Proper Laplacian Representation Learning

by

Diego Fernando Gomez Noriega

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

Master of Science

Department of Computing Science

University of Alberta

Abstract

The ability to learn good representations of states is essential for solving large reinforcement learning problems, where exploration, generalization, and transfer are particularly challenging. The *Laplacian representation* is a promising approach to address these problems by inducing intrinsic rewards for temporally-extended action discovery and reward shaping that can be used, among other things, to generate informative state encoding. To obtain the Laplacian representation one needs to compute the eigensystem of the graph Laplacian, which is often approximated through optimization objectives compatible with deep learning approaches. These approximations, however, depend on hyperparameters that are impossible to tune efficiently, converge to arbitrary rotations of the desired eigenvectors, and are unable to accurately recover the corresponding eigenvalues. In this dissertation we introduce a theoretically sound objective and corresponding optimization algorithm for approximating the Laplacian representation. Our approach naturally recovers both the true eigenvectors and eigenvalues while eliminating the hyperparameter dependence of previous approximations. We provide theoretical guarantees for our method and we show that those results translate empirically into robust learning across multiple environments.

Preface

Most of the content of this thesis is based on a paper submitted to ICLR (Gomez et al., 2023a) and co-authored with my supervisors, Marlos C. Machado and Michael Bowling.

Bound to this flesh.
This guise, this mask.
This dream.
Wake up remember.
We are born of one breath, one word.
We are all one spark, sun becoming.

– *Pneuma*, Tool.

In the beginning was the Word, and the Word was with God, and the Word was God. He was with God in the beginning. Through him all things were made; without him nothing was made that has been made. In him was life, and that life was the light of all mankind.

– John 1.

Acknowledgements

First and foremost, I want to thank Marlos. Without him, this project would have not seen the light of day, and specially not so soon. Not only he introduced me to the concept of the Laplacian in RL, but he was wise enough to push me in the right direction and to not let me concede ahead of time. Second, I want to thank Mike, whose skeptic gaze always enriches my perspective upon my own work. I am grateful to both for always having a constructive and practical feedback that did not sacrifice any of the scientific spirit. Lastly, I am thankful to Rich and Csaba. With only some brief interactions during classes or climbing, I learned a lot from them and, most importantly, they influenced my understanding of artificial intelligence and the role of RL in it.

In addition, I want to thank my previous supervisors, Nicanor and Alonso. From Nicanor comes my love for control theory and dynamical systems, upon which my intuition about machine learning and reinforcement learning stems from. From Alonso comes my grasp of probability and information theory, and, consequently, of representation learning. In addition, working with them certainly impacted my perspective on research and science and the present dissertation is a clear result from these interactions.

Finally, I thank all the people that I have come to call friends in Edmonton, including those working and sharing time with me at RLAI and Amii. They certainly have made my stay here both fun and productive, full of exploration and learning. Particularly, I am chronologically grateful with Adeel, Amir S., Kushagra, Aakash, Freddy, Tales, Yazeed, Amir B., Kevin, Subho, Javier, Erfan, Shivam, Anna, Parham, Rohini, Jordan, Vlad, Deep, Alireza K., Jacob, Fátima, Ken, Alex A., Karan, Abdul, Esraa, Haseeb, Shibhansh, Prabhat, Manan, and Katyani.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Reinforcement learning is a framework for decision-making where an agent continually takes actions in its environment and, in doing so, controls its future states. After each action, given the current state and the action itself, the agent receives a reward and a next state from the environment. The objective of the agent is to maximize the sum of these rewards. In principle, the agent has to visit all states and try all possible actions a reasonable number of times to determine the optimal behavior. However, in complex environments, e.g., when the number of states is large or the environment changes with time, this is not a plausible strategy. Instead, the agent needs the ability to learn state and action abstractions that facilitate continual exploration (i.e., having the ability to visit all states at will) and generalization (i.e., being able to transfer the knowledge from one state to unexplored states).

