the number of updates. Curves are averaged over 30 runs; shaded regions denote 95% confidence intervals.

$s \in \{1, \ldots, 8\}$ and two actions $a \in \{1, 2\}$. Transitions are deterministic given (s, a) and each state–action pair yields a Gaussian reward with unit variance and state–action–dependent mean.

The target policy π_{target} we wish to evaluate is given in Table 4.1. Behavior data are generated under three options executed in call-and-return fashion. Their intra-option policies π_ω are shown in Table 4.1, where we list only $\pi_\omega(s, a = 1)$,

Table 4.1: Target policy $\pi_{\text{target}}(s, a)$ and intra-option policies $\pi_\omega(s, a)$ for the small MDP. For the options, only $\pi_\omega(s, a = 1)$ is shown; the probability of action 2 is $1 - \pi_\omega(s, a = 1)$.

State s	$\pi_{\text{target}}(s, a = 1)$	$\pi_{\omega=1}(s, a = 1)$	$\pi_{\omega=2}(s, a = 1)$	$\pi_{\omega=3}(s, a = 1)$
1	0.7	0.8	0.5	0.2
2	0.7	0.5	0.8	0.3
3	0.4	0.6	0.7	0.5
4	0.9	0.6	0.6	0.1
5	0.5	0.9	0.5	0.2
6	0.2	0.4	0.5	0.4
7	0.3	0.7	0.9	0.4
8	0.4	0.6	0.8	0.1

$\pi_\omega(s, a = 2) = 1 - \pi_\omega(s, a = 1)$. Each option can be initiated in any state with probabilities $[0.2, 0.1, 0.7]$ over $(\omega_1, \omega_2, \omega_3)$, and terminates with probabilities $[0.5, 0.5, 0.2]$ for $(\omega_1, \omega_2, \omega_3)$, independently of the state.

From an offline dataset generated by these options, we estimate the value function of π_{target} using four methods: on-policy Monte Carlo updates (oracle baseline, using data collected directly under π_{target}); our multi-step importance sampling with options (MIS); a naive IS scheme that assumes a uniform behavior distribution over primitive actions; and a naive IS scheme that assumes a uniform distribution over options when forming the behavior policy.

Figure 4.2 reports the mean squared error (MSE) of the value estimate as a function of the number of steps, averaged over 30 runs with 95% confidence intervals. The results show that our MIS estimator closely tracks the on-policy Monte Carlo curve and converges to the correct values. However, since it has a higher variance, it presents a slightly higher MSE on average. In contrast, both naive IS variants exhibit substantially higher MSE and do not converge to correct values: assuming a uniform primitive behavior or a uniform option distribution leads to persistent bias because

Table 4.2: K-armed bandit: target policy and option-induced action distributions.

Policy	Pr(policy)	a_1	a_2	a_3	a_4	a_5
Target policy π	–	0.35	0.20	0.30	0.05	0.10
Option ω_1	0.30	0.30	0.20	0.10	0.10	0.30
Option ω_2	0.20	0.10	0.50	0.10	0.05	0.25
Option ω_3	0.10	0.10	0.30	0.10	0.20	0.30
Option ω_4	0.10	0.40	0.10	0.15	0.15	0.20
Option ω_5	0.30	0.10	0.30	0.40	0.05	0.15
Reward distribution	–	$\mathcal{N}(3,1)$	$\mathcal{N}(-2,1)$	$\mathcal{N}(1,1)$	$\mathcal{N}(0,1)$	$\mathcal{N}(-3,1)$

the importance weights are mismatched with the true option-induced behavior policy.

4.4.2 Importance of coverage

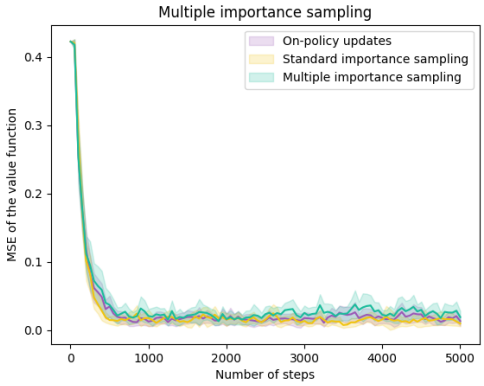
To evaluate the importance of coverage over all options, that is, Assumption 1, we conduct a simple experiment using a 5-armed bandit problem. In this setup, we define five distinct options, ω_i, where each is a policy with a different probability distribution over the arms. The reward for each arm is sampled from a Gaussian distribution with unit variance and arm-specific means, as summarized in Table 4.2 and 4.3.

We evaluate three different approaches in this setting. The first applies standard on-policy learning without any importance sampling, where the behavior policy coincides with the target policy. The second uses conventional importance sampling, weighting updates according to the true probability of selecting each arm. The third employs our MIS method described in the previous section, which does not rely on knowing the probability of selecting each individual arm.

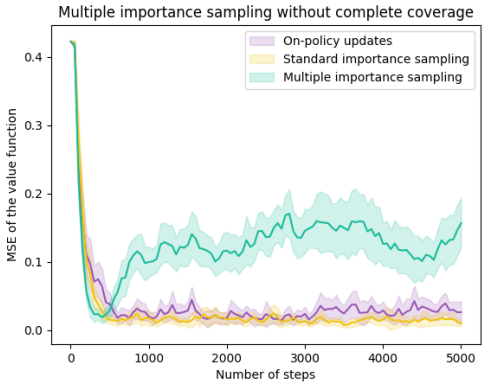
The first two methods serve as oracle baselines: in realistic offline settings we may neither be able to collect on-policy data nor have access to the exact behavior probabilities required for standard IS. Nonetheless, these oracles are useful to characterize how quickly and how stably (in terms of variance) one could learn under idealized

Table 4.3: K-armed bandit without coverage: target policy and option-induced action distributions. Some options assign zero probability to certain arms.

