A Geometric Study on Policy Space Compression

Master's Thesis in
Telecommunication Engineering

Author: **Majid Molaei Dehsorkhi**

Student ID: 897472
Advisor: Prof. Marcello Restelli
Co-advisors: Mirco Mutti, Albero Maria Metelli
Academic Year: 2023-24

Acknowledgements

Dear Marcello Restelli, Alberto Maria Metelli and, in particular and most notably, Mirco Mutti, I would like to express my sincere gratitude to all of you for your invaluable guidance and support throughout my Master's thesis. Your insights and suggestions significantly improved the quality of my research and made it a more comprehensive and well-rounded work.

Although due to some personal problems I was obliged to do my thesis remotely from my country. I appreciate your availability and willingness to meet with me online regularly to discuss my progress and address my concerns.

Your guidance has not only helped me complete my thesis but has also enriched my understanding of doing an appropriately constructed scientific research.

I look forward to the possibility of continuing our academic journey together and collaborating on future research projects.

Once again, I am truly grateful for the opportunity to have had you as my supervisors.

Sincerely.

Abstract

In the last decades, reinforcement learning emerged as a powerful tool to address sequential decision making under uncertainty, counting a number of success stories ranging from game solving, to robotic manipulation, and text generation. However, to achieve those impressive breakthroughs, the reinforcement learning algorithms require to be trained on a massive number of samples collected from the environment, which limits the applicability of these techniques to a broader set of real-world problems. A significant source of this samples inefficiency is due to the size of the so-called policy space, i.e., the set of decision strategies from which to learn the one to be deployed. Critically, the more choices of decision strategies we have, the more we can expect to achieve a desirable performance, but the less efficient will the learning process be.

A recent work by Mutti et al. [12] addresses the endemic sample inefficiency of reinforcement learning trying to compress the set of decision strategies as a pre-processing. The goal of this process, which is called policy space compression, is to reduce the size of the policy space by retaining most of its expressive power, so that a small dip in the performance reinforcement learning can achieve is compensated by improved efficiency. Practically, policy space compression tries to find a (small) set of K representative policies that covers the set of state-action distributions induced by all the policies in the original policy space, where the covering is approximated with a covering threshold σ. While an algorithm for policy space compression is also presented in [12], the original paper leaves some important questions open.

In this thesis, we provide a geometric study of the policy space compression problem to provide answers to two pressing questions: How to properly set the covering threshold σ and the number of covering policies K.

First, we provide a study on the value of σ, showing that setting σ to be equal to the number of state-action pairs, the default choice in the previous work, can admit a trivial solution that defeats the purpose of the compression under certain conditions. Then, we provide a more meaningful range of σ that is better aligned with the ultimate goal of policy space compression. Finally, we present computationally tractable algorithms to check whether a tentative compression satisfies the desired covering threshold or not.

In a second part of the dissertation, we instead provide lower and upper bounds on the number of representative policies K that are needed to solve the policy space compression. The derived bounds have a direct dependence with the covering threshold σ as well as all the main parameters of the problem.

To conclude, we hope that this thesis will provide a critical advancement in the understanding of the policy space compression problem, fostering its applicability to a wide set of real-world applications.

Keywords: Reinforcement Learning, Policy Space Compression

Abstract in lingua italiana

Negli ultimi anni le tecniche di apprendimento per rinforzo sono emerse come importante strumento per risolvere problemi di decisione sequenziali nell'ambito dell'intelligenza artificiale. In particolare, l'apprendimento per rinforzo ha portato una serie di successi in giochi strategici, robotica e generazione di testo. Tuttavia, per raggiungere questi successi, gli algoritmi di apprendimento per rinforzo richiedono di essere addestrati su una gigantesca mole di dati, limitandone l'applicazione ad una più ampia classe di problemi. Una delle principali sorgenti di questa inefficienza nell'utilizzo dei dati è da imputare alla dimensione dello spazio delle politiche, ovvero dell'insieme di strategie di decisione da cui estrarre la strategia migliore. Chiaramente, più grande è lo spazio delle politiche, maggiore è la performance che possiamo aspettarci dalla strategia ottima, ma trovare quest'ultima strategia risulterà più inefficiente.

Un articolo recente di Mutti et al. [12] ha proposto di attaccare il problema di inefficienza dell'apprendimento del rinforzo cercando di comprimere lo spazio delle politiche come pre-processing. L'obiettivo di questo pre-processing. che viene chiamato compressione dello spazio delle politiche, è quello di ridurre la dimensione dello spazio delle politiche mantenendo la maggior aprte del suo potere espressivo, così che un fisiologico calo nella performance raggiunta dall'apprendimento per rinforzo è compensata da una migliore efficienza. In pratica, la comrpessione dello spazio delle politiche cerca di identificare un (possibilmente piccolo) insieme di K politiche rappresentative che copre tutte le distribuzioni su stati-azioni indotte dalle politiche dello spazio originale, dove la copertura è approssimata con una soglia di copertura σ. L'articolo originale propone un algoritmo per la compressione dello spazio delle politiche, ma lascia aperte alcune imporanti domande.

Questa tesi fornisce uno studio geometrico del problema di compressione dello spazio delle politiche per rispondere a due importanti domande: come scegliere la soglia di copertura σ e il numero di politiche rappresentative K.

Innanzitutto, la tesi studia il valore di σ, mostrando che scegleire un valore uguale al numero delle coppie stato-azione, scelta standard del lavoro precedente, potrebbe ammettere una soluzione banale che mina l'utilità della compressione dello spazio delle politiche. Quindi proponiamo un più significativo range di valori σ che meglio si allinea con l'obiettivo della compressione. Infine, presentiamo un insieme di algoritmi trattabili per verificare se un tentativo di compressione soddisfa la soglia σ desiderata oppure no.

In una seconda parte del contributo, la tesi fornisce estermi inferiori e superiori sul numero di politiche K che è necessario per risolvere il problema di compressione. Le relazioni in questione hanno una dipendenza diretta con la soglia σ e le quantità più rilevanti del problema.

In conclusione, speriamo che questa tesi possa fornire un avanzamento cruciale nella compresione del problema di compressione dello spazio delle politiche, così da permetterne la sua applicazione ad una vasta classe di problemi.

Parole chiave: Apprendimento per Rinforzo, Compressione dello Spazio delle Politiche

Contents

1 | Introduction

Reinforcement Learning (RL) [19] is a powerful tool to address sequential decision making problems from interactions. In RL, a learning agent (the decision maker) interacts with an unknown environment to learn a policy, i.e., a decision strategy, to maximize some notion of long-term goal, usually expressed in terms of the cumulative reward collected during the interactions. The policy space refers to the set of all possible policies that the agent can choose from. This recipe lead to several breakthroughs, such as in strategic games [1, 11, 18], robotic control [7, 17], and text generation [13].

However, RL problems can become challenging when dealing with a large and potentially infinite policy space. This is particularly true when the policies are parameterized in complex ways, such as deep neural networks. In such cases, searching for an optimal policy or finding a good strategy for interacting with the environment becomes computationally expensive and sample-inefficient.

The *policy space compression* problem [12] aims to address this challenge by identifying a smaller subset of the policy space that retains most of the expressive power of the original space. In other words, it seeks to reduce the size of the policy space while ensuring that the selected subset can still achieve similar state-action distributions as the original space.

The key idea is to find a compressed representation of policies, denoted as Θ', such that for any policy θ in the original policy space Θ, there exists a policy θ' in Θ' such that the difference between their induced state-action distributions is bounded by a positive constant σ. This means that Θ' contains a representative set of policies that can achieve similar outcomes as the original policy space, when employed for RL.

Technically, the problem can be framed as a (soft) version of the set cover problem, where the goal is to find a minimal set of state-action distributions that cover the set of all possible state-action distributions within a certain divergence threshold σ. Unfortunately, set cover problems are known to be NP-hard [3], which means finding an optimal solution can be computationally infeasible. Researchers are actively exploring various approaches to tackle this problem, including approximations and heuristics, to make it more tractable and efficient.

In this context, the problem has being reformulated [12] as a game between two players: A leader and a follower. The leader's role is to distribute a set of policies (represented by $\theta_1, \theta_2, \theta_3, \dots, \theta_K$) with the goal of covering the state-distribution induced by the follower's policy. Conversely, the follower's role is to find a policy (represented by μ) that is not well-covered by the leader's policies. This means the follower aims to find a strategy that maximizes the difference between its state-action distribution and the one represented by the closest leader's policy.

The objective function of the latter game-theoretic reformulation is neither convex nor concave in the leader's and follower's parameters. Thus, Mutti et al. [12] developed an iterative first-order method that is guaranteed to converge to a differential Stackelberg equilibrium [5] with a gradient descent-ascent procedure. However, the latter effort leaves open two important questions:

- How can we set the covering threshold σ in a meaningful way, and how can we check whether our locally optimal solution satisfies the threshold?
- How can we choose the number of representative policy K to be optimized in order to solve the policy space compeession problem for a given threshold σ?

Answering those questions is the main focus of the presented thesis.

1.1. Contributions

In this thesis, we contribute to the advancement of the policy space compression problem in two critical directions, thanks to a novel geometric perspective over the problem.

First, we provide a study on the value of the covering threshold σ. We show that the standard threshold considered in previous works, i.e., σ equal to the number of state-action pairs, can admit a trivial solution and thus is not sensible for the goal of the policy space compression. Next, we study a more reasonable range of σ values, and we provide tractable algorithms to check whether a given set of representative policies meets the covering requirement globally.

Then, we tackle the problem of identifying the number K of covering policies that are needed to solve the policy space compression problem. Especially, we provide lower and upper bounds to the value of K, which directly depends on the σ threshold and all the other relevant quantities of the problem. Those bounds on the number of covering policies can be used to speed up the iterative compression procedure starting from reasonable values of K.

1.2. Thesis Structure

The presented thesis is structured as follows.

In Chapter 2, we revise the background useful to understand the remainder of the thesis. We start introducing the fundamentals of RL (Section 2.1) including the multi-armed bandit problem, Markov decision processes, and dynamic programming. Then, we recap the traditional sample-based algorithms for RL (Section 2.2) and their extensions to function approximation (Section 2.3).

In Chapter 3, we present the state of the art of the policy space compression problem, as presented by the seminal paper [12]. We first provide a general formulation of the problem (Section 3.1), followed by its game-theoretic version (Section 3.2), and a principled iterative first-order method to address the problem (Section 3.3).

In Chapter 4, we study the role of the covering threshold in the policy space compression problem. First, we revise the geometry of the problem in Section 4.1. Next, we discuss how to set a covering threshold properly in Section 4.2. Finally, we provide optimization and algorithmic solutions to check the covering requirement given a covering threshold (Section 4.3).

In Chapter 5, we study the number of covering policies that are necessary to solve the policy space compression problem with a given covering threshold. First, we provide an illustrative example in a simplified polytopal space (Section 5.1). Next, we provide a few fundamental (Section 5.2) that we exploit to derive a lower and an upper bound to the number of covering policies, reported in Section 5.3 and Section 5.4 respectively.

In Chapter 6, we wrap-up the thesis content to conclude the dissertation.

2 | Background

In this chapter, we will cover the necessary background to understand this thesis. First, Section 2.1 introduces the fundamentals of reinforcement learning and Markov decision processes. Then, Section 2.2 dives into the traditional learning methods for reinforcement learning. Finally, Section 2.3 provides a brief introduction to function approximation and policy gradient methods.

2.1. Fundamentals of Reinforcement Learning

Reinforcement Learning (RL) [19] is a subfield of machine learning that focuses on how agents can learn to make sequences of decisions in order to maximize a cumulative reward signal. It is inspired by behavioral psychology, where learning is driven by trial and error interactions with an environment. RL has found applications in a wide range of fields, including robotics, game playing, autonomous systems, recommendation systems, and more.

To understand the fundamentals of reinforcement learning, let us consider the main actors involved in the process. The agent, or decision-maker, is the learner that interacts with the environment, which represents the system external to the agent. The environment can be a real-world system, like a physical space where a robot is navigating, or a simulator, like a computer game. Interacting with the environment, the agent can take actions within a discrete, or continuous, pre-defined set of options. The agent takes actions based on its current knowledge and the current state of the environment. The state contains all the relevant information to make decisions, and it comes from a discrete or continuous set of states, depending on the problem. Finally, the policy is the, deterministic or stochastic, mapping from states to actions that the agent follows.

A reward is a numerical signal that the environment provides to the agent after each action. It quantifies how good or bad the action was in that particular state. The agent's goal is to maximize the cumulative reward over time.

A trajectory (sometimes called an episode) is a sequence of states, actions, and rewards that an agent experiences interacting with the environment over a certain number of time steps. The value function estimates the expected cumulative reward an agent can collect starting from a given state and following a specific policy. It helps the agent evaluate the desirability of different states.

In the following sections, we provide mathematical formulations of the described interaction process. First, we warm up the reader presenting the multi-armed bandit model, which formalize a simplified, stateless, version of the interaction process. Then, we describe a generalization of the latter setting that is called Markov decision process, which will be a crucial model in the remainder of this thesis.

2.1.1. The Multi-Armed Bandit Problem

The Multi-Armed Bandit (MAB) [10] is a classical problem of sequential decision-making under uncertainty. In this problem, the agent is faced with a row of k actions, called arms, from which rewards are collected according to an unknown distribution that is specific to each arm. The agent's goal is to maximize the total reward over time while dealing with the uncertainty about which arm is the most rewarding. The model is named after a typical casino situation, in which each arm represents a slot machine (or bandit), and the decision-maker wants to maximize the payoffs collected from the slot machines. The problem can be formulated through these components:

- **Time Steps:**
  - The problem unfolds over a series of time steps, denoted as $t = 1, 2, 3, \ldots$.
- **Arms:**
  - There are k arms, denoted as $A_1, A_2, \ldots, A_k$;
  - The agent can choose one arm to pull at each time step;
- **Rewards:**
  - Each arm is associated with an unknown probability distribution over rewards.
  - The expected (mean) reward for pulling arm i is denoted as $Q^*(A_i)$, where i varies from 1 to k.
  - The actual reward received when pulling arm i at time step t is denoted as $R_t(A_i)$.

The goal is to maximize the expected cumulative reward over a finite time horizon T:

$$\sum_{t=1}^{T} R_t(A_{i_t}),$$

where i_t is the arm chosen at time t. The agent's estimate of the expected reward for arm i at time t is denoted as $Q_t(A_i)$.

Action Values in the k-Armed Bandit Problem

The action value $Q^*(A_i)$ represents the expected reward you would obtain on average by selecting arm A_i from the set of arms $A_1, A_2, \ldots, A_k$. It is defined as:

$$Q^*(A_i) = \mathbb{E}[R_t(A_i)],$$

where $R_t(A_i)$ is the actual reward obtained by pulling arm A_i at time step t, and $\mathbb{E}[\cdot]$ denotes the expected value.

We can compute the estimate of the mean reward incrementally as

$$Q(A_t) \leftarrow Q(A_t) + \frac{1}{N(A_t)} \left(R_t(A_t) - Q(A_t)\right)$$

where $Q(A_t)$ is the updated action value for arm A_t, $N(A_t)$ is the number of times arm A_t has been chosen up to time t, $R_t(A_t)$ is the reward obtained when arm A_t was chosen at time t. The latter incremental form allows for updating action values estimates as new rewards are observed. This method is memory-efficient and computationally efficient.

With the previous definitions, we can rewrite the learning objective as:

$$\max_{A_{i1},\ldots,A_{iT}} \sum_{t=1}^{T} \mathbb{E}[R_t(A_{i_t})],$$

where T is the number of time steps, $R_t(A_{i_t})$ is the actual reward obtained by pulling arm A_{i_t} at time step t, A_{i_t} represents the arm chosen at time step t.

In summary, the action value represents the expected reward for selecting a specific arm, and the goal is to select arms over time in a way that maximizes the total expected cumulative reward. This involves making a trade-off between exploring different arms to gather information about their rewards and exploiting arms that are currently estimated to have high rewards. The challenge lies in finding a balance between exploration and exploitation strategies to achieve the highest possible cumulative reward over the long term.

Exploration-Exploitation Trade-off

The central challenge in MAB problems is to balance exploration (trying different arms to learn about their mean rewards) and exploitation (choosing the arm that appears to be the best based on the current information). A strategy (policy) is needed to decide which arm to pull at each time step.

In the context of a k-armed bandit problem there exists several strategies to balance the exploration-exploitation trade-off:

Epsilon-Greedy Strategy The epsilon-greedy strategy balances exploration and exploitation by randomly selecting arms to explore with probability ϵ, and selecting the greedy arm, i.e., the one associated with the highest estimated mean reward with probability $1-\epsilon$. The action selection process is given by:

$$A_t = \begin{cases} \text{random arm} & \text{with probability } \epsilon \\ \text{argmax}_i(Q_t(A_i)) & \text{with probability } 1-\epsilon \end{cases}$$

Recall that $Q_t(A_i)$ is the estimated action value for arm A_i at time t.

Upper Confidence Bound Strategy The Upper Confidence Bound (UCB) [9] strategy encourages exploration by considering both the estimated action value and the uncertainty in that estimate. It selects the arm that maximizes the estimated reward plus an uncertainty-based exploration bonus. The action selection process is:

$$A_t = \text{argmax}_i \left(Q_t(A_i) + c\sqrt{\frac{\ln(t)}{N_t(A_i)}} \right)$$

where c is a parameter controlling the level of exploration, t is the current time step, and $N_t(A_i)$ is the number of times arm A_i has been chosen up to time t.

Thompson Sampling Strategy Thompson Sampling (TS) [21] is a Bayesian approach to address the exploration-exploitation dilemma in the k-armed bandit problem. It combines probability theory with decision-making to make informed choices.

The core idea behind TS is to maintain a probability distribution over the true action values of each arm. At each time step, the arm to be pulled is chosen according to their probability of being the best arm.

The action selection process involves sampling from the posterior distribution over the action values and choosing the arm with the highest sampled value:

$$A_t = \text{argmax}_i\, \text{Sample}(\text{Posterior}_i)$$

Here, Posterior_i represents the posterior distribution over the true action value of arm A_i, and $\text{Sample}(\text{Posterior}_i)$ denotes a sample drawn from that distribution.

At each step, the posterior distribution associated to the selected arm is updated according to the collected payoff.

TS leverages the principles of Bayesian inference to make decisions that are both exploratory and exploitative. It's a powerful strategy that adapts naturally to uncertainty and provides a principled way to address the exploration-exploitation trade-off.

2.1.2. Markov Decision Processes

Having introduced the MAB setting in the previous section, we now present a more general formulation, which also admits different states in the environment. The latter is the Markov Decision Process (MDP) [16], a mathematical framework used to model decision-making problems with stateful environments. The main components of the model are:

- **States:**

  A finite set $\mathcal{S}$ of all the possible configurations in the environment where the agent can find itself.

- **Actions:**

  A finite set $\mathcal{A}$ of all the possible decisions that the agent can make at each state.

- **Transition Model:**

  A function $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ that describes the probability of transitioning from one state to another given a specific action. It is often represented as $P(s'|s, a)$, indicating the probability of moving to state s' when taking action a in state s.

- **Rewards:**

  A function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that associates each state-action pair with a real number, representing the immediate reward the agent receives when taking action a in state s. This is typically represented as $R(s, a)$.

- **Horizon:**

  A constant $T \in \mathbb{N}$ denoting the length of a single interaction episode.

Every interaction episode between the agent and the environment in an MDP unfolds as follows. The agent starts in an initial state s_0.[1] At each time step t, the agent takes an action a_t based on the current state s_t. The environment responds by transitioning the agent to a new state s_{t+1} with probability $P^{a_t}_{s_t s_{t+1}}$ and provides a reward $R^{a_t}_{s_t}$. This process is repeated upon to the episode termination, which is handled differently in episodic or continuing tasks.

Episodic Tasks In this setting the horizon is finite $T < \infty$. Thus, for every time step t, we can define the *return* as the reward collected from step t to the end of the episode

$$G_t = R_{t+1} + R_{t+2} + \ldots + R_T.$$

Similarly, the *expected return* is defines as

$$v(s) = \mathbb{E}[G_t | S_t = s, \text{ episode ends at } T].$$

Continuing Tasks In this setting, the episode does not have a predefined terminal step, but the reward are weighted by an exponential *discount factor* $\gamma \in [0, 1]$, so that the reward collected in the time step t is weighted more than the rewards collected in future steps. Specifically, the *discounted return* from step t onward is computed as

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

Similarly, the *expected discounted return* is given by

$$v(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right].$$

In an MDP, the primary goal of the agent is to make a sequence of decisions to maximize the expected cumulative reward over time. This objective is achieved by selecting actions that lead to higher rewards. The component that controls the action selection strategy of the agent is called a *policy*, which is formalized below.

[1]Here we consider a single initial state as a simplifying assumption. Note that this is without loss of generality w.r.t. considering an initial state distribution.

Policies

A policy in an MDP is a strategy that guides the agent's decision-making by specifying which action to take in each state. It is defined through a mapping $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ from states to actions and it is a fundamental concept for achieving the agent's goal of maximizing the expected return (in episodic or continuing tasks). In particular, we differentiate between two classes of policies:

- **Deterministic Policies** that assign a single action to each state, meaning that for a given state s, the policy specifies exactly one action a, i.e.,
$$\pi(s) = a.$$
- **Stochastic Policies** that assign a probability distribution over actions for each state. This means that for a given state s, the policy specifies the probability of taking each possible action a, i.e.,
$$\pi(a|s) = \mathbb{P}(A_t = a|S_t = s).$$

Value Functions

The value function is the mathematical object that represent the long-term value of being in a particular state of the environment while following a policy π. Technically, the so called *state-value function* $v_\pi : \mathcal{S} \to \mathbb{R}$ represents the expected cumulative reward that an agent can obtain from a specific state s while following a given policy π, i.e.,

$$v_\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, \pi\right]$$

where R_{t+k+1} is the reward at time step $t+k+1$, γ is the discount factor, and the expectation is taken over all the possible sources of randomness, which include the transition model, reward function, and policy.

Similarly, the *action-value function* $q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the expected cumulative reward that an agent can obtain from taking action a in state s and then following policy π thereafter. Thus, it quantifies the value of taking a specific action in a specific state under a given policy. Technically, it is given by

$$q_\pi(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a, \pi\right]$$

where R_{t+k+1} is the reward at time step $t+k+1$, γ is the discount factor, and the expectation is taken over all the possible sources of randomness.

As we shall see later, both value functions are critical in Markov decision processes and reinforcement learning, as they help the agent make decisions by providing estimates of the expected cumulative rewards associated with different states and actions.

An important property of the value functions is that they admit alternative recursive definitions by means of the so-called Bellman equations.