The *Laplacian framework* (Mahadevan, 2005; Mahadevan & Maggioni, 2007) proposes one such abstraction. This abstraction is based on the graph Laplacian, which is a linear map that encodes the topology of the state space based on both the policy the agent uses to select actions and the environment dynamics.

Specifically, the $d-$dimensional *Laplacian representation* is a map from states to vectors—thus an abstraction—whose entries correspond to d eigenvectors of the Laplacian.

Among other properties, the Laplacian representation induces a metric space where the Euclidean distance of two representations correlates to the temporal distance of their corresponding states—meaning the number of time steps necessary to go from one state to another—and its entries correspond to directions that maximally preserve state value information. Correspondingly, it has been used for reward shaping (Wu et al., 2019; Wang et al., 2023), as a state representation for value approximation (e.g., Mahadevan & Maggioni, 2007; Lan et al., 2022; Wang et al., 2022), as a set of intrinsic rewards for exploration via temporally-extended actions (see overview by Machado et al., 2023), and to achieve state-of-the-art performance in sparse reward environments (Klissarov & Machado, 2023).

When the number of states, $|\mathcal{S}|$, is small, the graph Laplacian can be represented as a matrix and one can use standard matrix eigendecomposition techniques to obtain its *eigensystem*[1] and the corresponding Laplacian representation. In practice, however, $|\mathcal{S}|$ is large, or even uncountable. Thus, at some point it becomes infeasible to directly compute the eigenvectors of the Laplacian. In this context, Wu et al. (2019) proposed a scalable optimization procedure to obtain the Laplacian representation in state spaces with uncountably many states. Such an approach is based on a general definition of the graph Laplacian as a linear operator, also introduced by Wu et al. (2019). Importantly, this definition allows us to model the Laplacian representation as a neural network and to learn it by minimizing an unconstrained optimization objective, the *graph drawing objective* (GDO).

[1]We use the term eigensystem to refer to both the eigenvectors and eigenvalues of a linear operator.

A problem with this objective, however, is that arbitrary rotations of the eigenvectors of the Laplacian also minimize it (Wang et al., 2021). This not only implies that the solution found could differ from the true eigenvectors, but also that the gradient dynamics could be unstable. As a solution, Wang et al. (2021) proposed the *generalized graph drawing objective* (GGDO), which breaks the symmetry of the optimization problem by introducing a sequence of decreasing hyperparameters to GDO. While the true eigenvectors are the only solution to this new objective, the rotations of the smallest eigenvectors[2] are still equilibrium points of the generalized objective and their stability is not guaranteed when minimizing this objective with stochastic gradient descent. Consequently, there is variability in the solutions one actually finds when minimizing such an objective, depending, for example, on the initialization of the network and on the hyperparameters chosen.

Considering the potential of the Laplacian representation to address some of the main challenges in reinforcement learning, namely the discovery of state and action abstractions for exploration and generalization, this work seeks to solve the main shortcomings of the recent techniques to learn the Laplacian representation with neural networks. Correspondingly, we propose an alternative optimization objective, the Augmented Lagrangian Laplacian Objective (ALLO), for approximating the Laplacian representation. We prove that the unique stable equilibrium point under gradient ascent-descent dynamics of ALLO corresponds to both the true eigenvectors and eigenvalues of the Laplacian, independently of the original hyperparameters of GGDO. Besides theoretical guarantees, we empirically demonstrate that our proposed approach is robust across different environments with different topologies and state representations, and that it is

[2]We refer to the eigenvectors with corresponding smallest eigenvalues as the "smallest eigenvectors". The same logic applies to the "largest eigenvectors".

able to accurately recover the eigenvalues of the graph Laplacian as well.

The rest of the thesis is organized as follows. In Chapter 2, we introduce the necessary background to understand and motivate both the Laplacian representation and ALLO. In particular, we introduce the reinforcement learning framework, the exploration and generalization problems in this framework, and how they motivate the need for action and state abstractions. We also introduce the definitions of GDO and GGDO, their extension to abstract measure spaces, and how this extension leads to the actual deep learning loss function being optimized. In Chapter 3, we motivate each of the components that define ALLO, namely the stop-gradients, the dual variables, and the barrier coefficient. Then, in Chapter 4 we explain why ALLO is the proper objective for learning the Laplacian representation. Specifically, we prove that the Laplacian representation is the unique stable equilibrium for ALLO, when there is no function approximation. In Chapter 5, we present different experiments that demonstrate the potential of ALLO for learning the Laplacian representation when using deep learning models and optimizers. Finally, in Chapter 6 we summarize the contributions of this work and possible future work building on top of it.