Policy	$Pr(\text{policy})$	a_1	a_2	a_3	a_4	a_5
Target policy π	–	0.35	0.20	0.30	0.05	0.10
Option ω_1	0.30	0.30	0.20	0.10	0.10	0.30
Option ω_2	0.20	0.10	0.00	0.10	0.55	0.25
Option ω_3	0.10	0.00	0.00	0.50	0.20	0.30
Option ω_4	0.10	0.40	0.00	0.10	0.30	0.20
Option ω_5	0.30	0.30	0.00	0.00	0.30	0.40
Reward distribution	–	$\mathcal{N}(3,1)$	$\mathcal{N}(-2,1)$	$\mathcal{N}(1,1)$	$\mathcal{N}(0,1)$	$\mathcal{N}(-3,1)$



(a) Experiment on the K-armed bandit described in Table 4.2. This experiment shows if the coverage assumption is respected, MIS for options converges is unbiased.



(b) Experiment on the K-armed bandit described in Table 4.3. This experiment shows that without coverage for all options, the MIS for options can be biased.

Figure 4.3: Comparison of three approaches: on-policy TD, standard IS (using the correct importance probabilities), and MIS over options, on the K-armed bandit described in Table 4.2. Results are averaged over 30 independent runs, with the shaded region denoting the 95% confidence interval.

conditions, and to contextualize the performance of our MIS-for-options estimator.

The results presented in Figure 4.3a, shows that with coverage, MIS for options correctly converges to the right value function. However, if we violate Assumption 1 from Theorem 1, if some options assign zero probability to certain arms, we can

no longer guarantee that the MIS estimator will converge to an unbiased estimate. Figure 4.3b shows the results for the k-armed bandit problem defined in Table 4.3, illustrating that the estimator converges to an incorrect value function, despite having full coverage when aggregating across all options. This shows the importance of having options with non-zero probability of picking each arm.

Assumption 1 is a *per-option* support condition. MIS treats each starting option ω as a distinct option-conditioned behavior component d_ω. Therefore, if some d_ω assigns zero probability to a trajectory that has positive probability under the target d_π, then d_π/d_ω is undefined and unbiasedness can fail, even if the mixture over options has full support.

Chapter 5

ROD Cycle and partial observability

In this chapter, we empirically investigate exploration using the ROD cycle (Machado et al. 2023) under partial observability. We start from the most common and intuitive approach, namely adding memory to the option learner. We then investigate the multi-step credit assignment for learning option policies using the MIS-Retrace update from Chapter 4.

5.1 Experimental setup

All experiments use the grid-based environments introduced in Figures 3.1 and 3.2. Each environment is defined as a deterministic MDP over grid cells, and partial observability is induced by mapping multiple underlying states to the same observation. As discussed earlier, aliasing can occur in large rooms, corridors, or bottlenecks; in some environments aliased cells lie precisely in narrow passages, which makes purely Markovian policies particularly brittle.

We measure exploration in terms of coverage of the true state space of the MDP, as defined in Equation 3.1.

In the ROD cycle, we start with an empty set of learned options Ω, and the agent interacts with the POMDP using a mixture of primitive random actions and options. The resulting dataset $\mathcal{D}$ consists of POMDP-state transitions and records both the

action-level and option policies. From $\mathcal{D}$ we estimate a successor representation over POMDP states using temporal-difference learning, and extract the eigenvectors with highest eigenvalues via standard linear algebra routines. Each eigenvector e induces an intrinsic reward function of the form $r_e(s, s') = e(s') - e(s),$. For each intrinsic reward function, we then train a corresponding option using either tabular Q-learning, ACER with MIS-Retrace, or one of the recurrent learners (recurrent ACER or DRQN), depending on the experimental condition. Once trained, the options are added to the option set Ω for use in the next ROD cycle.

At each time step, with probability $1 - p_{\text{option}}$, the agent selects a random primitive action, and with probability p_{option} it samples an option uniformly from the current option set Ω and executes it until termination. Episodes have a fixed length of 1000 steps. At the end of each episode, a new eigenoption ω is learned from the dataset composed with all trajectories and it is added to the set of options Ω.

For the termination of each option, we used the termination rule proposed by Machado, Bellemare, and Bowling (2017) when using Q-learning, in which $\beta(s, \omega) = 1$ if $Q_\omega(s, a) \leq 0$ for all $a \in \mathcal{A}$ and $\beta(s, \omega) = 0$ otherwise. In contrast, for ACER-based methods and DRQN we followed the approach proposed by Klissarov and Machado (2023), defining a constant probability $K \in [0, 1]$ for termination $\beta(\omega, s) = K$ for all $s \in \mathcal{S}$ and all $\omega \in \Omega$. This distinction is important because Machado, Bellemare, and Bowling (2017) proved that for an eigenoption ω learned using a value based method in a finite and ergodic MDP with $\gamma \in [0, 1)$, there is at least one state $s \in \mathcal{S}$ such that $\beta_\omega(s) = 1$. However, once we move to more complex algorithms such as ACER, DRQN, or recurrent ACER, these guarantees no longer hold especially considering generalization. Using a simple random termination avoids implicitly relying on specific properties of tabular Q-learning and provides a termination mechanism that does not conflict with the broader class of algorithms considered here.