State-Value Bellman Equation The Bellman equation for the state-value function expresses the value of a state s in terms of the immediate reward R_{t+1} obtained from transitioning to the next state s', the discount factor γ, and the expected value of the next state $v_\pi(s')$ as

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_\pi(s')\right]$$

where a represents an action, s' is the next state, r is the reward, $\pi(a|s)$ is the probability of taking action a in state s under policy π, and $p(s',r|s,a)$ is the probability of transitioning to state s' and receiving reward r when taking action a in state s.

Action-Value Bellman Equation The Bellman equation for the action-value function expresses the value of a state-action pair (s,a) in terms of the immediate reward R_{t+1} obtained from transitioning to the next state s', the discount factor γ, and the expected value of the next state-action pair $q_\pi(s',a')$ as

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a')\right]$$

where a' represents another possible action in the next state s', and all other symbols have the same meanings as in the state-value Bellman equation.

The Bellman equations are crucial recursive relationships in reinforcement learning. They provide a way to express the value of a state (or state-action pair) in terms of the rewards obtained from the current transition and the expected value of future states (or state-action pairs) under the given policy.

Solving the Bellman equations iteratively is a fundamental approach to finding optimal value functions, which in turn guide the agent's decision-making process to maximize the expected return.

Optimality

An *optimal policy* in an MDP is an action-selection strategy that maximizes the agent's expected cumulative reward over time. It guides the agent to make decisions that result in the highest possible long-term expected return.

For a given MDP, the optimal policy π^* is defined as

$$\pi^*(s) = \arg\max_{\pi} v_\pi(s)$$

where $v_\pi(s)$ is the state-value function corresponding to policy π, and $\arg\max_\pi$ represents the policy that maximizes the state-value function for each state s. The optimal policy for an MDP is not necessarily unique, which means that more than one policy can be optimal. An important result in MDPs [16] establishes that there always exists an optimal policy that is deterministic.

The *optimal state-value function* represents the maximum expected cumulative reward an agent can obtain from a specific state s under the optimal policy π^*, i.e.,

$$v_*(s) = \max_{\pi} v_\pi(s)$$

where $v_*(s)$ is the optimal value function for state s, and $\max_\pi$ represents the maximum value over all possible policies π. Intuitively, the optimal value function $v_*(s)$ quantifies the highest achievable value from being in state s under the optimal policy.

Similarly, one can define the *optimal action-value function* as

$$q_*(s, a) = \max_{\pi} q_\pi(s, a).$$

Notably, the optimal state-value function and action-value function admit a specific recursive relationship that is called the Bellman optimality equation.

The Bellman optimality equation for the state-value function is given by:

$$v_*(s) = \max_{a} \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma v_*(s')\right]$$

where a represents an action, s' is the next state, r is the reward, γ is the discount factor, and $p(s', r \mid s, a)$ is the transition probability.

This equation captures the idea that the optimal value of a state s is the maximum expected return achievable by selecting the best action a and then following the optimal policy from the next state s'. It forms the foundation for computing the optimal value function for each state.

Analogously, the Bellman optimality equation for the action-value function is given by:

$$q_*(s,a) = \sum_{s',r} p(s',r \mid s,a) \left[r + \gamma \max_{a'} q_*(s',a') \right]$$

where s' is the next state, r is the reward, γ is the discount factor, $\max_{a'}$ represents the maximum value over all possible actions in the next state s', and all other symbols have the same meanings as in the state-value Bellman equation.

This equation establishes a connection between the optimal value of a state-action pair and the maximum expected return achievable by taking action a and then following the optimal policy. It allows for the computation of optimal action values that guide the agent's decision-making.

The Bellman optimality equations provide a fundamental framework for finding optimal value functions for a given MDP. These equations enable agents to determine the best actions and policies to maximize expected cumulative rewards, forming the basis for various reinforcement learning algorithms, as we shall see in following sections.

The optimal state-value function $v_*(s)$ and the optimal policy $\pi^*(s)$ are closely related. The optimal policy is determined based on the optimal state-value function, and it provides the best action to take in each state to maximize the expected cumulative reward.

The optimal policy $\pi^*(s)$ is derived from the optimal state-value function $v_*(s)$ by selecting the action that maximizes the expression inside the Bellman equation:

$$\pi^*(s) = \arg\max_a \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_*(s') \right]$$

where a represents an action, s' is the next state, r is the reward, γ is the discount factor, and $p(s',r|s,a)$ is the probability of transitioning to state s' and receiving reward r when taking action a in state s.

The optimal action-value function $q_*(s,a)$ and the optimal policy $\pi^*(s)$ are also deeply connected. The optimal policy is guided by the optimal action-value function, as it suggests the best action to take in each state to maximize the expected cumulative reward.

The optimal policy $\pi^*(s)$ can be derived from the optimal action-value function $q_*(s,a)$ as follows:

$$\pi^*(s) = \arg\max_a q_*(s,a)$$

where a represents an action, and $\arg\max_a$ selects the action that maximizes the optimal action-value function for the given state s.

The relationships between the optimal state-value function, optimal action-value function, and the optimal policy are fundamental in reinforcement learning. The optimal policy is driven by these value functions, which capture the agent's long-term goals and guide its actions toward maximizing expected cumulative rewards. These connections are especially exploited in the desing of some popular algorithms for solving MDPs (i.e., computing an optimal policy) that we are going to present in the next section.

2.1.3. Dynamic Programming

In this section, we present a standard technique to solve an MDP, i.e., finding an optimal policy through computation, which goes under the name of *dynamic programming* [2]. To warm up the reader, we first present an algorithm to compute the value of a given (possibly sub-optimal) policy, which is called the *policy evaluation* problem. We then present two algorithms to compute an optimal policy: Policy Iteration and Value Iteration.

Iterative Policy Evaluation

Iterative Policy Evaluation is an algorithm used in reinforcement learning to estimate the state-value function ($v_\pi(s)$) for a given policy π. The goal is to find the expected cumulative reward starting from each state s and following policy π.

The iterative policy evaluation algorithm involves repeatedly updating the value of each state until it converges to the true state-value function $v_\pi(s)$. The update equation for each state s, which is inspired by the Bellman equation, is given by

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s')\right]$$

where

- $v_{k+1}(s)$ is the updated estimate of the state value for state s at iteration $k + 1$;
- $v_k(s)$ is the current estimate of the state value for state s at iteration k;
- $\pi(a|s)$ is the probability of taking action a in state s under policy π;
- $p(s', r|s, a)$ is the probability of transitioning to state s' and receiving reward r when taking action a in state s;
- γ is the discount factor.

The algorithm starts with an initial estimate of the state values ($v_0(s)$) and iteratively updates these estimates using the above equation until the values converge, which typically means they stop changing significantly between iterations. In short, the algorithm follows the three steps below:

1. Initialize $v_0(s)$ arbitrarily for all states s.
2. For each state s, update $v_{k+1}(s)$ using the update equation mentioned above.
3. Repeat until convergence (i.e., until $v_k(s)$ no longer changes significantly for all states s).

When the third step is completed, the algorithm outputs $v_k(s)$ of the last iteration.

Policy Iteration

Policy Iteration [2, 19] is an iterative algorithm to find the optimal policy for a given MDP. It alternates between two main steps, policy evaluation and policy improvement, until it converges to the optimal policy. Below, we describe Policy Iteration along with its mathematical expressions and formulations.

- **Policy Evaluation**

  In this step, the algorithm evaluates the current policy by iteratively estimating the state-value function (v_π) for that policy. The state-value function is calculated with the update rule

  $$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_\pi(s')\right]$$

  where a represents an action, s' is the next state, r is the reward, γ is the discount factor, $\pi(a|s)$ is the probability of taking action a in state s under the current policy, and $p(s',r|s,a)$ is the probability of transitioning to state s' and receiving reward r when taking action a in state s. Policy Evaluation continues until the change in the estimated state values is smaller than a predefined threshold.

- **Policy Improvement**

  Once the policy has been evaluated, the algorithm improves the policy by selecting actions that maximize the expected value of the state-action pairs, often called the *greedy* action. The new policy π' is defined as:

  $$\pi'(s) = \arg\max_a \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_\pi(s')\right]$$

where $\arg\max_a$ selects the action that maximizes the expression inside the brackets.

Policy Iteration iterates between these two steps until the policy no longer changes, which indicates that the optimal policy has been found. The algorithm guarantees convergence to the optimal policy thanks to the Policy Improvement Theorem (see [19]).

Value Iteration

Value Iteration is an iterative algorithm used to find the optimal value function of an MDP, from which the optimal policy can be recovered. It is a specific instance of Generalized Policy Iteration that focuses on updating the value function iteratively. Below a detailed explanation, including mathematical expressions and formulations:

1. **Initialization**

   Start with an initial value function $V_0(s)$, often initialized to zeros or any other arbitrary values.

2. **Loop**

   Update the value function $V_k(s)$ for each state s applying the Bellman optimality equation:

$$V_{k+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V_k(s')].$$

   Repeat until the change in V_k across all states is smaller than a predefined threshold or for a fixed number of iterations.

3. **Termination**

   Once the value function V has converged, derive the optimal policy $\pi^*(s)$ by selecting actions that maximize the equation inside the Bellman optimality equation:

$$\pi^*(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V^*(s')].$$

Value Iteration guarantees convergence to the optimal value function and policy in a finite number of iterations. The core idea is to iteratively improve the estimate of the optimal value function by applying the Bellman optimality equation. Once the value function has converged, the optimal policy can be easily extracted.

This algorithm is particularly useful when you are primarily interested in finding the optimal policy, as it does not require explicit representations of the intermediate policies and only focuses on the value function.

2.2. Sample-Based Methods for RL

In the previous sections, we considered algorithms that require full knowledge of the MDP in order to compute the optimal policy. In this section, we instead present algorithms that compute approximately optimal policies only relying on samples coming from the true MDP, which is unknown throughout the process. This is called the RL problem, for which we provide some traditional algorithms. Especially, we present two different approaches for sample-based policy evaluation: Monte Carlo methods (Section 2.2.1) and temporal difference learning (Section 2.2.2). Then, we provide control algorithms based on temporal difference in Section 2.2.3.

Sample-based learning methods can be further categorized as on-policy or off-policy. On-policy methods update the policy being used for collecting the samples, while off-policy methods update a different policy based on the observed data. We will provide instances of both classes below, and more details in Section 2.2.4.

Finally, sample-based learning methods are versatile and applicable to a wide range of RL problems. They are particularly useful when dealing with complex environments where modeling the dynamics is challenging or when the state and action spaces are high-dimensional. By collecting and learning from samples, these methods enable agents to improve their decision-making over time through interaction with the environment.

2.2.1. Monte Carlo Methods

Monte Carlo methods are sample-based techniques to estimate value functions by simulating episodes of interactions with the environment and taking sample averages of the collected rewards.

Monte Carlo Action-Value Estimation

The objective is to estimate the action-value function q_π for a given policy π. The process follows the steps below.

1. **Episode Generation**

   Generate one or more episodes by following the policy π from the initial state s_0 until a terminal state is reached. An episode is represented as:

$$(s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T).$$

2. **Estimating Returns for State-Action Pairs**

   For each state-action pair (s_t, a_t), estimate the return G_t as the cumulative discounted reward from time step t until the end of the episode:

   $$G_t = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{T-t-1} r_T.$$

3. **Update Action-Value Estimates**

   Use the returns G_t to update the estimate of the action-value function $q_\pi(s_t, a_t)$ for each state-action pair (s_t, a_t) encountered in the episode. This can be done using the sample average:

   $$q_\pi(s_t, a_t) \leftarrow \frac{1}{N(s_t, a_t)} \sum_{i=1}^{N(s_t,a_t)} G_t^{(i)}$$

   where $N(s_t, a_t)$ is the number of times the state-action pair (s_t, a_t) has been visited in the episode, and $G_t^{(i)}$ is the i-th return observed for the pair (s_t, a_t) in different episodes.

4. **Repeat Steps 1-4**

   Repeat the process by generating more episodes and refining the estimates of q_π until convergence.

Monte Carlo methods for action-value functions estimate the expected cumulative rewards of taking specific actions in specific states under a given policy by sampling episodes and averaging the returns. These methods are suitable for situations where the dynamics of the environment are unknown or when dealing with large state and action spaces. Like state-value estimation, Monte Carlo methods for action-value functions require a sufficient number of samples to converge to accurate estimates.

2.2.2. Temporal Difference Learning

Temporal Difference (TD) learning is a reinforcement learning technique used to estimate value functions and compute approximately optimal policies in MDPs. It combines elements of dynamic programming with sample-based methods.

Temporal Difference Learning State-Value Estimation

Let us focus on TD(0) for state-value estimation, which is a specific variant of TD learning (see [19] for a comprehensive survey). In TD(0), the estimate for each state is updated based on the current estimate and a single-step transition. The update equation for TD(0) is as follows:

$$V(s_t) \leftarrow V(s_t) + \alpha \left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$$

where

- $V(s_t)$ is the estimated value of state s_t;
- α is the learning rate, controlling the size of the update;
- r_{t+1} is the reward received after taking action a_t in state s_t and transitioning to state s_{t+1};
- γ is the discount factor;
- $V(s_{t+1})$ is the estimated value of the next state s_{t+1}.

Specifically, the algorithm follows the subsequent steps.

1. Initialize the value function $V(s)$ for all states.
2. For each episode:
   (a) Initialize the starting state s_t.
   (b) Repeat until the episode terminates:
      i. Take an action a_t following the policy π.
      ii. Observe the reward r_{t+1} and the next state s_{t+1}.
      iii. Update the value estimate for state s_t using the TD(0) update rule.
      iv. Set s_t to s_{t+1}.
3. Repeat the above process for multiple episodes or until convergence.

In summary, TD learning is a reinforcement learning technique used to estimate value functions by iteratively updating value estimates based on observed transitions and rewards. TD(0) and Q-learning are common variants of TD learning used for state-value and action-value estimation, respectively. These methods are model-free and suitable for scenarios where learning occurs through interaction with the environment.

2.2.3. Temporal Difference Learning for Control

Temporal difference can be also used for control tasks, i.e., for computing an approximately optimal policy. TD methods combine elements of both model-free and model-based approaches by updating value estimates based on the difference between the current estimated value and the estimated value of the next state.

SARSA

SARSA (State-Action-Reward-State-Action) [19] is an on-policy reinforcement learning algorithm based on TD learning. The SARSA update rule is used to estimate action values (q_π) for a given policy π, similarly as in the previous section:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)$$

where

- $Q(s_t, a_t)$ is the estimated action-value for taking action a_t in state s_t;
- α is the learning rate, controlling the size of the update;
- r_{t+1} is the reward received after taking action a_t in state s_t and transitioning to state s_{t+1};
- γ is the discount factor;
- $Q(s_{t+1}, a_{t+1})$ is the estimated action-value for the next action a_{t+1} in the next state s_{t+1} following the current policy π.

Specifically, SARSA follows the algorithmic steps below

1. Initialize the action-value function $Q(s, a)$ arbitrarily for all state-action pairs.
2. Choose an initial state s_0 and an initial action a_0 based on the current policy π.
3. Repeat until termination:
   (a) Take action a_t in state s_t based on the current policy π.
   (b) Observe the reward r_{t+1} and the next state s_{t+1}.
   (c) Choose the next action a_{t+1} based on the current policy π.
   (d) Update the action-value estimate for the current state-action pair (s_t, a_t) using the SARSA update equation.
   (e) Set s_t to s_{t+1} and a_t to a_{t+1}.
4. Repeat the above process for multiple episodes or until convergence.

Expected SARSA Expected SARSA slightly modifies the algorithm above by considering an update rule as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \sum_a \pi(a|s_{t+1}) Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

where the action at the next step is on average over the policy rather than a single realization from the policy.

Policy Improvement in SARSA SARSA typically uses an ϵ-greedy policy to handle the exploration-exploitation trade-off. After each episode or a certain number of time steps, the policy π is updated to be ϵ-greedy with respect to the current action-value estimates $Q(s, a)$.

In summary, SARSA is an on-policy reinforcement learning algorithm used to estimate action values (q_π) and find an optimal policy in MDPs. It updates action-value estimates based on observed transitions and rewards while following the current policy π. The policy is often improved by making it ϵ-greedy. SARSA is particularly suitable for scenarios where learning occurs through interactions with the environment.

Q-Learning

Q-Learning [22] is an off-policy reinforcement learning algorithm based on TD value estimates. The update rule for Q-Learning is used to estimate action values (q_π) for an optimal policy directly, taking inspiration from value iteration and the Bellman optimality equation. The update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$$

where

- $Q(s_t, a_t)$ is the estimated action-value for taking action a_t in state s_t;
- α is the learning rate, controlling the size of the update;
- r_{t+1} is the reward received after taking action a_t in state s_t and transitioning to state s_{t+1};
- γ is the discount factor;
- $\max_{a'} Q(s_{t+1}, a')$ is the maximum estimated action-value for the next state s_{t+1} over all possible actions a'.

The Q-Learning algorithm follows the algorithmic steps below.

1. Initialize the action-value function $Q(s, a)$ arbitrarily for all state-action pairs.
2. Choose an initial state s_0.
3. Repeat until termination:
    (a) Select an action a_t for the current state s_t based on the current policy (often ϵ-greedy with respect to $Q(s_t, a)$).
    (b) Execute action a_t in state s_t.
    (c) Observe the reward r_{t+1} and the next state s_{t+1}.
    (d) Update the action-value estimate for the current state-action pair (s_t, a_t) using the Q-Learning update equation.
    (e) Set s_t to s_{t+1}.

Policy Improvement in Q-Learning Similarly as SARSA, Q-Learning can be used with an ϵ-greedy to handle exploration and exploitation. However, being Q-Learning an off-policy algorithm, the policy to be deployed to collect samples does not have to be the current greedy policy, but any policy of choice.

In summary, Q-Learning is an off-policy reinforcement learning algorithm used to estimate action values (q_π) and find an optimal policy in MDPs. It updates action-value estimates based on observed transitions and rewards while using an ϵ-greedy policy for exploration and exploitation. Q-Learning is particularly suitable for scenarios where learning occurs through interaction with the environment, and exploration is crucial for discovering the optimal policy.

2.2.4. Off-Policy Learning and Importance Sampling

Off-policy learning is a reinforcement learning paradigm where an agent learns and evaluates a target policy (π) based on data collected from a different behavior policy (μ). It's especially valuable when you want to evaluate or improve a policy using historical data that was generated by another policy.

A classical approach to off-policy learning is importance sampling. Importance sampling [14] can be used in off-policy reinforcement learning to account for the mismatch between the target policy and the one collecting the data. This is done through a re-weighting with the importance sampling ratio (ρ). For a sequence of actions $a_1, a_2, \ldots, a_n$,

ρ is given by:

$$\rho = \frac{P(\text{sampled actions under target policy})}{P(\text{sampled actions under behavior policy})} = \frac{\pi(a_1|s_1) \cdot \pi(a_2|s_2) \cdot \ldots \cdot \pi(a_n|s_n)}{\mu(a_1|s_1) \cdot \mu(a_2|s_2) \cdot \ldots \cdot \mu(a_n|s_n)}$$

where

- $\pi(a|s)$ is the probability of taking action a in state s under the target policy;
- $\mu(a|s)$ is the probability of taking action a in state s under the behavior policy.

In summary, importance sampling is a crucial technique in off-policy reinforcement learning. It allows agents to estimate expected values and evaluate or improve policies under a target policy using data collected from a behavior policy. This technique is widely used in various reinforcement learning algorithms to handle situations where policies differ during learning and execution.

2.3. Function Approximation for RL

The methods we have presented so far rely on a crucial assumption: That the number of states and actions is finite, and possibly small, such that policies and value functions can all be represented through tables (and thus they are called *tabular* methods). When dealing with large or continuous state spaces and/or action spaces, it is often impractical to maintain a tabular representation of the value function and the policy. Instead, function approximation techniques [19] are used to approximate the value function.

One possibility is to use simple linear models trained through linear regression, in which a set of features (state or state-action features) is used instead of the state (action) index, and linear combinations of these features are learned to approximate the values. However, in many real-world scenarios, value functions are highly non-linear. Non-linear function approximation methods, such as artificial neural networks, can be used to approximate the value function more accurately. Deep learning techniques, including deep neural networks, are particularly powerful for capturing complex relationships.

Once we have a good approximator for the value function, one can use it to perform both policy evaluation and control tasks, which involve finding an optimal policy that maximizes the expected cumulative rewards. Actually, the control task often includes policy evaluation as an inner step.

Using function approximation in reinforcement learning comes with several challenges like controlling approximation errors, stability, as well as the common exploration-exploitation trade-off. To address these challenges, various reinforcement learning algorithms have been developed. We will present a few of them in the next sections. First, we will cover policy evaluation (a.k.a. prediction problem) in Section 2.3.1. Then, we will cover control tasks in Section 2.3.2.

2.3.1. Prediction with Function Approximation

Prediction with function approximation in reinforcement learning involves estimating the value functions using parameterized models.

Parameterizing Value Functions

Parameterizing value functions involves using function approximation techniques to represent these functions. This means replacing the explicit storage of values for each state or state-action pair with a parameterized mathematical function or model. These functions can be linear, non-linear (e.g., neural networks), or other forms that efficiently estimate values.

- **State-Value Function Parameterization**

  For state-value functions $v_\pi(s)$, parameterization means using a function $V(s;\theta_v)$ with parameters θ_v to estimate the value of each state s. This parameterized function takes the state s as input and provides an estimate of $v_\pi(s)$.

- **Action-Value Function Parameterization:**

  For action-value functions $q_\pi(s,a)$, parameterization involves using a function $Q(s,a;\theta_q)$ with parameters θ_q to estimate the value of taking action a in state s. This parameterized function takes both the state s and action a as inputs and estimates $q_\pi(s,a)$.

Using those parameterized models counts several benefits, which are listed below.

- **Generalization**

  Parameterized value functions have the ability to generalize across similar states or state-action pairs. This means that if the agent learns the value of one state or action, it can apply this knowledge to other similar states or actions. This is particularly valuable in high-dimensional or continuous state spaces.

- **Compact Representation**

  Instead of explicitly storing values for every state or state-action pair, parameterized value functions provide a compact representation. This makes it feasible to work with large state spaces where storing values explicitly would be impractical.

- **Learning from Data**

  Parameters θ in parameterized functions are learned from interactions with the environment. This adaptability allows the agent to learn and improve its value estimates as it interacts with different tasks and environments. It is then a data-driven approach.

- **Transferability**

  Knowledge gained from learning in one environment can be transferred to related environments. If the agent learns the value of certain states or actions in one context, it can apply this knowledge effectively in similar contexts, speeding up learning.