We tuned hyperparameters in a small grid search using five seeds for the six center environment. For each method we select the configuration that visits all the

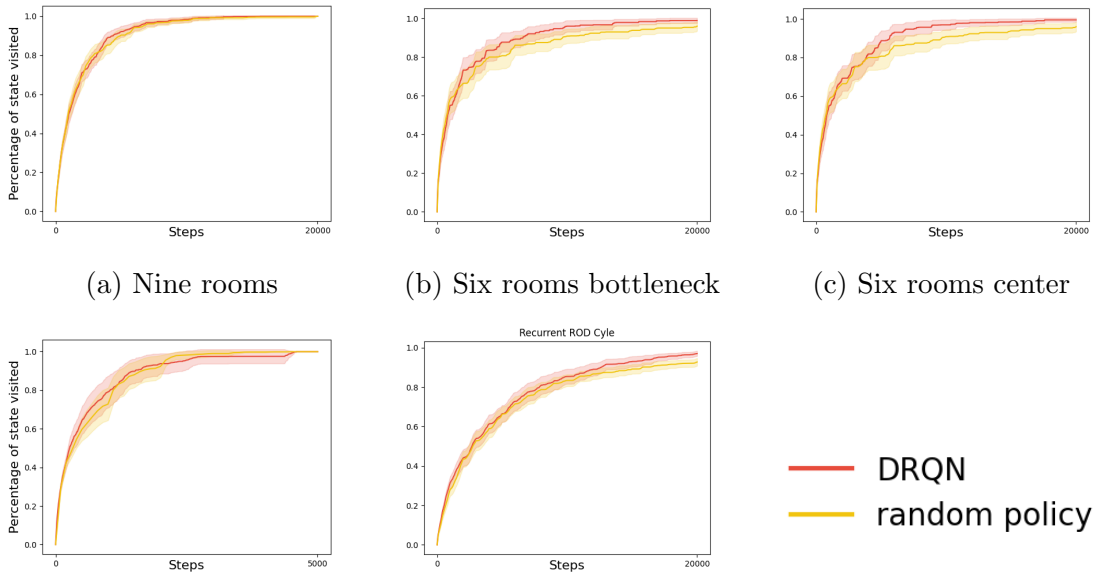
states the fastest on average. We then re-run the chosen configuration with thirty independent seeds for all environments. All plots report the mean over these thirty seeds, with shaded regions corresponding to a 95% confidence interval. Complete hyperparameters information is provided in Appendix B.

5.2 Memory-based eigenoptions

The most natural response to partial observability is to give the agent memory. If individual observations are aliased, then a recurrent option learner that conditions on past observations might recover a more informative internal state and thereby learn better temporal abstractions. We begin by testing exactly this idea.

In this experiment, we replaced the eigenoption learner, that is generally a DQN-like agent, by a DRQN agent based on an adaptation of PFRL's implementation (Fujita et al. 2021). Each observation is a one-hot vector encoding of the aliased cell. This vector is passed through a GRU (Cho et al. 2014) whose hidden state is fed into a linear head producing $Q_\omega(o_t, a)$ for primitive actions. Training is purely offline: complete episodes collected under the options framework are stored in a replay buffer, a short burn-in segment is used to warm up the GRU hidden state as described by Kapturowski et al. (2019), and TD updates are applied on the remaining steps using the intrinsic reward $r_e(s, s') = e^\top\big(\phi(s') - \phi(s)\big)$ associated with the current eigenvector.

Figure 5.1 compares the DRQN-based eigenoptions against a random baseline that selects primitive actions uniformly at random. Across all five environments, coverage curves for DRQN and random are nearly indistinguishable. There is no robust pattern where DRQN eigenoptions clearly outperform random exploration. We suspect that, under the constraint of few thousands of transitions, the recurrent network is simply too data-hungry. It must learn a belief-like hidden state and a policy from limited experience. We also tested increasing the number of training passes over the same offline dataset (i.e., more replay updates without collecting additional data), but



(a) Nine rooms (b) Six rooms bottleneck (c) Six rooms center

(d) Two rooms (e) Sixteen rooms

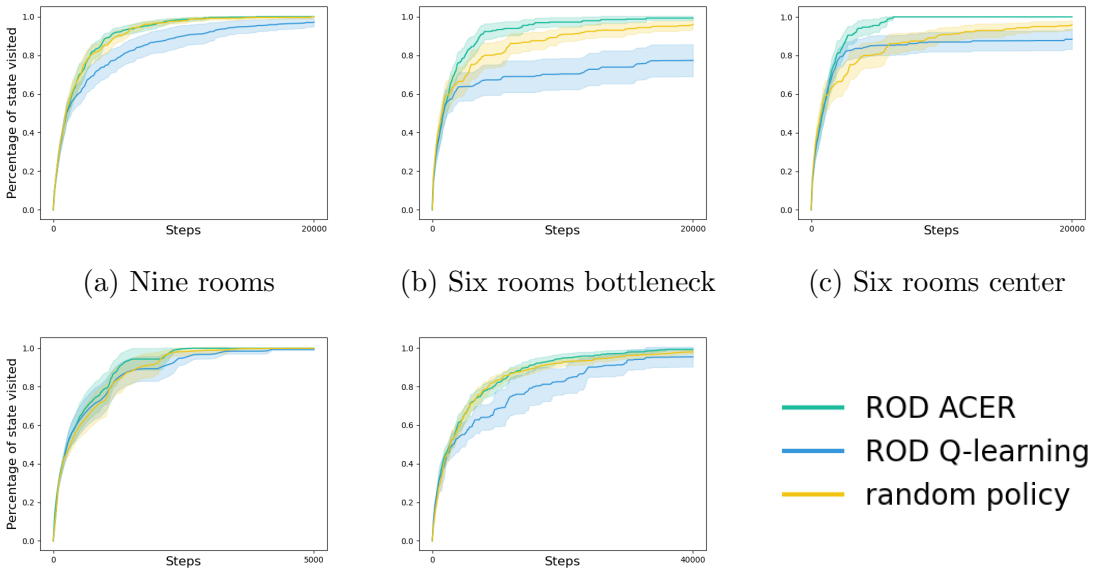
Figure 5.1: Coverage as a function of environment steps for DRQN-based eigenoptions versus a random primitive baseline in the grid-based POMDP environments. Each curve shows the mean over 30 independent runs, with shaded regions denoting 95% confidence intervals. The recurrent DRQN learner does not produce systematic improvements over random exploration.

this did not lead to consistent improvements, suggesting that the limitation is data diversity rather than insufficient optimization.