- **Scalability**

  Parameterization is scalable to complex problems with large state spaces. It enables reinforcement learning algorithms to handle situations where the number of states is huge, which would be infeasible with traditional tabular methods.

- **Flexibility in Function Approximation**

  Parameterization allows flexibility in choosing function approximation techniques. You can use linear models, nonlinear models like neural networks, or other methods depending on the problem's characteristics.

Having selected the model parameterization, the problem is then to define how to learn the parameters from data. As we shall see, the data distribution μ plays a crucial role on the quality of the estimation we can expect. In the following, we present value function estimation through supervised learning, gradient Monte Carlo methods, and semi-gradient temporal difference.

Supervised Learning

The problem of learning the parameters θ can be cast as a traditional supervised learning problem. In this setting, we have a parameterized function $v(s;\theta)$ and a dataset of state-value pairs $(s_1, v_\pi(s_1)), (s_2, v_\pi(s_2)), \ldots, (s_n, v_\pi(s_n))$. Then, we can use the Mean Squared Error (MSE) as the loss function of a corresponding regression problem, but now we incorporate the data distribution $\mu(s)$ into the loss:

$$\mathrm{MSE}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mu(s_i) \cdot (v(s_i;\theta) - v_\pi(s_i))^2.$$

Basically, we adjust the model parameters (θ) to fit the data considering the weighted contribution of each state based on $\mu(s_i)$. This simple procedures presents some challenges, listed below.

- **Lack of Exploration**

  Supervised learning relies on a fixed dataset, which may not cover the full range of state-action pairs encountered during actual policy execution. This can lead to biased estimates and an inability to generalize well to unvisited states.

- **Non-Stationarity**

  In reinforcement learning, the environment may change over time. Supervised learning assumes a fixed dataset, making it challenging to adapt to changes in the policy or environment.

- **Sample Efficiency**

  Supervised learning often requires a large amount of data to generalize effectively, which can be impractical in some real-world applications.

- **Distribution Mismatch**

  If the state visitation distribution (μ) used during training does not match the distribution under the policy being learned, the learned value function may not be accurate.

Despite these challenges, supervised learning for value function estimation, when applied carefully and with consideration of state visitation, can be a valuable tool in reinforcement learning.

Gradient Monte Carlo

An alternative to traditional supervised learning are instead Gradient Monte Carlo (GMC) methods [19]. The objective of GMC is to estimate the gradient of the mean squared value error w.r.t. θ. Specifically, the objective is:

$$J(\theta) = \frac{1}{2} \sum_{t=0}^{T} \mu(s_t) \left(G_t - \hat{V}(s_t; \theta) \right)^2$$

where

- $J(\theta)$ is the mean squared value error objective;
- θ represents the value function parameters to be learned;

- G_t is the estimated return (sum of rewards) for trajectory t;
- $\hat{V}(s_t;\theta)$ is the estimated value of state s_t using the current parameterization θ;
- $\mu(s_t)$ is the fraction of time spent in state s_t following the policy.

The GMC algorithm involves the following steps:

1. **Sample Trajectories**

   Generate multiple trajectories by following the current policy. Each trajectory consists of states, actions, and rewards. These trajectories are used to estimate returns G_t and state visitation probabilities $\mu(s_t)$.

2. **Compute Returns**

   For each time step t in the trajectories, compute the return G_t, which is the sum of rewards from time t onward:
   $$G_t = \sum_{k=t}^{T} R_k$$
   where R_k is the reward at time step k in the trajectory.

3. **Compute State Visitation Frequencies**

   Calculate the state visitation frequencies $\mu(s_t)$ based on the sampled trajectories. These frequencies represent the fraction of time spent in each state s_t following the policy.

4. **Gradient Estimation**

   Compute the gradient of the mean squared value error objective with respect to the value function parameters θ. The gradient is estimated using the returns G_t, the estimated values $\hat{V}(s_t;\theta)$, and the state visitation frequencies $\mu(s_t)$:
   $$\nabla_\theta J(\theta) = -\sum_{t=0}^{T} \mu(s_t)\left(G_t - \hat{V}(s_t;\theta)\right)\nabla_\theta \hat{V}(s_t;\theta)$$

5. **Gradient Update**

   Update the value function parameters θ using gradient descent or a related optimization method to minimize the mean squared value error objective
   $$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

where α is the learning rate.

6. **Repeat**

   Repeat the above steps for multiple iterations to iteratively improve the value function parameterization.

GMC combines Monte Carlo sampling, state visitation probability estimation, and gradient descent to update the value function parameters. It allows the agent to learn an accurate value function to improve policy optimization in reinforcement learning tasks, even when the true value function is unknown.

Semi-Gradient Temporal Difference

Semi-Gradient TD [19] is a method used to estimate the value function in reinforcement learning. It combines elements of TD learning with gradient methods for policy evaluation.

At each time step t, you calculate the TD error (δ_t), which represents the difference between the estimated value of the current state ($V(s_t)$) and the estimated value of the next state ($V(s_{t+1})$) plus the immediate reward (R_{t+1}):

$$\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

The policy parameters (θ) are updated using a semi-gradient method:

$$\theta \leftarrow \theta + \alpha \nabla_\theta V(s_t)\delta_t$$

where α is the learning rate, $\nabla_\theta V(s_t)$ is the gradient of the value function with respect to the policy parameters, and δ_t is the TD error at time t.

Semi-Gradient TD is used to estimate the value function and it is typical in value-based reinforcement learning. It provides a way to update policy parameters in a way that improves the policy performance.

2.3.2. Control with Function Approximation

Control with function approximation in reinforcement learning is the process of finding an optimal policy relying on a class of parameterized models (for the value function or the policy itself).

SARSA with Function Approximation

The SARSA algorithm that we presented in the previous section can be easily extend to function approximation. Instead of explicitly storing values for each state-action pair, we use a parameterized function $Q(s, a; \theta)$ to estimate $q_\pi(s, a)$, such that

$$q_\pi(s, a) \approx Q(s, a; \theta).$$

The update rules becomes

$$\theta \leftarrow \theta + \alpha \cdot \delta \cdot \nabla Q(s, a; \theta)$$

where α is the learning rate, δ is the temporal difference error. The latter is defined as

$$\delta = r + \gamma Q(s', a'; \theta) - Q(s, a; \theta).$$

The policy π can be updated by computing the policy improvement step just like in standard SARSA, such as taking the ϵ-greedy policy based on the estimated action-values from $Q(s, a; \theta)$.

The algorithm follows the algorithmic steps below.

1. Initialize the parameterized action-value function $Q(s, a; \theta)$ with random or predefined values.
2. Repeat episodes:
   (a) Initialize the starting state s.
   (b) Choose an action a using the policy π derived from $Q(s, a; \theta)$.
   (c) Repeat until the episode terminates:
      i. Take action a.
      ii. Observe the reward r and the next state s'.
      iii. Choose the next action a' using the policy π derived from $Q(s', a'; \theta)$.
      iv. Calculate the temporal difference error δ and update θ using the update rule.
      v. Set s to s' and a to a'.
   (d) If the episode terminates, update the policy π based on the updated $Q(s, a; \theta)$ using an exploration strategy.

3. Repeat until $Q(s, a; \theta)$ converges or for a predefined number of episodes.

Just like standard SARSA, this version working with function approximation can be modified with the expected SARSA update rule.

2.3.3. Policy Gradient

Policy Gradient (PG) [15] is a reinforcement learning technique that focuses on directly learning the policy function to optimize the agent behavior. It aims to find the policy parameters that maximize expected cumulative rewards by computing and then following the gradient of the expected return with respect to the policy parameters.

In PG methods, the objective is to maximize the expected return $J(\theta)$ over all possible trajectories:

$$J(\theta) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t]$$

where

- $J(\theta)$ is the expected return.
- γ is the discount factor.
- R_t is the reward at time step t.

The policy is parameterized as $\pi(a|s;\theta)$, where θ represents the policy parameters.

The Policy Gradient Theorem [20] provides the gradient of the expected return with respect to the policy parameters:

$$\nabla J(\theta) \propto \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla \log \pi(a_t|s_t;\theta) \cdot Q^{\pi}(s_t, a_t)\right]$$

where

- $\nabla J(\theta)$ is the gradient of the expected return.
- $\nabla \log \pi(a_t|s_t;\theta)$ is the gradient of the log-probability of taking action a_t in state s_t with respect to θ.
- $Q^{\pi}(s_t, a_t)$ is the action-value function representing the expected return starting from state s_t, taking action a_t, and following the policy π.

To derive the policy gradient, we start with the expression for the expected return:

$$J(\theta) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t]$$

To make gradient calculations tractable, we introduce a state distribution $d^{\pi}(s)$ that represents the probability of being in state s under the policy π. Now, we can express the expected return as:

$$J(\theta) = \sum_{s} d^{\pi}(s) \sum_{a} \pi(a|s;\theta) \sum_{t=0}^{\infty} \gamma^t R_t$$

Next, we take the gradient with respect to θ:

$$\nabla J(\theta) = \sum_{s} d^{\pi}(s) \sum_{a} \nabla\pi(a|s;\theta) \sum_{t=0}^{\infty} \gamma^t R_t$$

Using the property that the gradient and summation can be exchanged, we have:

$$\nabla J(\theta) = \sum_{s} d^{\pi}(s) \sum_{a} \nabla\pi(a|s;\theta) Q^{\pi}(s,a)$$

This is the policy gradient expression, and it tells us how to update the policy parameters to maximize the expected return.

Then, being α a learning rate, the policy parameters θ are updated in the direction of the policy gradient to maximize the expected return:

$$\theta \leftarrow \theta + \alpha \nabla J(\theta).$$

This process iteratively refines the policy to find the optimal policy that maximizes expected returns.

In summary, PG methods optimize policies directly by adjusting their parameters to maximize expected returns. The Policy Gradient Theorem provides a fundamental equation for computing policy gradients, and this gradient is used to update the policy in the direction of higher expected returns.

Actor-Critic Algorithm

The Actor-Critic (AC) algorithms [8] combine policy learning (the actor) and value function learning (the critic) to solve the control task. A typical AC algorithm follows the steps below.

1. Initialize policy parameters θ_{actor} and value function parameters θ_{critic}.
2. Repeat for each episode:
   (a) Initialize the environment and state s_0.
   (b) Repeat for each time step:
      i. **Actor (Policy Network)**:
         - Sample an action a_t from the policy $\pi(a_t|s_t;\theta_{\text{actor}})$.
         - Observe the reward r_t and the next state s_{t+1}.

      ii. **Critic (Value Network)**:
         - Calculate the TD error:

$$\delta_t = r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)$$

         - Update the value function using the TD error:

$$\theta_{\text{critic}} \leftarrow \theta_{\text{critic}} + \beta\delta_t \nabla Q^\pi(s_t, a_t; \theta_{\text{critic}})$$

      iii. **Actor (Policy Network)**: Update the policy using the policy gradient:

$$\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} + \alpha \nabla \log \pi(a_t|s_t;\theta_{\text{actor}}) \cdot Q^\pi(s_t, a_t)$$

3. Repeat until convergence.

The Actor (Policy Network) updates the policy parameters θ_{actor} using the Policy Gradient Theorem. It adjusts the policy in the direction that increases the expected return, guided by the action-value function $Q^\pi(s_t, a_t)$. The Critic (Value Network) updates the value function parameters θ_{critic} using the TD error. It learns to estimate the action-value function $Q^\pi(s_t, a_t; \theta_{\text{critic}})$ by minimizing the TD error. α and β are learning rates for the actor and critic updates, respectively.

The latter AC algorithm iteratively improves both the policy and the value function, enabling more effective reinforcement learning in complex environments.

3 | Policy Space Compression

In this chapter, we introduce the fundamental problem we will focus on this thesis, which is the *policy space compression*, introduced in [12]. In this problem, given an MDP and an (infinite) parametric policy space, we aim to identify a compact subset of the latter space that retains most of its expressive power in terms of addressing RL problems in the given MDP. As we shall see, solving the policy space compression problem have important implications for improving the efficiency of RL, but it brings arduous computational challenges, for which research is still underway. Before addressing those challenges from a novel geometric perspective on the problem, we provide the basics of policy space compression as studied in [12]. In Section 3.1, we give the formal definition of the problem and its main components. Then, we provide a game-theoretic formulation in Section 3.2, which comes as inspiration for the first policy space compression algorithm (Section 3.3).

3.1. Problem Formulation

In the policy space compression problem, we are given an MDP and a parametric policy space Θ. The latter represents the space of all the possible policies that an agent can choose from. Each policy $\boldsymbol{\theta}$ in this space defines a strategy that dictates the agent's actions based on its observations from the environment. The policy space can be infinite, encompassing a wide range of possible strategies. For instance, policies can be represented by parameters in a combination of linear functions or the weights of a deep neural network.

In policy space compression, the primary goal is to find a smaller subset of policies, denoted as Θ_σ and called σ-compression, within the infinite space Θ. This subset Θ_σ should have the following properties:

- It should retain most of the expressive power of the original policy space Θ, in terms of the state-action distributions the agent can induce over the given MDP;
- It should be sufficiently small to improve the efficiency of reinforcement learning with the reduced policy space.

The latter two desiderata can be incorporated into a single optimization problem in which the objective function asks to minimize the number of policies in Θ_σ, but under a coverage constraint that guarantees *coverage* of all the state dsitributions induced by the policies in the original policy space Θ. Formally, the optimization problem is:

$$\begin{aligned} \text{Minimize:} \quad & \sum_{\omega \in \Omega_\Theta} x_\omega \\ \text{Subject to:} \quad & \sum_{\omega : D_2(v \| \omega) \le \sigma} x_\omega \ge 1, \quad \forall v \in \Omega_\Theta \end{aligned}$$

where Ω_Θ represents the set of all state-action distributions that can be induced by policies in Θ, x_ω is a binary variable denoting whether a policy inducing the state-action distribution ω is in Θ_σ or not, and $D_2(v\|\omega)$ denotes the Rènyi divergence between the state-action distributions v and ω.

Complexity Unfortunately, previous works have shown that solving the problem above is NP-hard [12]. This implies that finding an exact optimal solution efficiently is computationally challenging, especially as the size and complexity of the policy space grow. Indeed, it is well-known that NP-hard problems do not admit polynomial time algorithms.

Instead of attempting to solve the NP-hard problem directly, researchers are exploring alternative approaches to make it computationally tractable. These approaches may involve approximations, heuristics, or insights from related areas to find a feasible solution that balances the trade-off between policy space size and coverage of state-action distributions.

3.2. Game-Theoretic Perspective

In this Section, we present a more handy formulation of the policy space compression problem, by adopting a game-theoretic approach. Thus, we introduce a two-player game where:

- Player 1 (referred to as the *leader*) selects a set of K policies denoted as $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K$. The goal of the leader is to cover a set of state-action distributions denoted as Ω_Θ;
- Player 2 (referred to as the *follower*) tries to find a policy $\boldsymbol{\mu}$ that is not adequately covered by the policies chosen by the leader.

As in the previous Section, the coverage is formulated in terms of the Rènyi divergence between distributions induced by the leader's policies and the follower's policy.

Formally, the game can be written as

$$\min_{\boldsymbol{\theta}\in\Theta^K} \max_{\boldsymbol{\mu}\in\boldsymbol{\Theta}} f(\boldsymbol{\theta}, \boldsymbol{\mu})$$
$$\text{such that } f(\boldsymbol{\theta}, \boldsymbol{\mu}) = \min_{k\in[K]} D_2(d^{sa}_{\boldsymbol{\mu}} || d^{sa}_{\boldsymbol{\theta}_k})$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$ is the set of leader's policies, $\boldsymbol{\mu}$ is the follower's policy, and $d^{sa}_{\boldsymbol{\mu}}$, $d^{sa}_{\boldsymbol{\theta}_k}$ denote their corresponding state-action distribution, while $[K]$ stands for the set of integers $\{1, \dots, K\}$.

The objective function $f(\boldsymbol{\theta}, \boldsymbol{\mu})$ is designed to capture the essence of the game. It represents the minimal Rènyi divergence between the state-action distribution of the follower's policy $\boldsymbol{\mu}$ and that of the leader's policies $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K$.

The intuitive goal of this game formulation is to iteratively find the follower's policy $\boldsymbol{\mu}$ that is the hardest for the leader to cover effectively, and then to adjust the leader's policy to cover the latter.

Complexity Unfortunately, also the optimization problem resulting from the game-theoretic perspective is computationally hard (see [12]), as it is neither convex nor concave.

Given the computational complexity of finding a global optimum, the focus shifts to finding a locally optimal solution. Specifically, the aim is to find a solution for which the function f does not change in any direction for the leader's strategies (local maximum) and does not change in any direction for the follower's strategies (local minimum).

3.2.1. Differential Stackelberg Equilibrium

The proposed solution concept is called the Differential Stackelberg Equilibrium (DSE) [5]. A DSE is a point in the objective function where both players have settled on strategies that are locally optimal in the sense that they meet conditions below:

- The leader's strategy should not change when considering slight variations in the follower's strategy;
- Similarly, the follower's strategy should be at a minimum and remain stable when the leader's strategy varies slightly;
- The determinant of the Hessian matrix (a matrix of second-order derivatives) should be positive for the leader's strategy, indicating that it is a local maximum;
- The determinant of the Hessian matrix should be negative (but close to zero) for

the follower's strategy, indicating that it is a local minimum.

In essence, a DSE represents a strategic balance between the leader and the follower, where neither can unilaterally improve their position. Recent research [4–6] has shown that a specific optimization technique called Gradient Descent Ascent (GDA) can be employed to find a DSE of a given game efficiently. GDA is an iterative method where the leader and the follower adjust their strategies following their respective gradients of the objective function. The crucial insight here is that, under certain conditions on the timescale separation, i.e., on the learning rate of the gradients' update of the leader and the follower, GDA is guaranteed to converge to a DSE [4].

3.2.2. Challenges

Unfortunately, there are still a few standing challenges to apply GDA to solve the policy space compression problem. Specifically,

- Ensuring that the value $f(\boldsymbol{\theta}^*, \boldsymbol{\mu}^*)$ obtained from a DSE guarantees that the set of policies $\boldsymbol{\theta}$ selected by the leader represents a valid σ-compression of the policy space Θ. This means that the leader's strategies effectively cover the state-action distribution space as required for a compression.
- Determining an appropriate value for the parameter K, which represents the number of policies chosen by the leader. The appropriate value of K is not obvious beforehand, and finding a balance between computational complexity and compression quality is a challenge.

To (partially) address those challenges, an algorithm have been proposed by [12], which we present in the next section. Those challenges will also be the focus of the thesis, for which we provide novel results coming from the geometric study on the problem in the remaining chapters.

3.3. The PSCA Algorithm

To address the challenges mentioned above, Mutti et al. [12] suggest an iterative first-order method. This method iteratively finds DSEs for larger instances of the game, referred to as the cover game. The goal is to continue this process until a conservative approximation of the desired global condition, i.e., $\max_{\boldsymbol{\mu}\in\Theta} f(\boldsymbol{\theta}^*, \boldsymbol{\mu}) \leq \sigma$, is met.

The algorithm starts with a smaller K and iteratively increases it until the game outputs a satisfactory solution. In this way, it does not require a predefined fixed value for K, and it ensures that the policies selected by the leader effectively compress the policy space to meet the σ-compression requirement.

The GDA procedure is at the core of the optimization process. It involves updating both the leader's and follower's parameters in the direction of their respective gradient. The updates are detailed below.

3.3.1. Leader's Update

The leader's parameters $\boldsymbol{\theta}$ are updated in the opposite direction of the gradient of the joint objective function $f(\boldsymbol{\theta}, \boldsymbol{\mu})$, i.e.,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{\mu})$$

where α denotes the learning rate, and the gradient is given by

$$\nabla_{\boldsymbol{\theta}_k} f(\boldsymbol{\theta}, \boldsymbol{\mu}) = - \mathop{\mathbb{E}}_{(s,a) \sim d^{sa}_{\boldsymbol{\theta}_k}} \left[\left(\frac{d^{sa}_{\boldsymbol{\mu}}(s,a)}{d^{sa}_{\boldsymbol{\theta}_k}(s,a)} \right)^2 \nabla_{\boldsymbol{\theta}_k} \log d^{sa}_{\boldsymbol{\theta}_k}(s,a) \right].$$

3.3.2. Follower's Update

The follower's parameters $\boldsymbol{\mu}$ are updated in the direction of the gradient of the joint objective function $f(\boldsymbol{\theta}, \boldsymbol{\mu})$, i.e.,

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \beta \nabla_{\boldsymbol{\mu}} f(\boldsymbol{\theta}, \boldsymbol{\mu})$$

where β denotes the learning rate, and the gradient is given by

$$\nabla_{\boldsymbol{\mu}} f(\boldsymbol{\theta}, \boldsymbol{\mu}) = 2 \mathop{\mathbb{E}}_{(s,a) \sim d^{sa}_{\boldsymbol{\theta}_k}} \left[\left(\frac{d^{sa}_{\boldsymbol{\mu}}(s,a)}{d^{sa}_{\boldsymbol{\theta}_k}(s,a)} \right)^2 \nabla_{\boldsymbol{\mu}} \log d^{sa}_{\boldsymbol{\mu}}(s,a) \right].$$

Here, $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{\mu})$ and $\nabla_{\boldsymbol{\mu}} f(\boldsymbol{\theta}, \boldsymbol{\mu})$ represent the gradients of the joint objective function with respect to the leader's and follower's parameters, respectively.

3.3.3. Algorithmic Steps

Having introduced the leadear's and follower's updates, we can now present the main algorithmic steps below.

1. Initial Policy Space: Start with a small number of policies, e.g., $K = 1$.
2. Find a Differential Stackelberg Equilibrium (DSE): Compute a DSE for the given number of policies.
3. Check Global Requirement: Verify if the leader's strategy satisfies the global requirement, which is defined as:

   $$\max_{\boldsymbol{\mu} \in \Theta} f\left(\boldsymbol{\theta}^*, \boldsymbol{\mu}\right) \leq \sigma$$

   Here, σ is a predefined threshold.
4. Compression Outcome: If the global requirement is met, it means the policy space compression problem is successfully solved, and $\boldsymbol{\theta}^*$ represents a σ-compression of Θ.
5. Increment Policies: If the global requirement is not met, increment the number of policies K and repeat the process. This iterative approach ensures that a valid σ compression will eventually be found.

A crucial aspect of the algorithm is to ensure a large timescale separation, denoted as $\tau := \beta/\alpha$. This separation guarantees convergence to a Differential Stackelberg Equilibrium (DSE), as studied in [4].

In summary, the PSCA algorithm combines gradient-based optimization techniques with specific strategies for handling non-differentiable objective functions to systematically achieve policy space compression while satisfying global requirements in the context of a cover game. However, it is not immediate how the global requirement can be checked in step 3 of the algorithm, or the number of policies K that are required to find a valid σ-compression. The next chapters will dive into this questions, for which the corresponding answers constitutes the central contribution of this thesis.