5.3 Eigenoptions with multi-step credit assignment

Since increasing model capacity via memory does not improve exploration under our data constraints, we now turn to the alternative emphasized in the introduction: focus on better multi-step credit assignment. In this section, eigenoptions are learned by memoryless agents, but their updates use the Multiple Importance Sampling Retrace (MIS-Retrace) scheme proposed in Chapter 4.

Eigenpurposes are constructed as before. We consider two learners. The first is a tabular Q-learner baseline over POMDP observation, which updates $Q_\omega(o, a)$ from one-step TD targets using the intrinsic reward $r(o, o')$. The second is a linear ACER agent also based on an adaptation of PFRL's implementation, but using MIS-Retrace



(a) Nine rooms (b) Six rooms bottleneck (c) Six rooms center

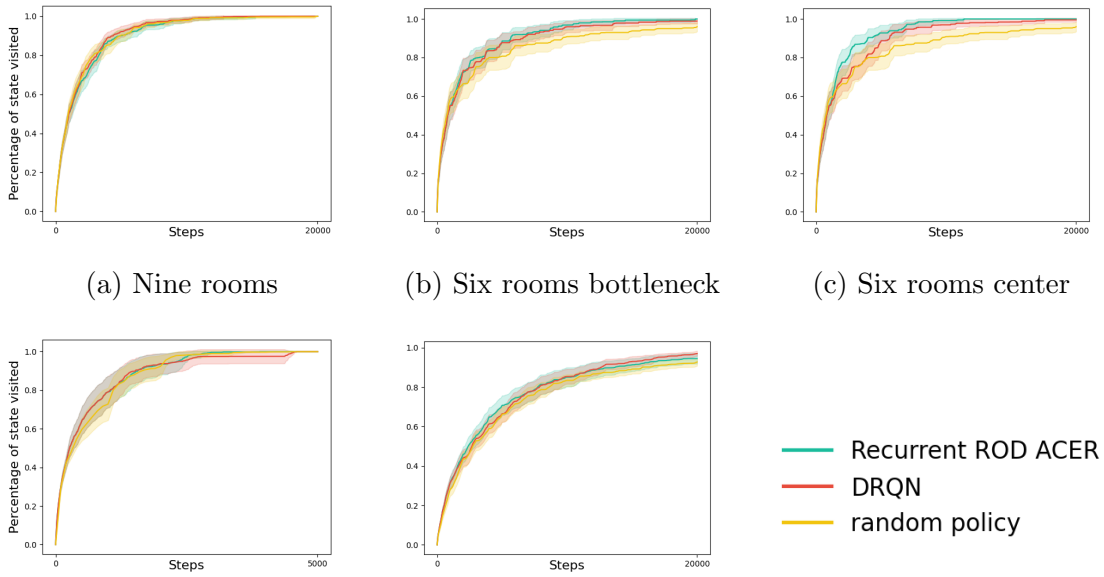
(d) Two rooms (e) Sixteen rooms

Figure 5.2: Coverage over time in the grid-based POMDP environments for three exploration strategies: random actions, ROD Q-learning, and ROD ACER. Each curve shows the mean over 30 independent runs, with shaded regions denoting 95% confidence interval. Tabular Q-learning eigenoptions perform worse than random in most enviornments due to loops induced by partial observability.

to perform multi-step off-policy learning from the same trajectories. Figure 5.2 summarizes the results.

Two main effects are visible. First, in general, tabular Q-learning eigenoptions perform worse than random exploration. This behavior exactly matches the analysis in Chapter 3, where we showed that eigenoptions can create infinite loops, becoming trapped under partial observability until the episode finishes.

Second, ACER eigenoptions trained with multi-step credit assignment using MIS-Retrace improve, or is at least the same as, the random baseline. Even though the learner is memoryless, the multi-step off-policy update makes better use of option-generated trajectories and avoids problems caused by the bootstrapping. The multiple importance sampling correction behavior allows the agent to assign credit across longer time spans without biasing value estimates.



(a) Nine rooms (b) Six rooms bottleneck (c) Six rooms center

(d) Two rooms (e) Sixteen rooms

Figure 5.3: Coverage over time for recurrent ACER eigenoptions trained with MIS-Retrace versus a random primitive baseline. Each curve shows the mean over thirty seeds, with shaded regions denote 95% confidence intervals over 30 runs. Even though the learner combines memory with multi-step importance sampling, there is no consistent improvement over random exploration under this data regime.

5.3.1 Recurrent ACER with multi-step credit assignment

As an ablation study, we also tested a recurrent version of ROD ACER. In this variant, POMDP observations are again fed into a GRU, and the hidden state is used by actor and critic heads. Training uses the MIS-Retrace update to perform multi-step off-policy evaluation and policy improvement from trajectories generated by a mixture of primitive actions and options.

This model combines both ideas: it has memory, and it uses multi-step credit assignment tailored to data generated under options.

Figure 5.3 shows coverage curves for recurrent ACER with MIS-Retrace compared to the same random baseline. The outcome closely mirrors the DRQN case. Despite using MIS-Retrace internally, the recurrent ACER eigenoptions do not show clear gains over random exploration. The presence of memory and a more principled multi-

step update is not enough to compensate for the scarcity of data.

5.4 Reward maximization

In the previous sections we evaluated eigenoptions only through their effect on exploration, measuring how quickly the ROD cycle increases coverage under partial observability. Reward maximization provides a more direct test of whether the discovered options are actually useful for solving tasks. In this section, each grid-based POMDP is turned into a goal-seeking problem by designating a single goal state in the environment. Episodes terminate as soon as the agent reaches the goal—giving the agent a +1 reward (the reward is 0 otherwise)— or after a maximum of 1000 interaction steps, whichever comes first. The underlying transition dynamics, observation mapping, and aliasing pattern are identical to those used in the coverage experiments; only the task and stopping criterion are changed.