4 | On the Covering Threshold

In this chapter, we study the role of the covering threshold σ in the policy space compression problem, and how to check the corresponding requirement. First, we revise the geometry of the problem in Section 4.1. Then, in Section 4.2, we discuss how to set a covering threshold properly. Finally, we provide solutions to check whether the requirement implied by a covering threshold is satisfied (Section 4.3), both in unconstrained polytopal space (Section 4.3.1) and constrained polytopal space (Section 4.3.2).

4.1. Geometry of the Problem

The policy space is a bounded convex polytope which has $|SA|$ zero-dimensional vertices, $\binom{|SA|}{2}$ one-dimensional edges, $\binom{|SA|}{3}$ two-dimensional faces, and so on up to our original polytope, which is basically one $|SA|$-dimensional shape exactly similar to the convex hull of all the vertices.

The space of all the state distributions is an $|S|$-dimensional polytope, whose vertices denote the situation in which a single state is visited with probability one. The space of all the state-action distributions is an $|SA|$-dimensional polytope, whose vertices denote the situation in which a single state is visited with probability one, and a single action is taken in that state with probability one.

Having established the basic geometry, it is worth taking a look at the values of the parameter σ, i.e., the coverage threshold, which are meaningful from a geometric standpoint.

4.2. How to Set the Covering Threshold

It is easy to see that $\sigma \geq |SA|$ is not meaningful, as it admits a trivial solution to the policy space compression problem.[1] Indeed, this threshold allows to cover the whole original policy space with just one representative policy, which is the one inducing a uniform distribution over all the state-action pairs.

[1]Note that this analysis does not consider MDP constraints yet.

We can easily prove the latter by contradiction. Let us assume

$$\frac{d_\mu^2(s,a)_1}{\frac{1}{|SA|}} + \cdots + \frac{d_\mu^2(s,a)_{|SA|}}{\frac{1}{|SA|}} > |SA|.$$

Then, we have

$$\int \frac{d_\mu^2(s,a)}{\frac{1}{|SA|}} \,\mathrm{d}s\,\mathrm{d}a > |SA| \quad \Rightarrow \quad \int d_\mu^2(s,a) \,\mathrm{d}s\,\mathrm{d}a > 1,$$

which always holds true if one of the two conditions below hold true:

- $\exists (s,a) \in SA$ such that $d_\mu(s,a) > 1$
- $\int d_\mu(s,a) \,\mathrm{d}s\,\mathrm{d}a > 1$

Since none of the latter respect our assumption on d_μ being a distribution, we can conclude that

$$\frac{d_\mu^2(s,a)_1}{\frac{1}{|SA|}} + \cdots + \frac{d_\mu^2(s,a)_{|SA|}}{\frac{1}{|SA|}} \leq |SA|,$$

which reveals that $\sigma = |SA|$ is not a challenging covering threshold.

4.2.1. A Sensible Range of σ

In a polytope of $|SA|$ dimensions the range of σ values from $|SA|/2$ to $|SA|$ has a very peculiar property: We can show that the number of covering policies needed to solve the policy space compression is at most $|SA| + 1$ when σ is selected this range.

Intuitively, if we choose one of our covering policies such that its state-action distribution is at the center of the state-action distribution polytope (i.e., the one corresponding with uniformly distributed state-action probabilities), then, by computing the Rènyi divergence between this policy and the policies with state-action distributions falling on the middle of the edges of the polytope, we reach the value of $|SA|/2$ for σ.

Indeed, we can easily compute the Rènyi divergence between the latter policies as follows:

$$\int \frac{d_\mu^2(s,a)}{d_\theta(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \frac{\left(\frac{1}{2}\right)^2}{\frac{1}{|SA|}} + \frac{\left(\frac{1}{2}\right)^2}{\frac{1}{|SA|}} + \frac{(0)^2}{\frac{1}{|SA|}} + \cdots + \frac{(0)^2}{\frac{1}{|SA|}} = \frac{|SA|}{2}$$

$$\int \frac{d_\mu^2(s,a)}{d_\theta(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \frac{\left(\frac{1}{2}\right)^2}{\frac{1}{|SA|}} + \frac{(0)^2}{\frac{1}{|SA|}} + \frac{\left(\frac{1}{2}\right)^2}{\frac{1}{|SA|}} + \frac{(0)^2}{\frac{1}{|SA|}} + \cdots + \frac{(0)^2}{\frac{1}{|SA|}} = \frac{|SA|}{2}$$

.

.

.

$$\int \frac{d_\mu^2(s,a)}{d_\theta(s,a)} \text{ ds da} = \frac{\left(\frac{1}{2}\right)^2}{\frac{1}{|SA|}} + \frac{(0)^2}{\frac{1}{|SA|}} + \cdots + \frac{(0)^2}{\frac{1}{|SA|}} + \frac{\left(\frac{1}{2}\right)^2}{\frac{1}{|SA|}} = \frac{|SA|}{2}$$

Then, we choose our other $|SA|$ representative policies such that their state-action distribution fall between the representative policy at the center and $|SA|$ vertices of our polytope. It can be shown that each one of the aforementioned representative policies cover all the remaining policies left uncovered by the representative policy at the center.

We can show the latter as

$$\int \frac{d_\mu^2(s,a)}{d_\theta(s,a)} \text{ ds da} = \frac{(1)^2}{d_\theta(s,a)_1} + \frac{(0)^2}{d_\theta(s,a)_2} + \cdots + \frac{(0)^2}{d_\theta(s,a)_{|SA|}} = \sigma$$

$$\int \frac{d_\mu^2(s,a)}{d_\theta(s,a)} \text{ ds da} = \frac{\left(\frac{1}{2}\right)^2}{d_\theta(s,a)_1} + \frac{\left(\frac{1}{2}\right)^2}{d_\theta(s,a)_2} + \frac{(0)^2}{d_\theta(s,a)_3} + \cdots + \frac{(0)^2}{d_\theta(s,a)_{|SA|}} = \sigma$$

$$\int \frac{d_\mu^2(s,a)}{d_\theta(s,a)} \text{ ds da} = \frac{\left(\frac{1}{2}\right)^2}{d_\theta(s,a)_1} + \frac{\left(\frac{1}{2}\right)^2}{d_\theta(s,a)_3} + \frac{(0)^2}{d_\theta(s,a)_4} + \cdots + \frac{(0)^2}{d_\theta(s,a)_{|SA|}} = \sigma$$

.

.

.

$$\int \frac{d_\mu^2(s,a)}{d_\theta(s,a)} \text{ ds da} = \frac{\left(\frac{1}{2}\right)^2}{d_\theta(s,a)_1} + \frac{(0)^2}{d_\theta(s,a)_2} + \cdots + \frac{(0)^2}{d_\theta(s,a)_{|SA|-1}} + \frac{\left(\frac{1}{2}\right)^2}{d_\theta(s,a)_{|SA|}} = \sigma$$

$$\int d_\theta(s,a) \text{ ds da} = 1$$

Solving the latter system of equations we have $\frac{|SA|+2}{3}$ for our unknown σ, which is less than $\frac{|SA|}{2}$ for $|SA|$ greater than 4, proving our claim.

4.3. How to Check the Covering Requirement

Having discussed about how to set the covering threshold σ, we now study how can we check whether a tentative compression, i.e., a set of leader's parameters $\theta_1, \dots, \theta_K$, satisfies the σ threshold *globally*. Formally, we want to check if it holds

$$\max_{\mu \in \Theta} \min_{k \in [K]} \int \frac{d_\mu^2(s,a)}{d_{\theta_k}(s,a)} \text{d}(s,a) \leq \sigma.$$

First, we study the setting in which the covering problem is defined on a polytopal space without additional constraints (Section 4.3.1), providing approximate solution from both an optimization and an algorithmic perspective. Then, in Section 4.3.2, we replicate similar reasoning for a covering problem defined on a polytopal space with additional constraints, such as those given by the MDP model, in which the stochastic vectors d_μ and d_{θ_k}, $\forall k \in [K]$, has to be state-action distributions induced by a policy in the MDP.

4.3.1. Unconstrained Polytopal Space

For checking the covering requirement over unconstrained polytopal spaces, we provide a family of solutions coming from optimization and algorithmic perspectives.

Two Types of Problem Reformulations to Check the Covering Requirement Approximately

Here we try to overcome the hardness of checking the covering guarantees from an optimization perspective, i.e., considering tractable approximate reformulations of the original problem.

Reformulation of the 1st Type Here we consider the follower's parameters μ as decision variables, to answer the question: Is there any remaining policy not covered by representative ones? Specifically, we consider the following program for some $j \in [K]$.

$$
\begin{aligned}
& \max \int \frac{d_\mu^2(s,a)}{d_{\theta_j}(s,a)} \,\mathrm{d}s\,\mathrm{d}a \\
& s.t. \\
& \qquad \int d_\mu(s,a)\,\mathrm{d}s\,\mathrm{d}a = 1 \\
& \qquad \int \frac{d_\mu^2(s,a)}{d_{\theta_k}(s,a)} \,\mathrm{d}s\,\mathrm{d}a > \sigma && \forall k \in K \mid \{k=1\} \\
& \qquad d_\mu(s,a) \geq 0 && \forall (s,a) \in SA \\
& \qquad d_\mu(s,a) \leq 1 && \forall (s,a) \in SA
\end{aligned}
$$

The intuition is the following: If there are remaining policies outside the σ distance of $\big(|K|-1\big)$ representative policies, then among them which one will be maximally far from the remaining representative policies and how much will be this maximum distance? If this distance is greater than σ, then it means our representative policies are not enough to cover the polytope.

By randomly choosing one of K representative policies to be used in the objective function and using the remaining $\big(K-1\big)$ ones in our constraints we reach our goal. Then, we just have to solve this optimization problem K times for all the different $j \in [K]$. The minimum value of the K values attained by the programs is the exact value of our original maximin formulation. Unfortunately, this reformulation is a *QCQP* that is NP-hard in general. However, there exists relaxations like *semidefinite programming* (*SDP*), whcih can be used to approximate the value of the program.

Reformulation of the 2nd Type Another way to reformulate our problem is through the following program, in which we have just substituted our objective function with our linear constraint and vice versa.

$$\min \int d_\mu(s,a)\, \mathrm{d}s\, \mathrm{d}a$$

$$s.t.$$

$$\begin{aligned}
&\int \frac{d^2_\mu(s,a)}{d_{\theta_k}(s,a)}\, \mathrm{d}s\, \mathrm{d}a > \sigma && \forall k \in K \\
&d_\mu(s,a) \geq 0 && \forall (s,a) \in SA \\
&d_\mu(s,a) \leq 1 && \forall (s,a) \in SA
\end{aligned}$$

In this new reformulation, if the min value of our objective function is less than 1, it means there is at least one remaining policy left uncovered by the representative policies. At the same time, by knowing the values of our $d_\mu(s,a)$ variables that gave the minimum value in the objective, we obtain a policy left uncovered.

Nonetheless, the latter program is a *Quadratic Constraint Linear Program*, which is NP-hard in general. Similarly as before, we can employ *SDP* to have an approximate solution.

Two Types of Algorithms to Check the Covering Requirement Exactly

To try to find an exact approach to check the covering requirement, we follow an algorithmic route instead of the optimization view described in the previous section. Especially, we present two types of algorithms that iteratively solve local problems to assess the requirement globally.

Algorithm of the 1st Type

The main idea of this algorithmic solution is to first find the representative policies (and polytope boundaries) in the neighborhood of the different representative policies, then solving some system of equations to assess local covering requirements around the representative policies one by one.

As we shall see, all the optimization problems involved in this algorithm are *Linear Programs* (*LPs*), the number of their decision variables and their constraints are finite and the whole algorithm components can be executed in polynomial time.

Finding the Neighbors of One Representative Policy Without loosing generality, we consider one representative policy $j \in [K]$. Then, we consider different selections of $|SA|$ members from the remaining $K-1$ representative policies and polytope sides set SA. For each of these selections, we have to solve different systems of linear equations. The number of systems is equal to $\binom{K-1+|SA|}{|SA|}$. We differentiate between systems of Type A and Systems of Type B.

- **System of Type A**

  Let us define the set X of systems such that, for each element $x \in X$, we define the set

  $$K' = \{|SA| \textit{ members } k'\textit{'s from } \big(K-1\big) \textit{ representative policies}\}.$$

  We can say the number $|X|$ of systems to be solved is equal to $\binom{K-1}{|SA|}$.

  Now, for every $x \in X$ and corresponding K', we can define a system of $|SA|$ quadratic equations and $|SA|$ unknowns as follows

  $$\int \frac{d^2_{\mu_x}(s,a)}{d_{\theta_{k'}}(s,a)} \,\mathrm{d}s \,\mathrm{d}a = \sigma \qquad \forall k' \in K'.$$

  For each of the systems above, we check whether the policy found by that system $\mu^\star_x$ is beyond a σ distance from all the other representative policies and within a σ distance from the representative policy j. In other words, we check whether the two conditions below hold true:

  $$1 \cdot \quad \int \frac{d^2_{\mu^\star_x}(s,a)}{d_{\theta_k}(s,a)} \,\mathrm{d}s \,\mathrm{d}a > \sigma \qquad \forall k \in K \mid \{k = j, k \in K'\}$$

$$2\cdot \quad \int \frac{d^2_{\mu^\star_x}(s,a)}{d_{\theta_j}(s,a)}\,\mathrm{d}s\,\mathrm{d}a \le \sigma$$

If the answer is positive, then all the related $|SA|$ representative policies are in the neighborhood of the representative policy j.

- **System of Type B**

Here we define e new type of system in which both representative policies and polytope sides are involved. Let us define the set of systems Y and its members y as

$$y = \{|SA| \textit{ members from } \big(|K|-1\big) \textit{ representative policies and polytope sides on condition of at least one and at most } \big(|SA|-1\big) \textit{ polytope sides}\}.$$

We can say the number $|Y|$ of systems to be solved is equal to $\binom{|K|-1+|SA|}{|SA|} - \binom{|K|-1}{|SA|}$.

Then, defining two sets K' and $S'A'$ as

$$K' = \{|K'| \textit{ members } k'\textit{'s from } \big(|K|-1\big) \textit{ representative policies}\}$$

$$S'A' = \{|S'A'| \textit{ members } (s',a')\textit{'s from polytope sides set } SA\}$$

where $|k'| + |S'A'| = |SA|$ and $0 < |S'A'| < \big(|SA|-1\big)$.

Now, for each $y \in Y$ and corresponding K' and $S'A'$, we can define a system with $|SA|$ quadratic equations and $|SA|$ unknowns as follows

$$\int \frac{d^2_{\mu_y}(s,a)}{d_{\theta_{k'}}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \sigma \qquad \forall k' \in K'$$

$$d_{\mu_x}(s',a') = 0 \qquad \forall (s',a') \in S'A'$$

Then, for each of the systems above we check whether the policy found by that system $\mu^\star_y$ is beyond a σ distance from all the other representative policies and within a σ distance from the representative policy j. In other words, we check whether both of the two conditions below hold true:

$$1\cdot \quad \int \frac{d^2_{\mu^\star_y}(s,a)}{d_{\theta_k}(s,a)}\,\mathrm{d}s\,\mathrm{d}a > \sigma \qquad \forall k \in K \mid \{k=j, k \in K'\}$$

$$2\cdot \quad \int \frac{d^2_{\mu^\star_y}(s,a)}{d_{\theta_j}(s,a)}\,\mathrm{d}s\,\mathrm{d}a \le \sigma$$

If the answer is positive, then all the related $|SA|$ representative policies and polytope sides are in the neighborhood of the representative policy j.

Checking the Covering Requirement Locally As in the paragraph above, we consider the representative policy $j \in [K]$ without loosing generality. We aim to check the covering requirement on this representative policy locally. In other words, we try to answer the following question: Is there any remaining policy among our representative policy j and its neighbor representative policies and polytope sides that is not covered?

To answer the question, we consider different selections of $\big(|SA|-1\big)$ members from our representative policies set K' and polytope sides set $S'A'$. Now, for these different selections we have to solve different systems of equations. The number of systems is equal to $\binom{|K'|+|S'A'|}{|SA|-1}$, and the systems can be of the types A or B.

- **System of Type A**

  For each system in X such that only representative policies are involved, we define a set K'' as

  $$K'' = \{\big(|SA|-1\big) \textit{ members } k''\textit{'s from representative policies set } K'\}.$$

  We can say the number $|Y|$ of systems to be solved is equal to $\binom{|K'|}{|SA|-1}$.

  Then, we introduce another unknown denoted as σ_x^j. Now, for each $x \in X$ and its corresponding set K'', we have to solve a system with $\big(|SA|+1\big)$ linear and quadratic equations and $\big(|SA|+1\big)$ unknowns defined as

  $$\int \frac{d^2_{\mu_x^j}(s,a)}{d_{\theta_j}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \sigma_x^j$$

  $$\int \frac{d^2_{\mu_x^j}(s,a)}{d_{\theta_{k''}}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \sigma_x^j \qquad \forall k'' \in K''$$

  $$\int d_{\mu_x^j}(s,a) \,\mathrm{d}s\,\mathrm{d}a = 1$$

  Among the systems above, if there is at least one system for which we get a value greater than σ for our unknown σ_x^j, it means there is at least a remaining policy left uncovered. Instead, no system with unknown σ_x^j greater than σ means that no remaining policy is left uncovered and we can stop there.

  In the former case, in order to find the exact value of our original maximin problem, we see for which system we reached the maximum value for our unknown σ_x^j and we save it[2]

$$\boldsymbol{\sigma}^{\boldsymbol{j}}_{\boldsymbol{x}^\star} = \max_{\boldsymbol{x}} \boldsymbol{\sigma}^{\boldsymbol{j}}_{\boldsymbol{x}}$$

and we also save the remaining policy found by that related system in $\boldsymbol{d}_{\boldsymbol{\mu}^{\boldsymbol{j}}_{\boldsymbol{x}^\star}}(\boldsymbol{s},\boldsymbol{a})$.

- **System of Type B**

  For each system in X such that only representative policies and polytope sides are involved, we define a set Y having memebers

  $y = \{\big(|SA|-1\big)$ *members from representative policies set* K' *and polytope sides set* $S'A'$ *on condition of at least one member selected from polytope sides*$\}$.

  We can say the number $|Y|$ of systems to be solved is equal to $\binom{|K'|+|S'A'|}{|SA|-1} - \binom{|K'|}{|SA|-1}$.

  We further define two sets K'' and $S''A''$ as

$$K'' = \{|K''| \textit{ members } k''\textit{'s from representative policies set } K'\}$$

$$S''A'' = \{|S''A''| \textit{ members } (s'',a'')\textit{'s from polytope sides set } S'A'\}$$

  where $|k''| + |S''A''| = |SA| - 1$ and $|S''A''|$ is at least equal to 1.

  Then, we introduce another unknown denoted as σ^j_y instead of the value of σ. Now, for each $y \in Y$ and corresponding sets K'' and $S''A''$, we have to solve a system with $\big(|SA|+1\big)$ linear and quadratic equations and $\big(|SA|+1\big)$ unknowns as follows

$$\int \frac{d^2_{\mu^j_y}(s,a)}{d_{\theta_j}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \sigma^j_y$$

$$\int \frac{d^2_{\mu^j_y}(s,a)}{d_{\theta_{k''}}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \sigma^j_y \qquad \forall k'' \in K''$$

$$\int d_{\mu^j_y}(s,a)\,\mathrm{d}s\,\mathrm{d}a = 1$$

$$d_{\mu^j_y}(s'',a'') = 0 \qquad \forall (s'',a'') \in S''A''$$

  If among the systems above there is at least one system that gives a value greater than σ for our unknown σ^j_y, it means there is at least one remaining policy left uncovered. Instead, no system with unknown σ^j_y greater than σ means no remaining policy is left uncovered and we can stop there.

[2]let us consider $\boldsymbol{\sigma}^{\boldsymbol{j}}_{\boldsymbol{x}}$ as the values of unknowns σ^j_x's and $\boldsymbol{d}_{\boldsymbol{\mu}^{\boldsymbol{j}}_{\boldsymbol{x}}}$'s as the values of state action pairs distributions of different remaining policies $d_{\mu^j_x}$ found by different systems.

In the former case, in order to find the exact value of our original maximin problem, we see for which one of those systems we have reached the maximum value for our unknown σ_y^j and we save it[3]

$$\boldsymbol{\sigma}_{\boldsymbol{y}^\star}^{\boldsymbol{j}} = \max_{\boldsymbol{y}} \boldsymbol{\sigma}_{\boldsymbol{y}}^{\boldsymbol{j}}$$

and we also save the remaining policy found by that related system $\boldsymbol{d}_{\boldsymbol{\mu}_{\boldsymbol{y}^\star}^{\boldsymbol{j}}}(\boldsymbol{s}, \boldsymbol{a})$.

Checking the Covering Requirement Globally Since we have K different representative policies, the total number of the systems we have to solve is $K\binom{|K'|+|S'A'|}{|SA|-1}$, in which $|K|\binom{|K'|}{|SA|-1}$ systems are of type A and $|K|\binom{|K'|+|S'A'|}{|SA|-1} - |K|\binom{|K'|}{|SA|-1}$ systems are of type B. Now, we have to find

$$\boldsymbol{\sigma}_{\boldsymbol{x}^\star}^{\boldsymbol{j}^\star} = \max_{\boldsymbol{j}} \max_{\boldsymbol{x}} \boldsymbol{\sigma}_{\boldsymbol{x}}^{\boldsymbol{j}}$$

for which we save the values of their unknowns $\boldsymbol{d}_{\boldsymbol{\mu}_{\boldsymbol{x}^\star}^{\boldsymbol{j}^\star}}(\boldsymbol{s}, \boldsymbol{a})$, and

$$\boldsymbol{\sigma}_{\boldsymbol{y}^\star}^{\boldsymbol{j}^\star} = \max_{\boldsymbol{j}} \max_{\boldsymbol{y}} \boldsymbol{\sigma}_{\boldsymbol{y}}^{\boldsymbol{j}}$$

and again we save the values of $\boldsymbol{d}_{\boldsymbol{\mu}_{\boldsymbol{y}^\star}^{\boldsymbol{j}^\star}}(\boldsymbol{s}, \boldsymbol{a})$.