We compare four agents in this setting. The first is a tabular Q-learning baseline that selects primitive actions from a POMDP-state value function and optimizes the extrinsic goal-reaching reward directly. The second augments this baseline with eigenoptions discovered by the ROD cycle, yielding an agent whose behavior is a mixture of primitive Q-learning and options executed in the POMDP. The third agent is an ACER learner with linear function approximation over observations, again using only primitive actions. The fourth combines ACER with eigenoptions trained from option-generated trajectories using the MIS-Retrace update from Chapter 4, resulting in a ROD ACER agent whose behavior policy interleaves primitive ACER actions with options as in the coverage experiments.

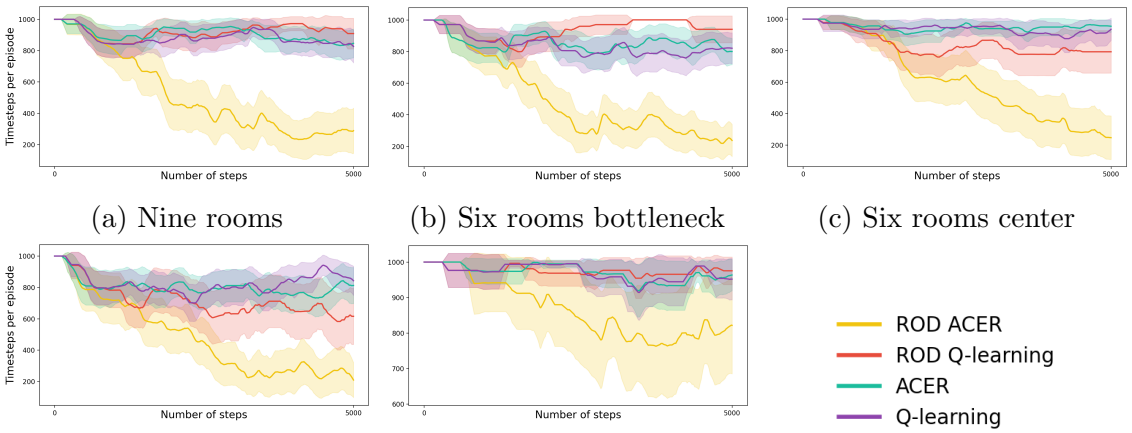
At a high level, the ROD cycle alternates between three stages. First, a behavior policy that can already use options interacts with the POMDP and collects a dataset of trajectories. Each transition is recorded together with the option (if any) that was active and the probabilities with which the behavior policy selected that option and the underlying primitive actions. Second, this dataset is used to estimate a

successor representation over POMDP states (or observations), capturing how often each state tends to be followed by others under the current behavior. A spectral decomposition of this successor representation yields eigenvectors that encode broad modes of transition structure in the environment. Third, each eigenvector is turned into an intrinsic reward function, and a new option policy is trained to maximize this intrinsic reward while respecting the option's termination rule. The new options are then added to the option set used by the behavior policy, and the cycle repeats. In the ACER-based setting, the option-learning phase uses MIS-Retrace to obtain a multi-step, off-policy update that remains unbiased despite being trained from replay data rather than fresh on-policy rollouts.

For evaluation, we focus exclusively on the number of steps per episode. Shorter episodes indicate that the agent is reaching the goal more quickly, whereas episodes that consistently last close to 1000 steps reflect failure to solve the task. As training proceeds, we record the length of each episode and align these measurements with the total number of environment steps taken so far. For each algorithm and environment, we then compute the average episode length as a function of training progress, together with 95% confidence intervals across random seeds, using the same smoothing and plotting procedure as in the coverage experiments. Figure 5.4 shows the resulting curves.

The tabular Q-learning agent with ROD eigenoptions does not reliably improve over its primitive-only counterpart and can even be slightly worse in some environments. This is consistent with the analysis in Chapter 3: under partial observability, eigenoptions learned from the aliased POMDP can generate loops, leading to long episodes that frequently hit the 1000-step limit.

In contrast, the ROD ACER agent equipped with MIS-Retrace eigenoptions outperforms the simple ACER baseline. Across environments, the ROD ACER learning curves drop more rapidly and stabilize at shorter episode lengths, showing that the agent reaches the goal in fewer steps after the same amount of interaction. This



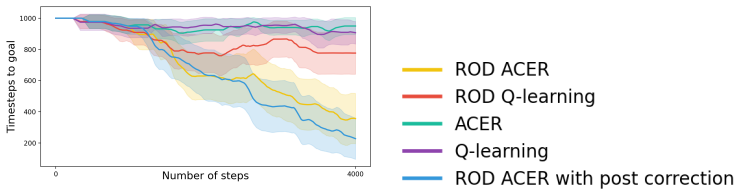
(a) Nine rooms (b) Six rooms bottleneck (c) Six rooms center

(d) Two rooms (e) Sixteen rooms

Figure 5.4: Moving average number of steps per episode. Each curve has a window of 100, and shows the mean over 30 independent runs, with shaded regions denoting 95% confidence interval. Smaller number is better since it represents the agent arriving faster at the goal.

indicates that the eigenoptions discovered by the ROD cycle are useful for reward maximization tasks even in partially observable environments. Notably, in all environments plain ACER eventually achieves similar performance, but it typically requires substantially more interaction to do so—on the order of 50,000 to 250,000 compared to 5,000 environment steps depending on the environment.

Posterior-correction ablation. We add a fifth curve that differs from ROD ACER with MIS-Retrace only in the computation of the per-step ratios used inside Retrace. The default implementation uses the prior-only option belief $\tilde{p}_k^{\text{pre}}$ propagated by Equation (4.26), forming $\tilde{b}_{\omega_t}$ and $\tilde{\rho}_k$ as in Equation (4.27). The posterior-corrected variant instead computes p_k^{pre} by filtering using Equations (4.10)–(4.12), forming $b_{\omega_t}(a_k \mid H_k^{\text{pre}})$ and $\rho_k = \pi(a_k \mid s_k)/b_{\omega_t}(a_k \mid H_k^{\text{pre}})$. Figure 5.5 shows that the posterior-corrected curve is qualitatively similar to the prior-only variant in this environment.



(a) Nine rooms

Figure 5.5: Moving average number of steps per episode on *Six rooms center*, comparing the default MIS-Retrace ratios (prior-only option belief) against the posterior-corrected ratios. Shaded regions denote 95% confidence intervals over 30 runs.

5.5 Discussion

Starting from the intuitive choice of adding memory, we found that recurrent eigenoption learners, both DRQN-based and recurrent ACER with MIS-Retrace do not outperform a random baseline when each eigenoption is trained from only a few 1000-step episodes.

When we instead keep the option learners memoryless and change only the learning rule, the results reverse. A simple tabular Q-learning eigenoption learner performs worse than random exploration because, under partial observability, it tends to produce deterministic eigenoptions that become trapped in loops. In contrast, a memoryless ACER learner using MIS-Retrace produces eigenoptions that reliably accelerate coverage and reduce diffusion time in the same environments.

The reward maximization experiments highlight two key points. First, eigenoptions learned from aliased observations do not automatically help reward maximization when combined with simple one-step tabular methods, and can even be detrimental if they present loops. Second, when option discovery is coupled with a multi-step off-policy algorithm that admits unbiased corrections—here, ACER with MIS-Retrace—the ROD cycle can produce options that significantly accelerate learning on sparse goal-reaching tasks. The performance of ROD ACER compared to plain ACER demonstrates that the ROD cycle can be beneficial beyond pure coverage metrics in partially observable environments.

A practical caveat is that these gains come with substantially higher computational cost. ACER with MIS-Retrace performs many more gradient updates per transition (often multiple passes over the same offline dataset) and incurs additional overhead to compute multi-step targets and importance ratios, whereas the random baseline and tabular Q-learning updates are comparatively cheap. As a result, the comparisons in this chapter should be interpreted primarily as data-efficiency comparisons: for a fixed amount of interaction data, MIS-Retrace improves how effectively that data is used, but it does not necessarily improve compute-efficiency when measured in wall-clock time or total FLOPs.

Beyond these specific comparisons, the experiments also provide an illustration of how the offline learning algorithm developed in Chapter 4 can be used in practice. MIS-Retrace was designed as a principled way to perform multi-step importance sampling when trajectories are generated under options, and here it serves as a plug-in replacement for standard TD targets in a simple actor-critic eigenoption learner. Although our goal was not to show that MIS-Retrace is universally superior, the results demonstrate that it can be applied successfully in an offline setting, more specifically for option-discovery, where the behavior policy is a mixture of options and primitive actions, and the reward used for learning differs from the reward under which the data were collected.

Chapter 6

Conclusion

This thesis characterizes the interaction between eigenoptions and partial observability. Our analysis shows that partial observability reshapes not only the agent's perception of the environment but also the geometry of the intrinsic control problems that the successor representation induce.

Furthermore, we introduced an unbiased multi-step learning algorithm for options using importance sampling correction, addressing the distribution mismatch problem inherent in learning from option-generated data. Empirically, the work demonstrates that while Q-learning often becomes trapped in loops due to state aliasing, an ACER learner with the proposed MIS-Retrace generates eigenoptions that accelerate state-space coverage and reduce diffusion time on most experiments.

Finally, our reward-maximization experiments demonstrate that these insights extend beyond exploration and into downstream control. The proposed MIS-Retrace approach enables ACER to effectively learn beneficial option policies even in the presence of aliased observations. The ROD cycle instantiated with our multi-step learner consistently outperforms standard baselines, suggesting that unbiased multi-step credit assignment is not only compatible with option discovery, but in fact essential for stable optimization under partial observability. Together, these findings illustrate that temporal abstractions remain a powerful tool for RL, but only with the proper care during learning, especially in the face of partial observability.

While this thesis evaluates eigenoptions in partially observable tabular environments, an avenue for future research is to validate these methods in more complex, high-dimensional environments using function approximation. A possible set of environments is the POBAX benchmark (Tao et al. 2025), which offers a suite of higher complexity domains designed to test agents under difficult conditions of partial observability. Additionally, another open interesting investigation is exploring different values for weights $\eta_\omega(s)$ that balance the importance of each option ω for each state. As shown by Veach (1998), smartly picked weights have theoretical lower variance, and this could improve MIS-Retrace learning.

References

Allen, C., Kirtland, A., Tao, R. Y., Lobel, S., Scott, D., Petrocelli, N., Gottesman, O., Parr, R., Littman, M., & Konidaris, G. (2024). Mitigating partial observability in sequential decision processes via the lambda discrepancy. *Neural Information Processing Systems (NeurIPS)*.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-13*, 834–846.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural computation*, *5*(4), 613–624.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., & Kavukcuoglu, K. (2018). IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. *International Conference on Machine Learning (ICML)*.

Fujita, Y., Nagarajan, P., Kataoka, T., & Ishikawa, T. (2021). ChainerRL: A deep reinforcement learning library. *Journal of Machine Learning Research (JMLR)*, *22*(77), 1–14.

Guo, Z. D., Thomas, P. S., & Brunskill, E. (2017). Using options and covariance testing for long horizon off-policy policy evaluation. *Neural Information Processing Systems (NeurIPS)*.

Hausknecht, M. J., & Stone, P. (2015). Deep recurrent Q-learning for partially observable MDPs. *AAAI Fall Symposia*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, *79*(8), 2554–2558.

Jain, A., & Precup, D. (2018). Eligibility traces for options. *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1), 99–134.

Kapturowski, S., Ostrovski, G., Dabney, W., Quan, J., & Munos, R. (2019). Recurrent experience replay in distributed reinforcement learning. *International Conference on Learning Representations (ICLR)*.