Then, we compare the values of unknowns $\boldsymbol{\sigma}_{\boldsymbol{x}^\star}^{\boldsymbol{j}^\star}$ and $\boldsymbol{\sigma}_{\boldsymbol{y}^\star}^{\boldsymbol{j}^\star}$. Each of them that is greater than one another is the exact value of our original maximin problem and the values of other unknowns of the relevant system is the state-action distribution of the policy left uncovered at the longest distance from our representative policies. Hence, we have

$$\max_{\mu} \min_{k} \int \frac{d_\mu^2(s,a)}{d_{\theta_k}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \max\{\boldsymbol{\sigma}_{\boldsymbol{x}^\star}^{\boldsymbol{j}^\star}, \boldsymbol{\sigma}_{\boldsymbol{y}^\star}^{\boldsymbol{j}^\star}\}.$$

An Alternative Algorithm to Find the Neighbors of One Representative Policy We presented before an algorithm to find the neighbors of one representative policy. Here we present an alternative algorithm that is more convenient when the neighbors of the representative policy are *symmetrically distributed*. We say that a set of representative policies are symmetrically distributed if there is an intersection point in the polytope space at a particular distance (measured in terms of Rènyi divergence) from all of them.

By eliminating linear constraints from our optimization problem and substituting our

[3]Let us consider $\boldsymbol{\sigma}_{\boldsymbol{y}}^{\boldsymbol{j}}$'s as the values of unknowns σ_y^j's and $\boldsymbol{d}_{\boldsymbol{\mu}_{\boldsymbol{y}}^{\boldsymbol{j}}}$'s as the values of state action pairs distributions of different remaining policies $d_{\mu_y^j}$ found by different systems

decision variables from $d_\mu(s,a)$ to $d_M(s,a)$, we obtain the following program

$$\min \int \frac{d_M^2(s,a)}{d_{\theta_1}(s,a)}\, \mathrm{d}s\ \mathrm{d}a$$

$s.t.$

$$\begin{aligned}
&\int \frac{d_M^2(s,a)}{d_{\theta_k}(s,a)}\, \mathrm{d}s\ \mathrm{d}a \geq \sigma && \forall k \in K \mid \{k=1\} \\
&d_M(s,a) \geq 0 && \forall (s,a) \in SA \\
&d_M(s,a) \leq 1 && \forall (s,a) \in SA
\end{aligned}$$

Then, we convert this problem into an LP problem by introducing the variables $D_M(s,a)$, which stands for the squared variable $d_M(s,a)$ for different $|SA|$ state action pairs. We obtain

$$\min \int \frac{D_M(s,a)}{d_{\theta_1}(s,a)}\, \mathrm{d}s\ \mathrm{d}a$$

$s.t.$

$$\begin{aligned}
&\int \frac{D_M(s,a)}{d_{\theta_k}(s,a)}\, \mathrm{d}s\ \mathrm{d}a \geq \sigma && \forall k \in K \mid \{k=1\} \\
&D_M(s,a) \geq 0 && \forall (s,a) \in SA \\
&D_M(s,a) \leq 1 && \forall (s,a) \in SA
\end{aligned}$$

which can be solved in polynomial time.

We have to solve the problem K times for the K different representative policies, changing the objective function and remaining constraints accordingly:

$$\min \int \frac{D_M(s,a)}{d_{\theta_j}(s,a)}\, \mathrm{d}s\ \mathrm{d}a$$

$s.t.$

$$\begin{aligned}
&\int \frac{D_M(s,a)}{d_{\theta_k}(s,a)}\, \mathrm{d}s\ \mathrm{d}a \geq \sigma && \forall k \in K \mid \{k=j\} \\
&D_M(s,a) \geq 0 && \forall (s,a) \in SA
\end{aligned}$$

$$D_M(s,a) \leq 1 \qquad\qquad \forall (s,a) \in SA$$

Then, for each one of the K optimization problems we save the values of our $|SA|$ variables $D_M(s,a)$ that attain the minimum of the objective functions:

$$M^{\star}_{\theta_j} = \underset{M}{\operatorname{argmin}} \int \frac{D_M(s,a)}{d_{\theta_j}(s,a)} \,\mathrm{d}s\,\mathrm{d}a \qquad \forall j \in K$$

Based on our polytope characteristics 4.1, each one of $|SA|$ values of $D_{M^{\star}_{\theta_j}}(s,a)$ that is equal to zero refers to one side of our polytope in the neighbourhood of the representative policy j.

On the other hand, each one of the K constraints of our optimization problem that are active refer to one representative policy in the neighbourhood of representative policy j.

We report below the pseudocode of the alternative algorithm.

Algorithm 4.1 The Alternative Algorithm

```
1: BEGIN
2: $|K'| := 0;$
3: $|S'A'| := 0;$
4: $K' := \{j\};$
5: $S'A' := \emptyset;$
6: for $k \in K \mid \{k = j\}$ do
7:     if $\int \frac{D_{M^{\star}_{\theta_j}}(s,a)}{d_{\theta_k}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \sigma$ then
8:         $|K'| := |K'| + 1;$
9:         $K' := K' \cup \{k\};$
10:    end if
11: end for
12: for $(s,a) \in SA$ do
13:    if $D_{M^{\star}_{\theta_j}}(s,a) = 0$ then
14:        $|S'A'| := |S'A'| + 1;$
15:        $S'A' := S'A' \cup \{(s,a)\};$
16:    end if
17: end for
18: END
```

Algorithm of the 2nd Type

In an $|SA|$-dimensional polytope, we consider different selections of $|SA|$ members from our representative policies set K and polytope sides set SA. Then, for these different selections we have to solve different systems of equations. The number of systems are equal to $\binom{|K|+|SA|}{|SA|}$. As usual, we differentiate between systems of type A or type B.

- **System of Type A**

  In the selections in which just representative policies are involved, we say the system is of type A, and we define the set of systems X and its members $x \in X$ in a way that each one of them indicates to a set K' defined as

  $$K' = \{|SA| \textit{ members k''s from representative policies set } K\}$$

  We can say that the number $|X|$ of systems to be solved is equal to $\binom{K}{|SA|}$.

  We introduce another unknown denoted as σ_x instead of the value of σ. Then, for every $x \in X$ and corresponding set K', we have to solve a system with $\big(|SA|+1\big)$ linear and quadratic equations and with $\big(|SA|+1\big)$ unknowns:

  $$\int \frac{d^2_{\mu_x}(s,a)}{d_{\theta_{k'}}(s,a)} \,\mathrm{d}s\, \mathrm{d}a = \sigma_x \qquad \forall k' \in K'$$

  $$\int d_{\mu_x}(s,a) \,\mathrm{d}s\, \mathrm{d}a = 1$$

  For each of the systems above where the value of variable σ_x is greater than σ. Then, for each of the systems above where the value of variable σ_x is greater than σ, we must see if there is any other representative policy in set K that the policy found by the related system $\mu^\star_x$ that is within the σ distance from it or not:

  $$\int \frac{d^2_{\mu^\star_x}(s,a)}{d_{\theta_k}(s,a)} \,\mathrm{d}s\, \mathrm{d}a \leq \sigma \qquad \forall k \in K \mid \{k \in K'\}$$

- **System of Type B**

  In the selections that both representative policies and polytope sides are involved, we say the system is of type B. We define this set of systems as Y and its members y as

  $y = \{|SA|$ *members from representative policies set* K *and polytope sides set* SA *on condition of at least one and at most* $\big(|SA|-1\big)$ *members selected from polytope sides*$\}$

  We can say that the number of systems $|Y|$ to be solved is equal to $\binom{|K|+|SA|}{|SA|} - \binom{|K|}{|SA|}$.

  We further define two sets K' and $S'A'$ as

  $$K' = \{|K'| \textit{ members k''s from representative policies set } K\}$$

$$S'A' = \{|S'A'| \textit{ members } (s',a')\textit{'s from polytope sides set } SA\}$$

where $|k'| + |S'A'| = |SA|$ and $|S'A'|$ is at least equal to 1 and at most equal to $\big(|SA| - 1\big)$.

The we introduce antoher unknown denoted by σ_y instead of the value of σ. Then, for each $y \in Y$ and corresponding K' and $S'A'$, we have to solve a system with $\big(|SA|+1\big)$ linear and quadratic equations and with $\big(|SA|+1\big)$ unknowns as follows

$$\int \frac{d^2_{\mu_y}(s,a)}{d_{\theta_{k'}}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \sigma_y \qquad \forall k' \in K'$$

$$\int d_{\mu_y}(s,a)\,\mathrm{d}s\,\mathrm{d}a = 1$$

$$d_{\mu_x}(s',a') = 0 \qquad \forall (s',a') \in S'A'$$

Then, for each of the systems above where the value of variable σ_y is greater than σ, we must see if there is any other representative policy in set K that the policy found by the related system $\mu^\star_y$ that is within the σ distance from it or not:

$$\int \frac{d^2_{\mu^\star_y}(s,a)}{d_{\theta_k}(s,a)} \,\mathrm{d}s\,\mathrm{d}a \le \sigma \qquad \forall k \in K \mid \{k \in K'\}$$

Finally, in Appendix A, we provide a numerical example of the machinery pf the presented algorithms.

4.3.2. Constrained Polytopal Space

Let us now consider of subpolytope of the original $|SA|$-dimensional polytope induced by addittional constraints. We assume to know the vertices V of the subpolytope, we have to find the subpolytope sides.

An Algorithm to Find Subpolytope Sides

We start defining the set v':

$$v' = \{\big(|SA|-1\big) \textit{ members from vertices set } V\}$$

Then, we find hyperplane equations passing through all the $\big(|SA|-1\big)$ members of the sets v' and the central point, which is a zero vector with the length of $|SA|$. By calling the set of sets v' as V', the number of sets $|V'|$ is $\binom{|V|}{|SA|-1}$. Thus, we can have at most $|V'|$

unique hyperplane equations as follows:

$$\int a_{v'}(s,a) \cdot d_{\mu}(s,a) \text{ ds da} = b_{v'} \qquad \forall v' \in V'$$

For the hyperplane equations found above, by eliminating the hyperplane equations who have satisfied none of the conditions below:

$$1 \cdot \quad \int a_{v'}(s,a) \cdot d_{v}(s,a) \text{ ds da} \geq b_{v'} \qquad \forall v \in V$$

$$2 \cdot \quad \int a_{v'}(s,a) \cdot d_{v}(s,a) \text{ ds da} \leq b_{v'} \qquad \forall v \in V$$

and by eliminating the remaining repeated hyperplane equations, we reach the polytope sides h. From now on, let us denote $a_h(s,a)$ and $b_h(s,a)$ instead of $a_{v'}(s,a)$ and $b_{v'}(s,a)$ in the related hyperplane equations respectively. Thus, our polytope sides are as follows:

$$\int a_h(s,a) \cdot d_{\mu}(s,a) \text{ ds da} = b_h \qquad \forall h \in H$$

Some of the subpolytope sides can coincide with our original polytope sides. For them, we have the hyperplane equations with $a_h(s,a)$ equal to one for a particular state action pair and equal to zero for all the other $a_h(s,a)$ and $b_h(s,a)$.

Let us also denote as $h^{(1)}$ and $h^{(2)}$ for the hyperplanes who satisfied the first and second condition respectively. We have

$$1 \cdot \quad \int a_{h^{(1)}}(s,a) \cdot d_{v}(s,a) \text{ ds da} \geq b_{h^{(1)}} \qquad \forall v \in V,\ \forall h^{(1)} \in H^{(1)}$$

$$2 \cdot \quad \int a_{h^{(2)}}(s,a) \cdot d_{v}(s,a) \text{ ds da} \leq b_{h^{(2)}} \qquad \forall v \in V,\ \forall h^{(2)} \in H^{(2)}$$

and also $H = H^{(1)} \cup H^{(2)}$ and $|H| = |H^{(1)}| + |H^{(2)}|$.

Two Types of Reformulations to Check the Covering Requirement Approximately

Here we replicate the reformulations provided in the previous section with slight variations to adapt them to the subpolytope setting. Most of the comments provided in Section 4.3.1 applies verbatim.

- **Reformulation of the 1st Type**

$$\max \int \frac{d_\mu^2(s,a)}{d_{\theta_j}(s,a)}\,\mathrm{d}s\,\mathrm{d}a$$

$s.t.$

$$\int d_\mu(s,a)\,\mathrm{d}s\,\mathrm{d}a = 1$$

$$\int \frac{d_\mu^2(s,a)}{d_{\theta_k}(s,a)}\,\mathrm{d}s\,\mathrm{d}a > \sigma \qquad \forall k \in K \mid \{k = j\}$$

$$\int a_{h^{(1)}}(s,a)\cdot d_\mu(s,a)\,\mathrm{d}s\,\mathrm{d}a \geq b_{h^{(1)}} \qquad \forall h^{(1)} \in H^{(1)}$$

$$\int a_{h^{(2)}}(s,a)\cdot d_\mu(s,a)\,\mathrm{d}s\,\mathrm{d}a \leq b_{h^{(2)}} \qquad \forall h^{(2)} \in H^{(2)}$$

$$d_\mu(s,a) \geq 0 \qquad \forall (s,a) \in SA$$

$$d_\mu(s,a) \leq 1 \qquad \forall (s,a) \in SA$$

- **Reformulation of the 2nd Type**

$$\min \int d_\mu(s,a)\,\mathrm{d}s\,\mathrm{d}a$$

$s.t.$

$$\int \frac{d_\mu^2(s,a)}{d_{\theta_k}(s,a)}\,\mathrm{d}s\,\mathrm{d}a > \sigma \qquad \forall k \in K$$

$$\int a_{h^{(1)}}(s,a)\cdot d_\mu(s,a)\,\mathrm{d}s\,\mathrm{d}a \geq b_{h^{(1)}} \qquad \forall h^{(1)} \in H^{(1)}$$

$$\int a_{h^{(2)}}(s,a)\cdot d_\mu(s,a)\,\mathrm{d}s\,\mathrm{d}a \leq b_{h^{(2)}} \qquad \forall h^{(2)} \in H^{(2)}$$

$$d_\mu(s,a) \geq 0 \qquad \forall (s,a) \in SA$$

$$d_\mu(s,a) \leq 1 \qquad \forall (s,a) \in SA$$

Two Types of Algorithms to Check the Covering Requirement Exactly

Here we replicate the same steps of the previous section to derive two types of algorithmic solutions to the problem of checking the covering requirement exactly. Most of the comments provided in Section 4.3.1 apply verbatim.

Algorithm of the 1st Type

We present similar algorithmic components as before.

Finding the Neighbors of One Representative Policy In an $|SA|$-dimensional polytope, we consider different selections of $|SA|$ members from our $\big(|K|-1\big)$ representative policies and subpolytope sides set H. Now, for these different selections we have to solve different systems of equations. The number of systems is equal to $\binom{|K|-1+|H|}{|SA|}$. As usual, we differentiate between systems of type A and B.

- **System of Type A**

  All of the arguments made in Section 4.3.1 can be replicated here.

- **System of Type B**

  In the selections in which both representative policies and subpolytope sides are involved, we say the system is of type B. By defining the set of systems Y and its members y by

  $$y = \{|SA| \textit{ members from } \big(|K|-1\big) \textit{ representative policies and subpolytope sides set } H \textit{ on condition of at least one and at most } \big(|SA|-1\big) \textit{ members selected from subpolytope sides}\}$$

  We can say the number $|Y|$ of systems to be solved is at most $\binom{|K|-1+|H|}{|SA|} - \binom{|K|-1}{|SA|}$.

  For each y, we further define the sets K' and H' as:

  $$K' = \{|K'| \textit{ members } k'\textit{'s from } \big(|K|-1\big) \textit{ representative policies}\}$$

  $$H' = \{|H'| \textit{ members } h'\textit{'s from subpolytope sides set } H\}$$

  where $|k'|+|H'| = |SA|$ and $|H'|$ is at least equal to 1 and at most equal to $\big(|SA|-1\big)$.

  For each y and its corresponding sets K' and H' we have to solve a system with $|SA|$ quadratic equations and $|SA|$ unknowns as follows

  $$\int \frac{d^2_{\mu_y}(s,a)}{d_{\theta_{k'}}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \sigma \qquad \forall k' \in K'$$

$$\int a_{h'}(s,a) \cdot d_{\mu_y}(s,a)\ \mathrm{d}s\ \mathrm{d}a = b_{h'} \qquad \forall h' \in H'$$

Then, for each one of the systems above we see whether the policy found by that system, i.e., $\mu^\star_y$, is within the subpolytope and whether it is beyond a σ distance from all of the other representative policies and within a σ distance from the representative policy j. In other words, we check whether all of four conditions below hold true:

$$1 \cdot \quad \int a_{h^{(1)}}(s,a) \cdot d_{\mu^\star_y}(s,a)\ \mathrm{d}s\ \mathrm{d}a \geq b_{h^{(1)}} \qquad \forall h^{(1)} \in H^{(1)}$$

$$2 \cdot \quad \int a_{h^{(2)}}(s,a) \cdot d_{\mu^\star_y}(s,a)\ \mathrm{d}s\ \mathrm{d}a \leq b_{h^{(2)}} \qquad \forall h^{(2)} \in H^{(2)}$$

$$3 \cdot \quad \int \frac{d^2_{\mu^\star_y}(s,a)}{d_{\theta_k}(s,a)}\ \mathrm{d}s\ \mathrm{d}a > \sigma \qquad \forall k \in K \mid \{k = j, k \in K'\}$$

$$4 \cdot \quad \int \frac{d^2_{\mu^\star_y}(s,a)}{d_{\theta_j}(s,a)}\ \mathrm{d}s\ \mathrm{d}a \leq \sigma$$

If the answer is positive, then all the related $|SA|$ representative policies and subpolytope sides are in the neighborhood of our representative policy j.

Checking the Covering Requirement Locally As in the previous section, we have to answer the following question: Is there any remaining policy among our representative policy j and its neighboring representative policies and subpolytope sides that is farther from a σ distance from them?

To answer this question we consider different selections of $|SA|$ members from our representative policies set K' and subpolytope sides set H'. For these different selections, we have to solve different systems of equations. The number of systems is equal to $\binom{|K'|+|H'|}{|SA|-1}$, which we differentiate between systems of type A and B.

- **System of Type A**

  All of the arguments made in Section 4.3.1 can be replicated here.

- **A System of Type B**

  In the selections in which both representative policies and subpolytope sides are involved, we say the system is of type B. We define the set of such systems as Y and its members y as:

  $y = \{(|SA| - 1)$ *members from representative policies set* K' *and subpolytope sides* *set H' on condition of at least one member selected from subpolytope sides*$\}$

We can say the number of systems $|Y|$ to be solved is equal to $\binom{|K'|+|H'|}{|SA|-1} - \binom{|K'|}{|SA|-1}$.

For each $y \in Y$, we further define the sets K'' and H'' as

$$K'' = \{|K''| \textit{ members } k'' \textit{'s from representative policies set } K'\}$$

$$H'' = \{|H''| \textit{ members } h'' \textit{'s from subpolytope sides set } H'\}$$

where $|k''| + |H''| = |SA| - 1$ and $|H''|$ is at least equal to 1.

We introduce another unknown denoted as σ_y instead of the value of σ. For each $y \in Y$ and its corresponding sets K'' and H'', we have to solve a system with $\big(|SA|+1\big)$ linear and quadratic equations and $\big(|SA|+1\big)$ unknowns:

$$\int \frac{d^2_{\mu^j_y}(s,a)}{d_{\theta_j}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \sigma^j_y$$

$$\int \frac{d^2_{\mu^j_y}(s,a)}{d_{\theta_{k''}}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \sigma^j_y \qquad \forall k'' \in K''$$

$$\int a_{h''}(s,a) \cdot d_{\mu^j_y}(s,a) \,\mathrm{d}s\,\mathrm{d}a = b_{h''} \qquad \forall h'' \in H''$$

$$\int d_{\mu^j_y}(s,a) \,\mathrm{d}s\,\mathrm{d}a = 1$$

If there is at least one system among the systems above for which we get a policy within our subpolytope space and a value greater than σ for our unknown σ^j_y or, in other words, if all of three conditions below hold true[4]

$$1\cdot \quad \int a_{h^{(1)}}(s,a) \cdot \boldsymbol{d}_{\boldsymbol{\mu}^j_y}(s,a) \,\mathrm{d}s\,\mathrm{d}a \geq b_{h^{(1)}} \qquad \forall h^{(1)} \in H^{(1)}$$

$$2\cdot \quad \int a_{h^{(2)}}(s,a) \cdot \boldsymbol{d}_{\boldsymbol{\mu}^j_y}(s,a) \,\mathrm{d}s\,\mathrm{d}a \leq b_{h^{(2)}} \qquad \forall h^{(2)} \in H^{(2)}$$

$$3\cdot \quad \boldsymbol{\sigma}^j_y > \sigma$$

it means there is at least one remaining policy left uncovered. If no system satisfies the conditions above, it means there is no remaining policy left uncovered and we can stop.

[4]let's consider $\boldsymbol{\sigma}^j_y$'s as the values of unknowns σ^j_y's and $\boldsymbol{d}_{\boldsymbol{\mu}^j_y}$'s as the values of state action pairs distributions of different remaining policies $d_{\mu^j_y}$ found by different systems.

In the former case, in order to find the exact value of our original maximin problem, we see for which one of those systems we have reached the maximum value for our unknown σ_x^j and we save it as

$$\boldsymbol{\sigma}_{\boldsymbol{y}^\star}^{j} = \max_{\boldsymbol{y}} \boldsymbol{\sigma}_{\boldsymbol{y}}^{j}$$

and we also save the remaining policy found by that related system $\boldsymbol{d}_{\mu_{\boldsymbol{y}^\star}^{j}}(\boldsymbol{s}, \boldsymbol{a})$.

Checking the Covering Requirement Globally Then, checking the covering requirement globally can be done exactly as in Section 4.3.1.

An Alternative Algorithm to Find the Neighbors of One Representative Policy Similarly, we can extend the arguments in Section 4.3.1 to derive the alternative algorithm below.

Algorithm 4.2 The Alternative Algorithm

```
 1: BEGIN
 2: $|K'| := 0;$
 3: $|H'| := 0;$
 4: $K' := \{j\};$
 5: $H' := \emptyset;$
 6: for $k \in K \mid \{k = j\}$ do
 7:     if $\int \frac{D_{M^\star_{\theta_j}}(s,a)}{d_{\theta_k}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \sigma$ then
 8:         $|K'| := |K'| + 1;$
 9:         $K' := K' \cup \{k\};$
10:     end if
11: end for
12: for $h \in H$ do
13:     if $\int a_h(s,a) \cdot \sqrt{D_{M^\star_{\theta_j}}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = b_h$ then
14:         $|H'| := |H'| + 1;$
15:         $H' := H' \cup \{h\};$
16:     end if
17: end for
18: END
```

Algorithm of the 2nd Type

Finally, we extend the "algorithm of the 2nd type" to the subpolytope setting. We consider different selections of $|SA|$ members from our $\big(|K| - 1\big)$ representative policies and subpolytope sides set H. For these different selections we have to solve different systems of equations. The number of systems is at most $\binom{|K|+|H|}{|SA|}$, which we differentiate as systems of type A and type B.