Klissarov, M., Bagaria, A., Luo, Z., Konidaris, G., Precup, D., & Machado, M. C. (2025). Discovering temporal structure: An overview of hierarchical reinforcement learning. *ArXiv*, 2506.14045.

Klissarov, M., & Machado, M. C. (2023). Deep Laplacian-based options for temporally-extended exploration. *International Conference on Machine Learning (ICML)*.

Klissarov, M., & Precup, D. (2021). Flexible option learning. *Neural Information Processing Systems (NeurIPS)*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*(3), 293–321.

Machado, M. C., Barreto, A., Precup, D., & Bowling, M. (2023). Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research (JMLR)*, *24*(80), 1–69.

Machado, M. C., Bellemare, M. G., & Bowling, M. (2017). A Laplacian framework for option discovery in reinforcement learning. *International Conference on Machine Learning (ICML)*.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., & Campbell, M. (2018). Eigenoption discovery through the deep successor representation. *International Conference on Learning Representations (ICLR)*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533.

Munos, R., Stepleton, T., Harutyunyan, A., & Bellemare, M. (2016). Safe and efficient off-policy reinforcement learning. *Neural Information Processing Systems (NeurIPS)*.

Precup, D., Sutton, R. S., & Singh, S. (2000). Eligibility traces for off-policy policy evaluation. *International Conference on Machine Learning (ICML)*.

Ramesh, R., Tomar, M., & Ravindran, B. (2019). Successor options: An option discovery framework for reinforcement learning. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Rubinstein, R. Y., & Kroese, D. P. (1981). *Simulation and the Monte Carlo method*. Wiley Publishing.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. The MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*(1), 181–211.

Tao, R. Y., Guo, K., Allen, C., & Konidaris, G. (2025). Benchmarking partial observability in reinforcement learning with a suite of memory-improvable domains. *Reinforcement Learning Journal*.

Veach, E. (1998). *Robust Monte Carlo methods for light transport simulation* [Doctoral dissertation, Stanford University].

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., & de Freitas, N. (2017). Sample efficient actor-critic with experience replay. *International Conference on Learning Representations (ICLR)*.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292.

Appendix A: State space exploration with eigenoptions

This appendix provides full details required to reproduce the eigenoption discovery and exploration experiments reported in Chapter 3. We describe the computation of the successor representation (SR), the construction of eigenoptions, and the hyper-parameters used throughout.

A.1 Successor representation and eigenoptions

For each environment, we estimate the *successor representation* (SR) of a uniform random policy. In the fully observable case, the random-policy transition matrix is

$$P_\pi(s, s') = \tfrac{1}{4} \sum_{a=0}^{3} P(s'|s, a).$$

In the partially observable case, this same construction is applied over observations, treating aliased observations as indistinguishable.

The SR is computed using the closed-form expression

$$\Psi = (I - \gamma P_\pi)^{-1}, \qquad \gamma = 0.975.$$

We discard the constant eigenvector, and each remaining eigenvector e_i defines two intrinsic rewards:

$$r_e(s, s') = e(s') - e(s), \qquad r_{-e}(s, s') = e(s) - e(s').$$

Each intrinsic reward induces one eigenoption. We compute eigenoption policies

via **tabular value iteration**:

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)\big(r(s,s') + \gamma V_k(s')\big),$$

until convergence ($\|V_{k+1} - V_k\|_\infty < 10^{-3}$). The greedy policy $\pi_\omega(s) = \arg\max_a Q(s,a)$ is used as the intra-option policy, and termination is set to $\beta_\omega(s) = 1$ whenever no positive continuation value remains.

A.2 Exploration protocol

During exploration, the agent mixes primitive actions with eigenoptions. At each time step it:

- executes a random primitive action with probability 0.95; or
- selects a random eigenoption with probability 0.05 and acts according to its policy until its termination.

Each run lasts 50 000 steps (episodes capped at 1 000). Each configuration (0, 8, or 32 eigenoptions) is repeated over 30 seeds, reporting mean and 95% confidence intervals.

A.3 Hyperparameters

Table A.1 summarizes the hyperparameters used for SR estimation, eigenoption construction, and the exploration protocol.

Parameter	Value
Discount factor (γ)	0.975
Value iteration tolerance	10^{-3}
Number of eigenoptions	0, 8, 32
Option execution probability	0.05
Max steps per episode	1 000
Max steps per run	50 000
Seeds per configuration	30

Table A.1: Hyperparameters used in the eigenoption diffusion experiments.

Appendix B: Implementation and hyperparameter details

This appendix provides additional details about the implementation of the environments, the ROD cycle, and the option-learning algorithms used in Chapter 5. The goal is to make the experiments reproducible and to clarify the practical role of the hyperparameters.

B.1 Environment configuration

The underlying state space $\mathcal{S}$ consists of all reachable grid cells. A deterministic transition function maps (s, a) to a next state s' according to the intended movement (up, down, left, right) respecting walls and boundaries.

Partial observability is implemented by defining a mapping $\phi : \mathcal{S} \to \mathcal{O}$ that groups multiple states into aliased observations. In the figures, colored cells indicate states that share the same observation under ϕ, while white cells are uniquely identifiable. The starting state is fixed and highlighted in light green. The environments are constructed so that some rooms contain large clusters of aliased states, some corridors and bottlenecks (for example doorways) contain aliased states that make it ambiguous for a memoryless agent to infer whether it is entering or leaving a room, and in certain environments (such as PO-SIX-ROOMS BOTTLENECK) the only way to cross a bottleneck reliably may require stochastic policies, since the same observation can correspond to states on different sides of the bottleneck.

B.2 Tabular Q-learning options

In the tabular setting, each intrinsic reward induces a separate tabular Q-function, $Q_\omega(s, a)$, which is updated using the standard Q-learning rule:

$$Q_\omega(s_t, a_t) \leftarrow Q_\omega(s_t, a_t) + \alpha\Big[r_t^{\text{eig}} + \gamma \max_{a'} Q_\omega(s_{t+1}, a') - Q_\omega(s_t, a_t)\Big], \tag{B.1}$$

where α is chosen from the grid described in Section B.5. An ε-greedy policy with respect to Q_ω is used during training.

The termination function β_ω is implemented as a simple heuristic based on the magnitude of Q_ω.

B.3 ACER with MIS-Retrace

For the ACER-based option learner in the memoryless setting, we use a single layer network over one-hot POMDP states, with separate policy and value heads. Learning uses MIS-Retrace as introduced in Section 4.3, where importance sampling ratios are truncated to control variance and multi-step returns are constructed up to a horizon $t_{\max}$.

In the recurrent setting, we instead use a GRU-based model with the architecture

$$\text{one-hot}(s_t^{\text{P}}) \xrightarrow{\text{GRU}} h_t \xrightarrow{\text{LayerNorm}} \text{ACER head (policy and Q)}.$$

The same MIS-Retrace update is used, but the hidden state h_t carries information across time, allowing the option policy and values to depend on observation history.

B.4 DRQN-based options

The DRQN-based option learner uses a GRU core followed by a linear layer and a discrete action-value head:

$$\text{one-hot}(s_t^{\text{P}}) \xrightarrow{\text{GRU}} h_t \xrightarrow{\text{Linear}} Q(h_t, \cdot).$$

Training is performed via off-policy recurrent Q-learning on episodes drawn from an episodic replay buffer, with burn-in steps used to warm up the recurrent state. A greedy policy with respect to Q defines the option policy.

B.5 Hyperparameter sweeps

This section lists the hyperparameter grids used in the small hyperparameter searches mentioned in Chapter 5. For each algorithm, we evaluated all combinations in the grid using 5 seeds in PO-SIX-ROOM CENTER environment, selected the best configuration, and then re-ran that configuration with 30 new seeds for reporting for all environments.

Tabular Q-learning options

For tabular Q-learning, we tune only the step size for option training:

- Step size $\in \{0.1, 0.03, 0.01, 0.003\}$.

B.5.1 ACER-based options (memoryless)

For the memoryless ACER eigenoption learner, we perform a grid search over:

- Random termination $\in \{0, 0.03, 0.1, 0.3\}$,
- Total number of updates $\in \{100000, 200000\}$;
- Step size $\in \{0.01, 0.03, 0.003, 0.001\}$,
- T-max $\in \{5, 10, 20\}$,
- Truncation threshold $\in \{3, 10, 30\}$,

The larger number of tuned hyperparameters for ACER (and its recurrent variant) reflects the fact that ACER with MIS-Retrace is a significantly more complex algorithm than tabular Q-learning.

B.5.2 Recurrent ACER and DRQN options

For the recurrent ACER eigenoption learner, we use:

- GRU hidden size of 16;
- Step size $\in \{0.0001, 3e-5\}$;
- Total number of updates $\in \{50000, 100000\}$;
- Random termination, the random probability of terminating the option, $\in \{0, 0.03, 0.1, 0.3\}$;
- T-max $\in \{10, 20\}$;
- Truncation threshold $\in \{10, 30\}$;
- Burn-in length of 10,

with values chosen from small grids centred around the best memoryless configuration.

For the DRQN option learner (`ROD_DRQN_pfrl_batched_lstm.py`), we tune:

- GRU hidden size of 16;
- Step size $\in \{0.0001, 3e-5\}$;
- Total number of updates $\in \{50000, 100000\}$;
- Random termination $\in \{0, 0.03, 0.1, 0.3\}$;
- Truncation threshold $in\{10, 30\}$;
- Burn-in length of 10,
- Batch size $in\{32, 64\}$
- Target network update interval $\in \{1000, 3000\}$.

Again, a small grid of reasonable values is explored using 5 seeds per configuration, and the best configuration is then run with 30 new seeds.

B.6 Reward maximization

The reward-maximization experiments in Section 5.4 use the same offline ROD protocol as the coverage experiments. In each ROD cycle a behavior policy that can already call options interacts with the POMDP and collects a dataset of trajectories. This dataset is used to estimate a successor representation over POMDP states, whose leading eigenvectors define intrinsic rewards of the form $r_{\text{eig}}(s, s') = u(s') - u(s)$. For each intrinsic reward, a new option policy and termination rule are learned under this intrinsic reward (using either tabular Q-learning or ACER with MIS-Retrace, depending on the setting), and the resulting option is added to the option set Ω that the behavior policy can invoke in subsequent cycles.

For the tabular Q-learning agents (with and without options), step sizes, discount factors, and other Q-learning parameters are fixed to the values selected in the coverage experiments. We then perform a sweep over:

- Exploration rate $\epsilon \in \{0.05, 0.10, 0.20, 0.40\}$,
- Probability of starting an option $p_{\text{option}} \in \{0.0, 0.3, 0.6, 0.9\}$.

Each configuration is evaluated over five independent runs in the PO-SIX-ROOMS CENTER environment, and we select the pair that reaches the goal most frequently within the first 5 000 environment steps. The final values used in the reward-maximization plots are $\epsilon = 0.20$ and $p_{\text{option}} = 0.60$.

For the ACER-based agents, all ACER-specific hyperparameters (step size, t-max, truncation threshold, number of option-learning steps, discount factors, and network architecture) are kept fixed at the best values found in the memoryless MIS-Retrace experiments. We vary only:

- Probability of starting an option $\in \{0.05, 0.10, 0.20, 0.40\}$,
- Random termination $\in \{0.0, 0.03, 0.10, 0.30\}$.

As in the Q-learning case, each configuration is evaluated over five runs in the PO-SIX-ROOMS CENTER environment, and we choose the setting that yields the highest number of successful goal reaches within 5 000 steps. The final configuration used in the reward-maximization experiments is probability of starting an option in $= 0.10$ and random termination $= 0.30$.