- **System of Type A**

  The same as in Section 4.3.1.

- **System of Type B**

  In the selections in which both representative policies and subpolytope sides are involved, we say the system is of type B. By defining the systems set Y and its members y as:

  $$y = \{|SA| \textit{ members from } |K| \textit{ representative policies and subpolytope sides set } H \\ \textit{on condition of at least one and at most } \big(|SA|-1\big) \textit{ members selected from} \\ \textit{subpolytope sides}\}$$

  We can say the number of systems $|Y|$ to be solved is equal to $\binom{|K|+|H|}{|SA|} - \binom{|K|}{|SA|}$.

  We further define the sets K' and H' as

  $$K' = \{|K'| \textit{ members } k'\textit{'s from representative policies set } K\}$$

  $$H' = \{|H'| \textit{ members } h'\textit{'s from subpolytope sides set } H\}$$

  where $|k'|+|H'| = |SA|$ and $|H'|$ is at least equal to 1 and at most equal to $\big(|SA|-1\big)$.

  We introduce another unknown denoted as σ_y instead of the value of σ. For every $y \in Y$ and corresponding sets K' and H', we have to solve a system with $\big(|SA|+1\big)$ quadratic equations and $\big(|SA|+1\big)$ unknowns as follows:

  $$\int \frac{d^2_{\mu_y}(s,a)}{d_{\theta_{k'}}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \sigma_y \qquad \forall k' \in K'$$

  $$\int a_{h'}(s,a) \cdot d_{\mu_y}(s,a) \,\mathrm{d}s\,\mathrm{d}a = b_{h'} \qquad \forall h' \in H'$$

  $$\int d_{\mu_y}(s,a) \,\mathrm{d}s\,\mathrm{d}a = 1$$

  Finally, for each of the systems above where the value of variable σ_y is greater than σ, we must see if there is any other representative policy in set K that the policy found by the related system $\mu^\star_y$ that is within the σ distance from it or not:

  $$1 \cdot \int a_{h^{(1)}}(s,a) \cdot d_{\mu^\star_y}(s,a) \,\mathrm{d}s\,\mathrm{d}a \geq b_{h^{(1)}} \qquad \forall h^{(1)} \in H^{(1)}$$

$$2 \cdot \quad \int a_{h^{(2)}}(s,a) \cdot d_{\mu_y^\star}(s,a)\,\mathrm{d}s\,\mathrm{d}a \leq b_{h^{(2)}} \qquad \forall h^{(2)} \in H^{(2)}$$

$$3 \cdot \quad \int \frac{d^2_{\mu_y^\star}(s,a)}{d_{\theta_k}(s,a)}\,\mathrm{d}s\,\mathrm{d}a \leq \sigma \qquad \forall k \in K \mid \{k \in K'\}$$

5 | On the Number of Covering Policies

In this chapter, we derive lower and upper bounds on the number of covering policies needed to solve the policy space compression problem for a given σ.

The main idea is to find out how much space of the polytope is covered by one representative policy, from which we can directly compute the number of covering policies that are needed. We will show that the largest space will be covered by representative policies close to the center of the polytope (i.e., inducing state-action distributions close to uniform) and the smallest spaces will be covered by representative policies close to the edges of the polytope (i.e., inducing nearly deterministic state-action distribution).

First, in Section 5.1, we warm-up the reader on a simple yet illustrative example of covering in a two-dimensional polytope. Next, in Section 5.2, we revise a few fundamentals to derive our lower and upper bounds, which are discussed in Section 5.3 and 5.4 respectively.

5.1. Warm-Up: A Two-Dimensional Polytope

As a warm-up to derive more general bounds, here we provide a study of the illustrative setting in which the polytope is two-dimensional.

The main idea is to find the proportion of the space covered by adjacent representative policies, and then to compute the number of adjacent representative policies that can cover all of the space of the polytope. We say that representative policies are adjacent if there can be found a common policy at a σ distance from all of them.

Let us consider a representative policy with uniform state-action distribution and let us see what happens if we move away from this policy by a σ distance. We have:

$$\int \frac{d_\mu^2(s,a)}{d_\theta(s,a)} \,\mathrm{d}s\, \mathrm{d}a = \frac{d_\mu^2(s,a)_1}{\frac{1}{2}} + \frac{\big(1 - d_\mu(s,a)_1\big)^2}{\frac{1}{2}} = \sigma$$

$$\Rightarrow d^2_\mu(s,a)_1 - d_\mu(s,a)_1 - \frac{\sigma}{4} + \frac{1}{2} = 0$$

$$\Rightarrow d_\mu(s,a)_1 = \frac{1 \pm \sqrt{1 - 4\big(-\frac{\sigma}{4} + \frac{1}{2}\big)}}{2}$$

$$\Rightarrow d_\mu(s,a)_1 = \frac{1 \pm \sqrt{\sigma - 1}}{2}$$

and thus the policies are:

$$\left(\frac{1+\sqrt{\sigma-1}}{2}, \frac{1-\sqrt{\sigma-1}}{2}\right)$$
$$\left(\frac{1-\sqrt{\sigma-1}}{2}, \frac{1+\sqrt{\sigma-1}}{2}\right)$$

Now, by computing the Euclidean distance between these two points we can measure the space covered by our representative policy:

$$\sqrt{\left(\frac{1+\sqrt{\sigma-1}}{2} - \frac{1-\sqrt{\sigma-1}}{2}\right)^2 + \left(\frac{1+\sqrt{\sigma-1}}{2} - \frac{1-\sqrt{\sigma-1}}{2}\right)^2}$$
$$= \sqrt{\boldsymbol{2(\sigma - 1)}}$$

Let us consider one of the previous policies. Based on the definition of adjacent representative policies, let us try to find the other representative policies which are at a σ distance from this policy:

$$\int \frac{d^2_\mu(s,a)}{d_\theta(s,a)} \, \mathrm{d}s \, \mathrm{d}a = \frac{\left(\frac{1+\sqrt{\sigma-1}}{2}\right)^2}{d_\theta(s,a)_1} + \frac{\left(\frac{1-\sqrt{\sigma-1}}{2}\right)^2}{1 - d_\theta(s,a)_1} = \sigma$$

$$\Rightarrow \sigma d^2_\theta(s,a)_1 - \big(\sigma + \sqrt{\sigma-1}\big) d_\theta(s,a)_1 + \left(\frac{1+\sqrt{\sigma-1}}{2}\right)^2 = 0$$

$$\Rightarrow d_\theta(s,a)_1 = \frac{\sigma + \sqrt{\sigma-1} \pm \sqrt{\big(\sigma + \sqrt{\sigma-1}\big)^2 - 4\sigma\left(\frac{1+\sqrt{\sigma-1}}{2}\right)^2}}{2\sigma}$$

$$\Rightarrow d_\theta(s,a)_1 = \frac{1}{2} \pm \frac{\sqrt{\sigma-1}}{\sigma}$$

Let us repeat the steps above again to see what happens if we move away from this policy by a σ distance. We have:

$$\int \frac{d_\mu^2(s,a)}{d_\theta(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \frac{d_\mu^2(s,a)_1}{\frac{1}{2}+\frac{\sqrt{\sigma-1}}{\sigma}} + \frac{\big(1-d_\mu(s,a)_1\big)^2}{\frac{1}{2}-\frac{\sqrt{\sigma-1}}{\sigma}} = \sigma$$

$$\Rightarrow d_\mu^2(s,a)_1 - \left(1+\frac{2\sqrt{\sigma-1}}{\sigma}\right)d_\mu(s,a)_1$$
$$+\left(\frac{1}{2}+\frac{\sqrt{\sigma-1}}{\sigma}\right)\left(1-\sigma\left(\frac{1}{2}-\frac{\sqrt{\sigma-1}}{\sigma}\right)\right) = 0$$

$$\Rightarrow d_\mu(s,a)_1 = \frac{1+\frac{2\sqrt{\sigma-1}}{\sigma}}{2}$$
$$\pm\frac{\sqrt{\left(1+\frac{2\sqrt{\sigma-1}}{\sigma}\right)^2 - 4\big(\frac{1}{2}+\frac{\sqrt{\sigma-1}}{\sigma}\big)\left(1-\sigma\big(\frac{1}{2}-\frac{\sqrt{\sigma-1}}{\sigma}\big)\right)}}{2}$$

$$\Rightarrow d_\mu(s,a)_1 = \frac{1}{2}+\frac{\sqrt{\sigma-1}}{\sigma} \pm \frac{1}{2}\sqrt{\frac{-4\big(\sigma-1\big)^2}{\sigma^2}+\sigma-1}$$

and the resulting policies are:

$$\left(\frac{1}{2}+\frac{\sqrt{\sigma-1}}{\sigma}+\frac{1}{2}\sqrt{\frac{-4\big(\sigma-1\big)^2}{\sigma^2}+\sigma-1},\frac{1}{2}-\frac{\sqrt{\sigma-1}}{\sigma}-\frac{1}{2}\sqrt{\frac{-4\big(\sigma-1\big)^2}{\sigma^2}+\sigma-1}\right)$$

$$\left(\frac{1}{2}+\frac{\sqrt{\sigma-1}}{\sigma}-\frac{1}{2}\sqrt{\frac{-4\big(\sigma-1\big)^2}{\sigma^2}+\sigma-1},\frac{1}{2}-\frac{\sqrt{\sigma-1}}{\sigma}+\frac{1}{2}\sqrt{\frac{-4\big(\sigma-1\big)^2}{\sigma^2}+\sigma-1}\right)$$

After simplification, one of these points is exactly the same point we have already considered. Thus, there is a common point at the σ distance away from our so far two adjacent representative policies:

$$\left(\frac{1}{2}+\frac{\sqrt{\sigma-1}}{\sigma}+\frac{1}{2}\sqrt{\frac{-4\big(\sigma-1\big)^2}{\sigma^2}+\sigma-1},\frac{1}{2}-\frac{\sqrt{\sigma-1}}{\sigma}-\frac{1}{2}\sqrt{\frac{-4\big(\sigma-1\big)^2}{\sigma^2}+\sigma-1}\right)$$
$$=\left(\frac{1+\sqrt{\sigma-1}}{2},\frac{1-\sqrt{\sigma-1}}{2}\right)$$

Then, by computing the Euclidean distance between these two points we can measure the space covered by the latter representative policy:

$$\sqrt{\begin{aligned}&\left(\left(\frac{1}{2}+\frac{\sqrt{\sigma-1}}{\sigma}+\frac{1}{2}\sqrt{\frac{-4(\sigma-1)^2}{\sigma^2}+\sigma-1}\right)\right.\\&\left.-\left(\frac{1}{2}+\frac{\sqrt{\sigma-1}}{\sigma}-\frac{1}{2}\sqrt{\frac{-4(\sigma-1)^2}{\sigma^2}+\sigma-1}\right)\right)^2\\&+\left(\left(\frac{1}{2}-\frac{\sqrt{\sigma-1}}{\sigma}-\frac{1}{2}\sqrt{\frac{-4(\sigma-1)^2}{\sigma^2}+\sigma-1}\right)\right.\\&\left.-\left(\frac{1}{2}-\frac{\sqrt{\sigma-1}}{\sigma}+\frac{1}{2}\sqrt{\frac{-4(\sigma-1)^2}{\sigma^2}+\sigma-1}\right)\right)^2\end{aligned}}$$

$$=\boldsymbol{\frac{2-\sigma}{\sigma}\sqrt{2(\sigma-1)}}$$

Hence, the proportion of the polytope covered by two adjacent representative policies is as follows:

$$\frac{\frac{2-\sigma}{\sigma}\sqrt{2(\sigma-1)}}{\sqrt{2(\sigma-1)}}=\boldsymbol{\frac{2-\sigma}{\sigma}}$$

In order to determine how many representative policies we need to cover our original polytope, which here is a line of length $\sqrt{2}$, we have to solve the following equation:

$$\sqrt{2(\sigma-1)}+\sum_{i=1}^{n}2\left(\frac{2-\sigma}{\sigma}\right)^i\sqrt{2(\sigma-1)}=\sqrt{2}$$

Since in a two-dimensional polytope the σ value is less or equal to 2, $\frac{2-\sigma}{\sigma}$ is less than 1. We have:

$$\sqrt{\sigma-1}+2\sqrt{\sigma-1}\left(\frac{\frac{2-\sigma}{\sigma}-\left(\frac{2-\sigma}{\sigma}\right)^{n+1}}{1-\frac{2-\sigma}{\sigma}}\right)=1$$

$$\Rightarrow\left(\frac{2-\sigma}{\sigma}\right)^{n+1}=\frac{1-\sqrt{\sigma-1}}{\sigma}$$

$$\Rightarrow n=\left(\log_{\left(\frac{2-\sigma}{\sigma}\right)}\left(\frac{1-\sqrt{\sigma-1}}{\sigma}\right)\right)-1$$

and we can say:

$$N = 2\lceil n \rceil + 1$$

where N is an estimation for the needed number of representative policy up to an error of 1. Finally, we have:

$$N = 2\left\lceil \log_{\left(\frac{2-\sigma}{\sigma}\right)}\left(\frac{1-\sqrt{\sigma-1}}{\sigma}\right)\right\rceil - 1$$

5.2. Fundamentals

Here we provide a few fundamentals that will be useful in the following derivations.

5.2.1. The Volume of A Polytope

We can find the volume of an $|SA|$-dimensional polytope using the formula below:

$$V_{|SA|} = \left(\frac{1}{|SA|-1}\left(\frac{1}{|SA|-2}\cdots\left(\frac{1}{3}\left(\frac{1}{2}(a)h_3\right)h_4\right)\cdots h_{|SA|-1}\right)h_{|SA|}\right)$$

where

$$a = \sqrt{(1-0)^2 + (0-1)^2 + (0-0)^2} = \sqrt{2}$$

and

$$h_3 = \sqrt{(1-0)^2 + \left(0-\frac{1}{2}\right)^2 + \left(0-\frac{1}{2}\right)^2} = \sqrt{\frac{3}{2}}$$

$$h_4 = \sqrt{(1-0)^2 + \left(0-\frac{1}{3}\right)^2 + \left(0-\frac{1}{3}\right)^2 + \left(0-\frac{1}{3}\right)^2} = \sqrt{\frac{4}{3}}$$

$$\cdots$$

$$h_{|SA|} = \sqrt{(1-0)^2 + \left(0-\frac{1}{|SA|-1}\right)^2 + \cdots + \left(0-\frac{1}{|SA|-1}\right)^2} = \sqrt{\frac{|SA|}{|SA|-1}}$$

Thus, the volume of the polytope is

$$V_{|SA|} = \frac{1}{(|SA|-1)!}\cdot\sqrt{2}\cdot\sqrt{\frac{3}{2}}\cdot\sqrt{\frac{4}{3}}\cdots\sqrt{\frac{|SA|}{|SA|-1}} = \boldsymbol{\frac{\sqrt{|SA|}}{(|SA|-1)!}}$$

5.2.2. Volume of a Tetrahedral Subpolytope

The volume of a tetrahedral subpolytope of the original $|SA|$-dimensional polytope can be computed as follows:

$$V'_{|SA|} = \frac{1}{(|SA|-1)!} \cdot (\gamma \cdot \sqrt{2}) \cdot \left(\gamma \cdot \sqrt{\frac{3}{2}}\right) \cdot \left(\gamma \cdot \sqrt{\frac{4}{3}}\right) \cdots \left(\gamma \cdot \sqrt{\frac{|SA|}{|SA|-1}}\right)$$

$$= \frac{\sqrt{|SA|}}{(|SA|-1)!} \cdot \gamma^{|SA|-1}$$

where γ is the proportion of subpolytope edges to our original polytope edges, which gives

$$\boldsymbol{\frac{V'_{|SA|}}{V_{|SA|}} = \gamma^{|SA|-1}}$$

5.2.3. Looking at the Polytope from One Face

Let us consider the range of σ greater or equal to $\frac{|SA|}{|SA|-1}$. Let assume the polytope has been covered with a sufficient number of representative policies and we are looking at the polytope from one of its faces of $|SA|-1$ dimensions. If the number of observed representative policies are equal to n, then it can be shown that the needed number of representative policies will be

$$\left(1 + \left(\frac{1}{2}\right)^{|SA|-1}\right) n.$$

This comes directly from the fact that, by looking at the polytope of $|SA|$ dimensions from one of its faces of $|SA|-1$ dimensions, a tetrahedral subpolytope of the original polytope is unobservable. Consequently, we must add the number of representative policies covering that specific space to the number of observed representative policies. Based on the formulation derived in previous section 5.2.2, the number of unobservable representative policies is equal to

$$\left(\frac{1}{2}\right)^{|SA|-1} n.$$

By generalizing the concept above, if we look at the polytope from one of its faces of λ dimensions, and the number of observed representative policies are equal to n, then it can be shown that the needed number of representative policies will be

$$\Bigg(1+\left(\frac{1}{2}\right)^{\boldsymbol{\lambda-1}}+\left(\frac{1}{2}\right)^{\boldsymbol{\lambda}}\left(1+\left(\frac{1}{2}\right)^{\lambda-1}\right)$$
$$+\left(\frac{1}{2}\right)^{\boldsymbol{\lambda+1}}\left(1+\left(\frac{1}{2}\right)^{\lambda-1}+\left(\frac{1}{2}\right)^{\lambda}\left(1+\left(\frac{1}{2}\right)^{\lambda-1}\right)\right)+\ldots\Bigg)n$$

where it goes on until the exponent of base $\frac{1}{2}$ increase from $\lambda-1$ to $|SA|-1$.

The coefficient above can be computed running the following algorithm.

Algorithm 5.1

```
1: BEGIN
2:         $i := \lambda - 1$;
3:         $F(\lambda) := 1 + \left(\frac{1}{2}\right)^{\lambda-1}$;
4: while $i < |SA| - 1$ do
5:           $i := i + 1$;
6:           $F(\lambda) := F(\lambda) + \left(\frac{1}{2}\right)^{i} F(\lambda)$;
7: end while
8:         PRINT $F(\lambda)$;
9: END
```

We can give some instances of the coefficients. If $|\boldsymbol{SA}|-\boldsymbol{\lambda}=\mathbf{1}$, then $F(\lambda)$ will be:

$$\boldsymbol{F(\lambda)=1+2\left(\frac{1}{2}\right)^{\lambda}}$$

If $|\boldsymbol{SA}|-\boldsymbol{\lambda}=\mathbf{2}$, then $F(\lambda)$ will be:

$$\boldsymbol{F(\lambda)=1+2\left(\frac{1}{2}\right)^{2\lambda}+3\left(\frac{1}{2}\right)^{\lambda}}$$

If $|\boldsymbol{SA}|-\boldsymbol{\lambda}=\mathbf{3}$, then $F(\lambda)$ will be:

$$\boldsymbol{F(\lambda)=1+\left(\frac{1}{2}\right)^{3\lambda}+\frac{7}{2}\left(\frac{1}{2}\right)^{2\lambda}+\frac{7}{2}\left(\frac{1}{2}\right)^{\lambda}}$$

If $|\boldsymbol{SA}| - \boldsymbol{\lambda} = \mathbf{4}$, then $F(\lambda)$ will be:

$$\boldsymbol{F(\lambda) = 1 + \frac{1}{4}\left(\frac{1}{2}\right)^{4\lambda} + \frac{15}{8}\left(\frac{1}{2}\right)^{3\lambda} + \frac{35}{8}\left(\frac{1}{2}\right)^{2\lambda} + \frac{15}{4}\left(\frac{1}{2}\right)^{\lambda}}$$

With some approximation, it can be shown that if $|\boldsymbol{SA}| - \boldsymbol{\lambda} \geq \mathbf{5}$, then $F(\lambda)$ will be:

$$\boldsymbol{F(\lambda) = 1 + \frac{1}{32}\left(\frac{1}{2}\right)^{5\lambda} + \frac{31}{64}\left(\frac{1}{2}\right)^{4\lambda} + \frac{155}{64}\left(\frac{1}{2}\right)^{3\lambda} + \frac{155}{32}\left(\frac{1}{2}\right)^{2\lambda} + \frac{31}{8}\left(\frac{1}{2}\right)^{\lambda}}$$

5.3. The Lower Bound

In order to estimate the lower bound for the needed number of representative policies, we first determine the portion of the polytope covered by a representative policy θ_1 with uniform state-action distribution.

First, we derive its radius, finding one of the policies at the σ distance from our representative policy θ_1. We can move away from representative policy θ_1 in the direction of a specific vertex, for instance $(s,a)_1$, by considering

$$d_\mu(s,a)_2 = d_\mu(s,a)_3 = \cdots = d_\mu(s,a)_{|SA|} = \frac{1 - d_\mu(s,a)_1}{|SA| - 1}$$

which results in

$$\int \frac{d_\mu^2(s,a)}{d_{\theta_1}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \frac{d_\mu^2(s,a)_1}{\frac{1}{|SA|}} + \frac{\left(\frac{1-d_\mu(s,a)_1}{|SA|-1}\right)^2}{\frac{1}{|SA|}} + \cdots + \frac{\left(\frac{1-d_\mu(s,a)_1}{|SA|-1}\right)^2}{\frac{1}{|SA|}} = \sigma$$

$$\Rightarrow \frac{d_\mu^2(s,a)}{\frac{1}{|SA|}} + (|SA|-1)\frac{\left(\frac{1-d_\mu(s,a)_1}{|SA|-1}\right)^2}{\frac{1}{|SA|}} = \sigma$$

$$\Rightarrow |SA|d_\mu^2(s,a)_1 - 2d_\mu(s,a)_1 - \frac{\sigma(|SA|-1)}{|SA|} + 1 = 0$$

$$\Rightarrow d_\mu(s,a)_1 = \frac{1 \pm \sqrt{\big(|SA|-1\big)(\sigma-1)}}{|SA|}$$

$$\Rightarrow d_\mu(s,a)_2 = d_\mu(s,a)_3 = \cdots = d_\mu(s,a)_{|SA|} = \frac{1 - \left(\frac{1\pm\sqrt{\big(|SA|-1\big)(\sigma-1)}}{|SA|}\right)}{|SA|-1}$$

Now, we compute the Euclidean distance between our representative policy θ_1 and the previously found policy $d_\mu(s,a)$. We have

$$\sqrt{\begin{array}{l}\left(\frac{1+\sqrt{\big(|SA|-1\big)(\sigma-1)}}{|SA|}-\frac{1-\sqrt{\big(|SA|-1\big)(\sigma-1)}}{|SA|}\right)^2 \\ +(|SA|-1)\left(\frac{1-\left(\frac{1+\sqrt{\big(|SA|-1\big)(\sigma-1)}}{|SA|}\right)}{|SA|-1}-\frac{1-\left(\frac{1-\sqrt{\big(|SA|-1\big)(\sigma-1)}}{|SA|}\right)}{|SA|-1}\right)^2\end{array}}=2\sqrt{\frac{\sigma-1}{|SA|}}$$

which means the radius R of the spherical space covered by our representative policy θ_1 will be

$$R=\sqrt{\boldsymbol{\frac{\sigma-1}{|SA|}}}$$

Then, the volume of the space covered by our representative policy θ_1 can be computed as

$$V^{\theta_1}_{|SA|}=\frac{\pi^{\frac{|SA|-1}{2}}}{\Gamma\big(1+\frac{|SA|-1}{2}\big)}R^{|SA|-1}$$

where Γ is the Gamma function.

Hence, the lower bound for the needed number of representative policies is

$$N>\frac{V_{|SA|}}{V^{\theta_1}_{|SA|}}$$

If $|SA|$ is even we have:

$$N>\left(\frac{\sqrt{|SA|}}{\big(\pi\big)^{\frac{|SA|-1}{2}}}\right)\left(\frac{\left(\frac{|SA|-1}{2}\right)!}{(|SA|-1)!}\right)\left(\frac{|SA|}{\sigma-1}\right)^{\frac{|SA|-1}{2}}$$

If $|SA|$ is odd we have:

$$N>\left(\frac{\sqrt{|SA|}}{\pi^{\left\lfloor\frac{|SA|-1}{2}\right\rfloor}2^{\left\lceil\frac{|SA|-1}{2}\right\rceil}}\right)\left(\frac{(|SA|-1)!!}{(|SA|-1)!}\right)\left(\frac{|SA|}{\sigma-1}\right)^{\frac{|SA|-1}{2}}$$

where !! is the double factorial. Using the Stirling's approximation we derive

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n+1}}<n!<\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}$$

$$N > \sqrt{\frac{|SA|}{2}}\left(\frac{e|SA|}{2\pi(\sigma-1)(|SA|-1)}\right)^{\frac{|SA|-1}{2}}$$

$$\sqrt{\frac{|SA|}{2}}\left(\frac{e|SA|}{2\pi(\sigma-1)(|SA|-1)}\right)^{\frac{|SA|-1}{2}} > \sqrt{\frac{|SA|}{2}}\left(\frac{e}{2\pi(\sigma-1)}\right)^{\frac{|SA|-1}{2}}$$

which results in

$$\boldsymbol{N > \sqrt{\frac{|SA|}{2}}\left(\frac{e}{2\pi(\sigma-1)}\right)^{\frac{|SA|-1}{2}}}$$

The latter results works for a special range of σ values. The latter range is given by

$$\int \frac{d_\mu^2(s,a)}{d_{\theta_1}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \frac{(0)^2}{\frac{1}{|SA|}} + \frac{\left(\frac{1}{|SA|-1}\right)^2}{\frac{1}{|SA|}} + \cdots + \frac{\left(\frac{1}{|SA|-1}\right)^2}{\frac{1}{|SA|}}$$

$$\Rightarrow \frac{d_\mu^2(s,a)}{d_{\theta_1}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \big(|SA|-1\big)\frac{\left(\frac{1}{\alpha}\right)^2}{\frac{1}{|SA|}}$$

$$\Rightarrow \frac{d_\mu^2(s,a)}{d_{\theta_1}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \frac{|SA|}{|SA|-1}$$

so for the σ values that lie within the interval

$$\boldsymbol{\left(1, \frac{|SA|}{|SA|-1}\right]}.$$

To generalize the bound above we can follow an alternative route. We have

$$\int \frac{d_\mu^2(s,a)}{d_{\theta_1}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \frac{(0)^2}{\frac{1}{|SA|}} + \cdots + \frac{(0)^2}{\frac{1}{|SA|}} + \frac{\left(\frac{1}{\lambda}\right)^2}{\frac{1}{|SA|}} + \cdots + \frac{\left(\frac{1}{\lambda}\right)^2}{\frac{1}{|SA|}}$$

$$\Rightarrow \frac{d_\mu^2(s,a)}{d_{\theta_1}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \lambda\frac{\left(\frac{1}{\lambda}\right)^2}{\frac{1}{|SA|}}$$

$$\Rightarrow \frac{d_\mu^2(s,a)}{d_{\theta_1}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \frac{|SA|}{\lambda}$$

which means that, in order to determine the needed number of representative policies, first we determine for which value of λ the given σ will lie within the interval of:

$$\boldsymbol{\left(\frac{|SA|}{\lambda}, \frac{|SA|}{\lambda-1}\right]}$$

where λ can take values from 2 to $|SA|$ and clearly with λ equal to $|SA|$ we obtain the previous interval. The λ value can be computed as follows:

$$\boldsymbol{\lambda = \left\lfloor \frac{|SA|}{\sigma} \right\rfloor + 1}$$

Intuitively, when σ lies within the interval above means that the spherical space of λ dimensions formed at the intersection of spherical space of $|SA|$ dimensions covered by the representative policy θ_1 and a particular polytope face of λ dimensions can be thoroughly accommodated within this particular polytope face.

In order to estimate the lower bound for the needed number of representative policies first we have to determine the portion of the spherical space of λ dimensions covered by the representative policy θ_1. To find the radius, we find one of the policies at the σ distance from our representative policy θ_1 and simultaneously on that particular polytope face. We compute

$$d_\mu(s,a)_2 = d_\mu(s,a)_3 = \cdots = d_\mu(s,a)_\lambda = \frac{1 - d_\mu(s,a)_1}{\lambda - 1}$$

from which we have:

$$\int \frac{d_\mu^2(s,a)}{d_{\theta_1}(s,a)} \,\mathrm{d}s\,\mathrm{d}a = \frac{d_\mu^2(s,a)_1}{\frac{1}{|SA|}} + \frac{\left(\frac{1-d_\mu(s,a)_1}{\lambda-1}\right)^2}{\frac{1}{|SA|}} + \cdots + \frac{\left(\frac{1-d_\mu(s,a)_1}{\lambda-1}\right)^2}{\frac{1}{|SA|}} = \sigma$$

$$\Rightarrow \frac{d_\mu^2(s,a)}{\frac{1}{|SA|}} + (\lambda-1)\frac{\left(\frac{1-d_\mu(s,a)_1}{\lambda-1}\right)^2}{\frac{1}{|SA|}} = \sigma$$

$$\Rightarrow |SA| d_\mu^2(s,a)_1 - 2d_\mu(s,a)_1 - \frac{\sigma(\lambda-1)}{|SA|} + 1 = 0$$

$$\Rightarrow d_\mu(s,a)_1 = \frac{1 \pm \sqrt{\big(\lambda-1\big)(\sigma-1)}}{|SA|}$$

$$\Rightarrow d_\mu(s,a)_2 = d_\mu(s,a)_3 = \cdots = d_\mu(s,a)_{|SA|} = \frac{1 - \left(\frac{1\pm\sqrt{\big(\lambda-1\big)(\sigma-1)}}{|SA|}\right)}{\lambda - 1}$$

Then, computing the Euclidean distance between these two policies $d_\mu(s,a)$ we found, we obtain

$$\frac{\sqrt{\left(\frac{1+\sqrt{(\lambda-1)(\sigma-1)}}{|SA|}-\frac{1-\sqrt{(\lambda-1)(\sigma-1)}}{|SA|}\right)^2}}{+(|SA|-1)\left(\frac{1-\left(\frac{1+\sqrt{(\lambda-1)(\sigma-1)}}{|SA|}\right)}{\lambda-1}-\frac{1-\left(\frac{1-\sqrt{(\lambda-1)(\sigma-1)}}{|SA|}\right)}{\lambda-1}\right)^2}=\frac{2\sqrt{\lambda(\sigma-1)}}{|SA|}$$

Hence, the radius R of the spherical space of λ dimensions served by the representative policy θ_1 will be:

$$R=\frac{\sqrt{\boldsymbol{\lambda(\sigma-1)}}}{\boldsymbol{|SA|}}$$

and the volume of this spherical space can be computed as follows:

$$V_\lambda^{\theta_1}=\frac{\pi^{\frac{\lambda-1}{2}}}{\Gamma\left(1+\frac{\lambda-1}{2}\right)}R^{\lambda-1}$$

where Γ is the gamma function.

Since the volume of the particular polytope face of λ dimensions can be computed as follows:

$$V_\lambda=\frac{\sqrt{\lambda}}{(\lambda-1)!}$$

so the lower bound for the needed number of representative policies will be:

$$N>F(\lambda)\frac{V_\lambda}{V_\lambda^{\theta_1}}$$

where $F(\lambda)$ comes from the reasoning in section 5.2.3.

If λ is even we have:

$$N>F(\lambda)\left(\frac{\sqrt{\lambda}}{(\pi)^{\frac{\lambda-1}{2}}}\right)\left(\frac{\left(\frac{\lambda-1}{2}\right)!}{(\lambda-1)!}\right)\left(\frac{|SA|^2}{(\sigma-1)\lambda}\right)^{\frac{\lambda-1}{2}}$$

If λ is odd we have:

$$N>F(\lambda)\left(\frac{\sqrt{\lambda}}{\pi^{\left\lfloor\frac{\lambda-1}{2}\right\rfloor}2^{\left\lceil\frac{\lambda-1}{2}\right\rceil}}\right)\left(\frac{(\lambda-1)!!}{(\lambda-1)!}\right)\left(\frac{|SA|^2}{(\sigma-1)\lambda}\right)^{\frac{\lambda-1}{2}}$$

where !! is the double factorial. Using the Stirling's approximation, we can show:

$$\boldsymbol{N > F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e|SA|^2}{2\pi(\sigma-1)\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}}}$$

and with λ equal to $|SA|$ the two lower bounds are the same.

By substituting λ value we can also express our lower bound in this form:

$$N > F\left(\left\lfloor\frac{|SA|}{\sigma}\right\rfloor+1\right)\sqrt{\frac{\left\lfloor\frac{|SA|}{\sigma}\right\rfloor+1}{2}}\left(\frac{e|SA|^2}{2\pi(\sigma-1)\left\lfloor\frac{|SA|}{\sigma}\right\rfloor\left(\left\lfloor\frac{|SA|}{\sigma}\right\rfloor+1\right)}\right)^{\frac{\left\lfloor\frac{|SA|}{\sigma}\right\rfloor}{2}}$$

5.4. The Upper Bound

In order to estimate the upper bound for the needed number of representative policies, we first determine the volume of the space covered by a representative policy θ_2 with state action pairs distribution at the same distance from all polytope vertices except one of them, which is a specific vertex (for instance $(s,a)_1$) that our representative policy θ_2 covers at the σ distance.

We aim to compute the radius of the covered space by finding one of the policies at the σ distance from our representative policy θ_2. First, we compute the state-action distribution of our representative policy θ_2. Since

$$\int \frac{d_\mu^2(s,a)}{d_{\theta_2}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \frac{(1)^2}{d_{\theta_2}(s,a)_1}+\frac{(0)^2}{d_{\theta_2}(s,a)_2}+\cdots+\frac{(0)^2}{d_{\theta_2}(s,a)_{|SA|}} = \sigma$$

so $d_{\theta_2}(s,a)_1$ will be equal to $\frac{1}{\sigma}$. For the other state action pairs, we have

$$d_{\theta_2}(s,a)_2 = d_{\theta_2}(s,a)_3 = \cdots = d_{\theta_2}(s,a)_{|SA|}$$

and

$$\int d_{\theta_2}(s,a)\,\mathrm{d}s\,\mathrm{d}a = 1$$

which give

$$\frac{1}{\sigma}+\big(|SA|-1\big)d_{\theta_2}(s,a)_2 = 1 \Rightarrow d_{\theta_2}(s,a)_2 = d_{\theta_2}(s,a)_3 = \cdots = d_{\theta_2}(s,a)_{|SA|} = \frac{\sigma-1}{\sigma(|SA|-1)}$$

Similarly as in previous sections, we move away from representative policy θ_2 in the direction of the specific vertex $(s,a)_1$, by considering:

$$d_\mu(s,a)_2 = d_\mu(s,a)_3 = \cdots = d_\mu(s,a)_{|SA|} = \frac{1-d_\mu(s,a)_1}{|SA|-1}$$

We have

$$\int \frac{d_\mu^2(s,a)}{d_{\theta_2}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \frac{d_\mu^2(s,a)_1}{\frac{1}{\sigma}} + \frac{\left(\frac{1-d_\mu(s,a)_1}{|SA|-1}\right)^2}{\frac{\sigma-1}{\sigma(|SA|-1)}} + \cdots + \frac{\left(\frac{1-d_\mu(s,a)_1}{|SA|-1}\right)^2}{\frac{\sigma-1}{\sigma(|SA|-1)}} = \sigma$$

$$\Rightarrow \frac{d_\mu^2(s,a)}{\frac{1}{\sigma}} + \big(|SA|-1\big)\frac{\left(\frac{1-d_\mu(s,a)_1}{|SA|-1}\right)^2}{\frac{\sigma-1}{\sigma(|SA|-1)}} = \sigma$$

$$\Rightarrow \sigma d_\mu^2(s,a)_1 - 2d_\mu(s,a)_1 + 2 - \sigma = 0$$

$$\Rightarrow d_\mu(s,a)_1 = \frac{2 \pm \sqrt{4-4\sigma(2-\sigma)}}{2\sigma}$$

$$\Rightarrow d_\mu(s,a)_1 = 1 \qquad \text{OR} \qquad d_\mu(s,a)_1 = \frac{2-\sigma}{\sigma}$$

$$\Rightarrow d_\mu(s,a)_2 = d_\mu(s,a)_3 = \cdots = d_\mu(s,a)_{|SA|} = 0 \qquad \text{OR}$$

$$d_\mu(s,a)_2 = d_\mu(s,a)_3 = \cdots = d_\mu(s,a)_{|SA|} = \frac{2(\sigma-1)}{\sigma(|SA|-1)}$$

Computing the Euclidean distance between our representative policy θ_2 and the found policy $d_\mu(s,a)$ we have

$$\sqrt{\left(\frac{2-\sigma}{\sigma}-1\right)^2 + \big(|SA|-1\big)\left(\frac{2(\sigma-1)}{\sigma(|SA|-1)}-0\right)^2}$$

$$= \frac{2(\sigma-1)}{\sigma}\sqrt{\frac{|SA|}{|SA|-1}}$$

Thus, the longest Euclidean distance R covered by our representative policy θ_2 will be:

$$R = \boldsymbol{\frac{(\sigma-1)}{\sigma}\sqrt{\frac{|SA|}{|SA|-1}}}$$

We can also find the policy at the σ distance and simultaneously at the shortest Euclidean distance from our representative policy θ_2. In order to find such policy, we move away from representative policy θ_2 with fixed $d(s,a)_1$ equal to $\frac{1}{\sigma}$ and in the direction of a specific vertex (for instance $(s,a)_2$), by considering

$$d_\mu(s,a)_1 = \frac{1}{\sigma}$$

and

$$d_\mu(s,a)_3 = d_\mu(s,a)_4 = \cdots = d_\mu(s,a)_{|SA|} = \frac{1-\frac{1}{\sigma}-d_\mu(s,a)_2}{|SA|-2}$$

We have

$$\int \frac{d^2_\mu(s,a)}{d_{\theta_2}(s,a)}\,\mathrm{d}s\,\mathrm{d}a = \frac{\left(\frac{1}{\sigma}\right)^2}{\frac{1}{\sigma}} + \frac{d^2_\mu(s,a)_2}{\frac{\sigma-1}{\sigma(|SA|-1)}} + \frac{\left(\frac{1-\frac{1}{\sigma}-d_\mu(s,a)_2}{|SA|-2}\right)^2}{\frac{\sigma-1}{\sigma(|SA|-1)}} + \cdots$$
$$+ \frac{\left(\frac{1-\frac{1}{\sigma}-d_\mu(s,a)_2}{|SA|-2}\right)^2}{\frac{\sigma-1}{\sigma(|SA|-1)}} = \sigma$$

$$\Rightarrow \frac{1}{\sigma} + \frac{d^2_\mu(s,a)_2}{\frac{\sigma-1}{\sigma(|SA|-1)}} + \big(|SA|-2\big)\frac{\left(\frac{1-\frac{1}{\sigma}-d_\mu(s,a)_2}{|SA|-2}\right)^2}{\frac{\sigma-1}{\sigma(|SA|-1)}} = \sigma$$

$$\Rightarrow \frac{\sigma\big(|SA|-1\big)^2}{(\sigma-1)(|SA|-2)}d^2_\mu(s,a)_2 - 2\left(\frac{|SA|-1}{|SA|-2}\right)d_\mu(s,a)_2 + \frac{1}{\sigma}$$
$$+ \frac{(\sigma-1)(|SA|-1)}{\sigma(|SA|-2)} - \sigma = 0$$

$$\Rightarrow d_\mu(s,a)_2 = \frac{2\left(\frac{|SA|-1}{|SA|-2}\right)}{2\frac{\sigma\left(|SA|-1\right)^2}{(\sigma-1)(|SA|-2)}}$$
$$\pm \frac{\sqrt{\left(2\left(\frac{|SA|-1}{|SA|-2}\right)\right)^2 - 4\left(\frac{\sigma(|SA|-1)^2}{(\sigma-1)(|SA|-2)}\right)\left(\frac{1}{\sigma}+\frac{(\sigma-1)(|SA|-1)}{\sigma(|SA|-2)}-\sigma\right)}}{2\frac{\sigma\left(|SA|-1\right)^2}{(\sigma-1)(|SA|-2)}}$$

$$\Rightarrow d_\mu(s,a)_2 = \frac{\frac{1}{|SA|-2} \pm \sqrt{\frac{\sigma}{|SA|-2}}}{\frac{\sigma(|SA|-1)}{(\sigma-1)(|SA|-2)}}$$

$$\Rightarrow d_\mu(s,a)_3 = d_\mu(s,a)_4 = \cdots = d_\mu(s,a)_{|SA|} = \frac{1-\frac{1}{\sigma}-\left(\frac{\frac{1}{|SA|-2}\pm\sqrt{\frac{\sigma}{|SA|-2}}}{\frac{\sigma(|SA|-1)}{(\sigma-1)(|SA|-2)}}\right)}{|SA|-2}$$

Computing the Euclidean distance between our representative policy θ_2 and the found policy $d_\mu(s,a)$, we have

$$\frac{\sqrt{\left(\frac{1}{\sigma}-\frac{1}{\sigma}\right)^2+\left(\frac{\frac{1}{|SA|-2}+\sqrt{\frac{\sigma}{|SA|-2}}}{\frac{\sigma(|SA|-1)}{(\sigma-1)(|SA|-2)}}-\frac{\frac{1}{|SA|-2}-\sqrt{\frac{\sigma}{|SA|-2}}}{\frac{\sigma(|SA|-1)}{(\sigma-1)(|SA|-2)}}\right)^2}}{+\big(|SA|-2\big)\left(\frac{1-\frac{1}{\sigma}-\left(\frac{\frac{1}{|SA|-2}+\sqrt{\frac{\sigma}{|SA|-2}}}{\frac{\sigma(|SA|-1)}{(\sigma-1)(|SA|-2)}}\right)}{|SA|-2}-\frac{1-\frac{1}{\sigma}-\left(\frac{\frac{1}{|SA|-2}-\sqrt{\frac{\sigma}{|SA|-2}}}{\frac{\sigma(|SA|-1)}{(\sigma-1)(|SA|-2)}}\right)}{|SA|-2}\right)^2}$$
$$=\frac{2(\sigma-1)}{\sqrt{\sigma(|SA|-1)}}$$

so the shortest Euclidean distance r covered by our representative policy θ_2 will be

$$r=\boldsymbol{\frac{\sigma-1}{\sqrt{\sigma(|SA|-1)}}}$$

Since we are looking for an upper bound, we can just take the shortest Euclidean distance r into consideration in order to find the space covered by the representative policy θ_2. With this assumption, the volume of the space covered by our representative policy θ_2 can be computed as follows:

$$V^{\theta_2}_{|SA|}=\frac{\pi^{\frac{|SA|-1}{2}}}{\Gamma\big(1+\frac{|SA|-1}{2}\big)}r^{|SA|-1}$$

where Γ is the Gamma function.

Since it holds

$$\frac{V^{\theta_1}_{|SA|}}{V^{\theta_2}_{|SA|}}=\frac{\frac{\pi^{\frac{|SA|-1}{2}}}{\Gamma\left(1+\frac{|SA|-1}{2}\right)}\left(\sqrt{\frac{\sigma-1}{|SA|}}\right)^{|SA|-1}}{\frac{\pi^{\frac{|SA|-1}{2}}}{\Gamma\left(1+\frac{|SA|-1}{2}\right)}\left(\frac{\sigma-1}{\sqrt{\sigma(|SA|-1)}}\right)^{|SA|-1}}=\boldsymbol{\left(\frac{\sigma(|SA|-1)}{(\sigma-1)|SA|}\right)^{\frac{|SA|-1}{2}}}$$

the upper bound of the needed number of representative policies for σ values that lie within the interval of $\left(1,\frac{|SA|}{|SA|-1}\right]$ will be:

$$\boldsymbol{N<2\sqrt{\frac{|SA|}{2}}\left(\frac{e\sigma(|SA|-1)}{2\pi(\sigma-1)^2|SA|}\right)^{\frac{|SA|-1}{2}}}$$

In general the upper bound of the needed number of representative policies for σ values that lie within the interval of $\left(\frac{|SA|}{\lambda}, \frac{|SA|}{\lambda-1}\right]$ will be:

$$\boldsymbol{N < 2F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e\sigma|SA|(|SA|-1)}{2\pi\left(\sigma-1\right)^2\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}}}$$

where coefficient 2 above directly comes from the fact that while we are in search of the upper bound we must also take the overlapped spaces between adjacent representative policies into account. With λ equal to $|SA|$ the two upper bounds are the same.

Finally, by substituting the value of λ, we can also express our upper bound in this form:

$$N < 2F\left(\left\lfloor\frac{|SA|}{\sigma}\right\rfloor+1\right)\sqrt{\frac{\left\lfloor\frac{|SA|}{\sigma}\right\rfloor+1}{2}}\left(\frac{e\sigma|SA|(|SA|-1)}{2\pi\left(\sigma-1\right)^2\left\lfloor\frac{|SA|}{\sigma}\right\rfloor\left(\left\lfloor\frac{|SA|}{\sigma}\right\rfloor+1\right)}\right)^{\frac{\left\lfloor\frac{|SA|}{\sigma}\right\rfloor}{2}}$$

6 | Conclusions

In the upcoming chapter, we provide a concise recap of the primary findings presented in this document and we introduce compelling avenues for future research.

Recap In RL, agents learn policies to maximize rewards. When dealing with large policy spaces, finding optimal policies becomes expensive in terms of required interactions to be taken from the environment. The policy space compression problem [12] aims to identify a smaller subset of policies that retain the same effectiveness as the full set, which involves finding a compressed policy space while ensuring that the most significant state-action distributions can still be induced by the policies in the reduced set. Unfortunately, solving policy space compression is computationally challenging.

In [12], the problem is framed as a game between a leader and a follower. The leader's goal is to cover state-action distributions while the follower aims to find a policy that is not well-covered. The optimization problem involved in this game is also computationally challenging. To solve it, a first-order approximate algorithm operates iteratively, gradually increasing the number of leader-controlled policies to meet a global covering requirement.

In this thesis, we addressed some of the standing issues in the algorithmic procedure presented in [12]. Specifically, we have studied the role of the covering threshold in the policy space compression problem, and how to set it to avoid trivial solutions. Then, we have provided important findings on the number of covering policies needed to solve the policy space compression problem with a certain threshold. Especially, we provided a family of lower and upper bounds on the latter number, which can be used to feed a proper input to the first-order procedure of Mutti et al. [12].

Future Directions Our work is based on the assumption of knowing everything of the MDP, which makes the prblem a computational challenge where no estimation is involved. A future direction may target the corresponding learning problem, in which we can only draw samples from an unknown MDP, and we try to obtain an approximate policy space compression through estimated quantities. Finally, our work is restricted to tabular

MDPs. Future works may target a generalization of the problem and corresponding solutions to a function approximation setting.

Bibliography

[1] A. Bakhtin, D. J. Wu, A. Lerer, J. Gray, A. P. Jacob, G. Farina, A. H. Miller, and N. Brown. Mastering the game of no-press diplomacy via human-regularized reinforcement learning and planning. *arXiv preprint arXiv:2210.05492*, 2022.

[2] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[3] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[4] T. Fiez and L. J. Ratliff. Local convergence analysis of gradient descent ascent with finite timescale separation. In *Proceedings of the International Conference on Learning Representation*, 2021.

[5] T. Fiez, B. Chasnov, and L. Ratliff. Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In *International Conference on Machine Learning*, pages 3133–3144. PMLR, 2020.

[6] C. Jin, P. Netrapalli, and M. Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *International conference on machine learning*, pages 4880–4889. PMLR, 2020.

[7] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976): 982–987, 2023.

[8] V. Konda and J. Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[9] T. L. Lai, H. Robbins, et al. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[10] T. Lattimore and C. Szepesvári. *Bandit algorithms.* Cambridge University Press, 2020.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare,

A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[12] M. Mutti, S. Del Col, and M. Restelli. Reward-free policy space compression for reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3187–3203, 2022.

[13] OpenAI. Chatgpt, 2023. URL `https://chat.openai.com`.

[14] A. B. Owen. Monte carlo theory, methods and examples, 2013.

[15] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[16] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[17] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[18] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[19] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[20] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[21] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[22] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

A | Numerical Examples

A.1. Checking the Covering Requirement

Here we provide a numerical example on the algorithms to check the covering requirement exactly presented in Section 4.3.1. Especially, we presented two types of algorithms.

As previously discussed, booth algorithms run in polynomial time and they allow assess the global guarantee of our representative policies set K but also the exact value of our original maximin problem.

Although unrealistic, for the purpose of this example, $|SA|$ is set equal to 2.

Let us consider our representative policies are as follows:

$$\big\{(0.39, 0.61), (0.94, 0.06), (0.06, 0.94), (0.76, 0.24)\big\}$$

and let us consider our σ value equal to 1.1

Now, let us solve the problem using the first algorithm. First, we have to find other representative policies and polytope sides in the neighborhood of different representative policies. We consider the representative policy (0.39, 0.61) in order to find the neighbors of this representative policy. Since the representative policies in a polytope with $|SA|$ equal to 2 are symmetrically distributed in essence, we can use the alternative algorithm we presented. We have to solve the following optimization problem:

$$\min \quad 2.57 D_M(s,a)_1 + 1.63 D_M(s,a)_2$$

$$s.t.$$

$$1.05\ D_M(s,a)_1 + 18\ D_M(s,a)_2 \geq 1.1$$

$$18\ D_M(s,a)_1 + 1.05\ D_M(s,a)_2 \geq 1.1$$

$$1.32\ D_M(s,a)_1 + 4.09\ D_M(s,a)_2 \geq 1.1$$

$$0 \leq D_M(s,a)_1 \leq 1$$

$$0 \leq D_M(s,a)_2 \leq 1$$

The answer to the problem is 0.53 for $D_M(s,a)_1$ and $D_M(s,a)_2$ respectively equal to 0.04 and 0.25.

Since the active constraints are

$$18\ D_M(s,a)_1 + 1.05\ D_M(s,a)_2 \geq 1.1$$

$$1.32\ D_M(s,a)_1 + 4.09\ D_M(s,a)_2 \geq 1.1$$

the other representative policies in the neighborhood of our representative policy (0.39, 0.61) are (0.06, 0.94) and (0.76, 0.24) Since there is no state-action distribution entry having value equal to zero, there is no polytope side in the neighborhood of our representative policy (0.39, 0.61).

Then, we assess local covering guarantee of our representative policy (0.39, 0.61), for which we have to solve two systems of type A as follows:

$$2.57\ d^2_{\mu^1_1}(s,a)_1 + 1.63\ d^2_{\mu^1_1}(s,a)_2 = \sigma^1_1$$

$$18\ d^2_{\mu^1_1}(s,a)_1 + 1.05\ d^2_{\mu^1_1}(s,a)_2 = \sigma^1_1$$

$$d_{\mu^1_1}(s,a)_1 + d_{\mu^1_1}(s,a)_2 = 1$$

$$\Rightarrow \quad \sigma^1_1 = 1.21,\ d_{\mu^1_1}(s,a)_1 = 0.16,\ d_{\mu^1_1}(s,a)_2 = 0.84$$

and:

$$2.57\ d^2_{\mu^1_2}(s,a)_1 + 1.63\ d^2_{\mu^1_2}(s,a)_2 = \sigma^1_2$$

$$1.32\ d^2_{\mu^1_2}(s,a)_1 + 4.09\ d^2_{\mu^1_2}(s,a)_2 = \sigma^1_2$$

$$d_{\mu^1_2}(s,a)_1 + d_{\mu^1_2}(s,a)_2 = 1$$

$$\Rightarrow \quad \sigma^1_2 = 1.16,\ d_{\mu^1_2}(s,a)_1 = 0.58,\ d_{\mu^1_2}(s,a)_2 = 0.42$$

Now, we have to do the same for other representative policies.

In order to find the neighbors of the representative policy (0.94, 0.06) we have to solve the following optimization problem:

$$\min \quad 1.05\ D_M(s,a)_1 + 18\ D_M(s,a)_2$$

$$s.t.$$

$$2.57\ D_M(s,a)_1 + 1.63\ D_M(s,a)_2 \geq 1.1$$

$$18\ D_M(s,a)_1 + 1.05\ D_M(s,a)_2 \geq 1.1$$

$$1.32\ D_M(s,a)_1 + 4.09\ D_M(s,a)_2 \geq 1.1$$

$$0 \leq D_M(s,a)_1 \leq 1$$

$$0 \leq D_M(s,a)_2 \leq 1$$

The answer to the problem is 0.88 for $D_M(s,a)_1$ and $D_M(s,a)_2$ respectively equal to 0.83 and 0. Since the active constraint is:

$$1.32\ D_M(s,a)_1 + 4.09\ D_M(s,a)_2 \geq 1.1$$

the other representative policies in the neighborhood of our representative policy (0.94, 0.06) is (0.76, 0.24). Since the value of the state-action distribution $D_M(s,a)_2$ is equal to zero, there is one polytope side in the neighborhood of our representative policy (0.94, 0.06). In order to assess local covering guarantee of our representative policy $(0.94, 0.06)$ we have to solve the following

$$1.05\ d^2_{\mu_1^2}(s,a)_1 + 18\ d^2_{\mu_1^2}(s,a)_2 = \sigma_1^2$$

$$1.32\ d^2_{\mu_1^2}(s,a)_1 + 4.09\ d^2_{\mu_1^2}(s,a)_2 = \sigma_1^2$$

$$d_{\mu_1^2}(s,a)_1 + d_{\mu_1^2}(s,a)_2 = 1$$

$$\Rightarrow \quad \sigma_1^2 = 1.08,\ d_{\mu_1^2}(s,a)_1 = 0.88,\ d_{\mu_1^2}(s,a)_2 = 0.12$$

and based on the Section 4.3.1 one system of type B as follows:

$$1.05\ d^2_{\mu_2^2}(s,a)_1 + 18\ d^2_{\mu_2^2}(s,a)_2 = \sigma_2^2$$

$$d_{\mu_2^2}(s,a)_2 = 0$$

$$d_{\mu_2^2}(s,a)_1 + d_{\mu_2^2}(s,a)_2 = 1$$

$$\Rightarrow \quad \sigma_2^2 = 1.05,\ d_{\mu_2^2}(s,a)_1 = 1,\ d_{\mu_2^2}(s,a)_2 = 0$$

In order to find the neighbors of the representative policy (0.06, 0.94) we have to solve the following optimization problem:

$$\min \quad 18\ D_M(s,a)_1 + 1.05\ D_M(s,a)_2$$

$$s.t.$$

$$2.57\ D_M(s,a)_1 + 1.63\ D_M(s,a)_2 \geq 1.1$$

$$1.05\ D_M(s,a)_1 + 18\ D_M(s,a)_2 \geq 1.1$$

$$1.32\ D_M(s,a)_1 + 4.09\ D_M(s,a)_2 \geq 1.1$$

$$0 \leq D_M(s,a)_1 \leq 1$$

$$0 \leq D_M(s,a)_2 \leq 1$$

The answer to the problem is 0.71 for $D_M(s,a)_1$ and $D_M(s,a)_2$ respectively equal to 0 and 0.67. Since the active constraint is:

$$2.57\ D_M(s,a)_1 + 1.63\ D_M(s,a)_2 \geq 1.1$$

so the other representative policies in the neighborhood of our representative policy (0.06, 0.94) is (0.39, 0.61). Since the value of the state-action distribution $D_M(s,a)_1$ is equal to zero so there is one polytope side in the neighborhood of our representative policy (0.06, 0.94). In order to assess local covering guarantee of our representative policy (0.06, 0.94) we have to solve one system of type A as follows:

$$18\ d^2_{\mu_1^3}(s,a)_1 + 1.05\ d^2_{\mu_1^3}(s,a)_2 = \sigma_1^3$$

$$2.57\ d^2_{\mu_1^3}(s,a)_1 + 1.63\ d^2_{\mu_1^3}(s,a)_2 = \sigma_1^3$$

$$d_{\mu_1^3}(s,a)_1 + d_{\mu_1^3}(s,a)_2 = 1$$

$$\Rightarrow \quad \sigma_1^3 = 1.21,\ d_{\mu_1^3}(s,a)_1 = 0.16,\ d_{\mu_1^3}(s,a)_2 = 0.84$$

and one system of type B as follows:

$$18\ d^2_{\mu_2^3}(s,a)_1 + 1.05\ d^2_{\mu_2^3}(s,a)_2 = \sigma_2^3$$

$$d_{\mu_2^3}(s,a)_1 = 0$$

$$d_{\mu_2^3}(s,a)_1 + d_{\mu_2^3}(s,a)_2 = 1$$

$$\Rightarrow \quad \sigma_2^3 = 1.05,\ d_{\mu_2^3}(s,a)_1 = 1,\ d_{\mu_2^3}(s,a)_2 = 0$$

In order to find the neighbors of the representative policy (0.76, 0.24) we have to solve the following optimization problem:

$$\min \quad 1.32\ D_M(s,a)_1 + 4.09\ \ D_M(s,a)_2$$

$$s.t.$$

$$2.57\ D_M(s,a)_1 + 1.63\ D_M(s,a)_2 \geq 1.1$$

$$1.05\ D_M(s,a)_1 + 18\ D_M(s,a)_2 \geq 1.1$$

$$18\ D_M(s,a)_1 + 1.05\ D_M(s,a)_2 \geq 1.1$$

$$0 \leq D_M(s,a)_1 \leq 1$$

$$0 \leq D_M(s,a)_2 \leq 1$$

The answer to the problem is 0.68 for $D_M(s,a)_1$ and $D_M(s,a)_2$ respectively equal to 0.4 and 0.03 Since the active constraints are:

$$2.57\ D_M(s,a)_1 + 1.63\ D_M(s,a)_2 \geq 1.1$$

$$1.05\ D_M(s,a)_1 + 18\ D_M(s,a)_2 \geq 1.1$$

the other representative policies in the neighborhood of our representative policy (0.76, 0.24) are (0.39, 0.61) and (0.94, 0.06). Since there is no state-action distribution equal to zero, so there is no polytope side in the neighborhood of our representative policy (0.76, 0.24). In order to assess local covering guarantee of our representative policy (0.76, 0.24) we have to solve two systems of type A as follows:

$$
\begin{aligned}
&1.32\ d^2_{\mu^4_1}(s,a)_1 + 4.09\ d^2_{\mu^4_1}(s,a)_2 = \sigma^4_1 \\
&2.57\ d^2_{\mu^4_1}(s,a)_1 + 1.63\ d^2_{\mu^4_1}(s,a)_2 = \sigma^4_1 \\
&d_{\mu^4_1}(s,a)_1 + d_{\mu^4_1}(s,a)_2 = 1
\end{aligned}
$$

$$\Rightarrow \quad \sigma^4_1 = 1.16,\ d_{\mu^4_1}(s,a)_1 = 0.58,\ d_{\mu^4_1}(s,a)_2 = 0.42$$

and:

$$
\begin{aligned}
&1.32\ d^2_{\mu^4_2}(s,a)_1 + 4.09\ d^2_{\mu^4_2}(s,a)_2 = \sigma^4_2 \\
&1.05\ d^2_{\mu^4_2}(s,a)_1 + 18\ d^2_{\mu^4_2}(s,a)_2 = \sigma^4_2 \\
&d_{\mu^4_2}(s,a)_1 + d_{\mu^4_2}(s,a)_2 = 1
\end{aligned}
$$

$$\Rightarrow \quad \sigma^4_2 = 1.08,\ d_{\mu^4_2}(s,a)_1 = 0.88,\ d_{\mu^4_2}(s,a)_2 = 0.12$$

obviously the maximum value of σ's is 1.21 for the policy (0.16, 0.84) so:

$$\max_{\mu} \min_{k} \int \frac{d^2_\mu(s,a)}{d_{\theta_k}(s,a)}\, \mathrm{d}s\ \mathrm{d}a = \mathbf{1.21}$$

and the remaining policy left uncovered and at the farthest distance from all the representative policies is: **(0.16, 0.84)**

A.2. Number of Covering Policies

Here we provide some numerical examples on the lower and upper bounds on the number of covering policies.

- $|SA| = 100, \sigma = 30$

To compute the lower bound of the needed number of representative policies, first we have to find λ value in a way that σ lies within the interval of:

$$\left(\frac{|SA|}{\lambda}, \frac{|SA|}{\lambda - 1}\right]$$

As we discussed earlier λ can take values from 2 to $|SA|$.

Clearly, with $\lambda = 4$, σ lies within the interval $\left(\frac{100}{4}, \frac{100}{3}\right]$. Since $|SA| - \lambda \geq 5$:

$$F(\lambda) = 1 + \frac{1}{32}\left(\frac{1}{2}\right)^{5\lambda} + \frac{31}{64}\left(\frac{1}{2}\right)^{4\lambda} + \frac{155}{64}\left(\frac{1}{2}\right)^{3\lambda} + \frac{155}{32}\left(\frac{1}{2}\right)^{2\lambda} + \frac{31}{8}\left(\frac{1}{2}\right)^{\lambda}$$

so with $\lambda = 4$, $F(\lambda)$ will be equal to 1.26.

Based on the found λ and $F(\lambda)$ values and the formulation derived in previous sections, the lower bound and the upper bound for the needed number of representative policies will be as follows:

$$N > F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e|SA|^2}{2\pi(\sigma-1)\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{79}$$

$$N < 2F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e\sigma|SA|(|SA|-1)}{2\pi\left(\sigma-1\right)^2\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{162}$$

By Increasing $\boldsymbol{\sigma}$, the Needed Number Will Decrease

- $|SA| = 100, \sigma = 40$

Then our λ value will be equal to 3, as with this value our σ lies within the interval $\left(\frac{100}{3}, \frac{100}{2}\right]$. Since $|SA| - \lambda \geq 5$ so $F(\lambda) = 1.56$, and we have

$$N > F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e|SA|^2}{2\pi(\sigma-1)\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{36}$$

$$N < 2F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e\sigma|SA|(|SA|-1)}{2\pi\left(\sigma-1\right)^2\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{72}$$

By Decreasing $\boldsymbol{\sigma}$, the Needed Number Will Increase:

- $|SA| = 100, \sigma = 20$

Then our λ value will be equal to 6, as with this value our σ lies within the interval $\left(\frac{100}{6}, \frac{100}{5}\right]$. Since $|SA| - \lambda \geq 5$ so $F(\lambda) = 1.06$ and we have:

$$N > F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e|SA|^2}{2\pi(\sigma-1)\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{292}$$

$$N < 2F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e\sigma|SA|(|SA|-1)}{2\pi\big(\sigma-1\big)^2\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{647}$$

Based on the Property 4.2.1 in a Polytope of $|SA|$ Dimensions Where $|SA|$ Is Greater Than or Equal to 4, Then for σ Values Greater Than or Equal to $\frac{|SA|}{2}$, the Maximum Needed Number of Representative Policies Will Be Equal to $|SA|+1$:

- if $|SA| = 6$ and $\sigma = 4$ then our λ value will be equal to 2 because with this value our σ lies within the interval $\left(\frac{6}{2}, \frac{6}{1}\right]$ and since $|SA| - \lambda = 4$ so $F(\lambda) = 2.24$ and we have:

$$N > F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e|SA|^2}{2\pi(\sigma-1)\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{4}$$

$$N < 2F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e\sigma|SA|(|SA|-1)}{2\pi\big(\sigma-1\big)^2\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{8}$$

- if $|SA| = 80$ and $\sigma = 45$ then our λ value will be equal to 2 because with this value our σ lies within the interval $\left(\frac{80}{2}, \frac{80}{1}\right]$ and since $|SA| - \lambda \geq 5$ so $F(\lambda) = 2.3$ and we have:

$$N > F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e|SA|^2}{2\pi(\sigma-1)\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{13}$$

$$N < 2F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e\sigma|SA|(|SA|-1)}{2\pi\big(\sigma-1\big)^2\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{26}$$

- if $|SA| = 1000$ and $\sigma = 505$ then our λ value will be equal to 2 because with this value our σ lies within the interval $\left(\frac{1000}{2}, \frac{1000}{1}\right]$ and since $|SA| - \lambda \geq 5$ so $F(\lambda) = 2.3$ and we have:

$$N > F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e|SA|^2}{2\pi(\sigma-1)\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{48}$$

$$N < 2F(\lambda)\sqrt{\frac{\lambda}{2}}\left(\frac{e\sigma|SA|(|SA|-1)}{2\pi\big(\sigma-1\big)^2\lambda(\lambda-1)}\right)^{\frac{\lambda-1}{2}} = \mathbf{96}$$

For a Polytope of Two Dimensions, Since in Spite of Having the Lower and the Upper Bound Formulations, We Also Have the Formulation for the Exact

Number of the Needed Representative Policies, So We Can See the Results in Practice with Some Numerical Example:

- if $|SA| = 2$ and $\sigma = 1.3$ then since σ lies within the interval of $\left(1, \frac{|SA|}{|SA|-1}\right]$ so without calculating λ value we can apply the first formulations to determine the lower and upper bounds (although if we compute λ it will be equal to $|SA|$ and consequently both the intervals and formulations are the same):

$$N > \sqrt{\frac{|SA|}{2}}\left(\frac{e}{2\pi(\sigma-1)}\right)^{\frac{|SA|-1}{2}} = \mathbf{2}$$

$$N < 2\sqrt{\frac{|SA|}{2}}\left(\frac{e\sigma(|SA|-1)}{2\pi(\sigma-1)^2|SA|}\right)^{\frac{|SA|-1}{2}} = \mathbf{4}$$

and on the other hand the exact number of the needed representative policies can be computed as follows:

$$N = 2\left\lceil \log_{\left(\frac{2-\sigma}{\sigma}\right)}\left(\frac{1-\sqrt{\sigma-1}}{\sigma}\right)\right\rceil - 1 = \mathbf{3}$$

as it was expected the exact number lies within the interval of our estimated lower and upper bound.

- if $|SA| = 2$ and $\sigma = 1.1$ then σ lies within the interval of $\left(1, \frac{|SA|}{|SA|-1}\right]$ and we have:

$$N > \sqrt{\frac{|SA|}{2}}\left(\frac{e}{2\pi(\sigma-1)}\right)^{\frac{|SA|-1}{2}} = \mathbf{3}$$

$$N < 2\sqrt{\frac{|SA|}{2}}\left(\frac{e\sigma(|SA|-1)}{2\pi(\sigma-1)^2|SA|}\right)^{\frac{|SA|-1}{2}} = \mathbf{10}$$

and on the other hand the exact number of the needed representative policies can be computed as follows:

$$N = 2\left\lceil \log_{\left(\frac{2-\sigma}{\sigma}\right)}\left(\frac{1-\sqrt{\sigma-1}}{\sigma}\right)\right\rceil - 1 = \mathbf{5}$$

as it was expected the exact number lies within the interval of our estimated lower and upper bound.

- if $|SA| = 2$ and $\sigma = 1.05$ then σ lies within the interval of $\left(1, \frac{|SA|}{|SA|-1}\right]$ and we have:

$$N > \sqrt{\frac{|SA|}{2}} \left(\frac{e}{2\pi(\sigma - 1)} \right)^{\frac{|SA|-1}{2}} = \mathbf{3}$$

$$N < 2\sqrt{\frac{|SA|}{2}} \left(\frac{e\sigma(|SA| - 1)}{2\pi\left(\sigma - 1\right)^2|SA|} \right)^{\frac{|SA|-1}{2}} = \mathbf{20}$$

and on the other hand the exact number of the needed representative policies can be computed as follows:

$$N = 2\left\lceil \log_{\left(\frac{2-\sigma}{\sigma}\right)} \left(\frac{1 - \sqrt{\sigma - 1}}{\sigma} \right) \right\rceil - 1 = \mathbf{7}$$

as it was expected the exact number lies within the interval of our estimated lower and upper bound.