Chapter 2

Background

We first review the reinforcement learning setting, before presenting the Laplacian representation. Then, we introduce optimization objectives proposed in the literature for approximating this representation. After this, we generalize the Laplacian and the objectives to the setting of abstract measure spaces. This will allow us to introduce the deep learning approximation considered in the experiments where we learn the Laplacian representation.

2.1 Reinforcement learning

Here we introduce basic concepts in reinforcement learning along with the main challenges in the field. These will allow us to define and motivate the Laplacian representation.

2.1.1 Agent-environment system

We consider the setting in which an agent interacts with an environment. The environment is a *reward-agnostic Markov-decision process* $M = (\mathcal{S}, \mathcal{A}, P)$ with state space $\mathcal{S}$, finite action space $\mathcal{A} = \{1, \cdots, |\mathcal{A}|\}$, and transition probability map $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, which maps a state-action pair, (s, a), to a state distribution, $P(\cdot|s, a)$, in the simplex, $\Delta(\mathcal{S})$. The agent is characterized by the policy, $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, that it uses to choose actions. At time-step $t = 0$, an initial state S_0 is sampled from some given initial state distribution $\mu_0 \in \Delta(\mathcal{S})$. Then, the agent samples an action A_0 from its policy and, as a response, the environment transitions to a new state S_1, following the distribution $P(S_0, A_0)$. After this, the agent selects a new action, the environment transitions again according to P, and so on. The agent-environment interaction determines a Markov process, a stochastic dynamical system, which, when $\mathcal{S}$ is finite, is characterized by the *transition matrix* $\mathbf{P}_\pi$, where $(\mathbf{P}_\pi)_{s,s'} = \sum_{a \in \mathcal{A}} \pi(a|s) P(s'|s, a)$ is the probability of transitioning from state s to state s' while following policy π.

2.1.2 Reinforcement learning tasks

Typically, in reinforcement learning, when the agent executes an action and the environment transitions to a new state, the agent also receives a reward signal determined by a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Then, the objective of the agent becomes to find the policy that maximizes the expected future rewards. Correspondingly, we define a *reinforcement learning task* $T = (r, \gamma)$ as a tuple consisting of a reward function r and a discount factor $\gamma \in [0, 1)$, where the factor γ is used to weigh the contribution of future rewards (for a detailed discussion

on reinforcement learning tasks, see White, 2017).[1]

2.1.3 The trajectory space and value functions

To adequately define the objective of the agent once a reinforcement learning task is posed, let us first introduce the trajectory space. A trajectory τ is an infinite sequence of the form $(s_0, a_0, s_1, a_1, \cdots)$. Along with a dynamical system, the agent-environment interconnection together with the initial state distribution, μ_0, determine a unique probability measure $\mathbb{P}^{\pi}_{M,\mu_0}$ over the trajectory space $\Omega = (\mathcal{S} \times \mathcal{A})^{\mathbb{N}}$. Hence, one can reason about random variables and their expected values in the *trajectory measure space*[2] $(\Omega, \mathcal{P}(\Omega), \mathbb{P}^{\pi}_{M,\mu_0})$ (for an in-depth discussion, see Szepesvári, 2023).

In particular, given an MDP M and a task T, we can define the return G of a random trajectory $\tau = (S_0, A_0, S_1, A_1, \cdots)$ as the discounted sum of rewards $G(\tau) = \sum_{t=0}^{\infty} \gamma^t r(S_t, A_t)$. The *state value function* of a policy π is the expected value of the return given an initial state: $v_\pi(s) = \mathbb{E}^{\pi}_{M}[G|S_0 = s]$. Then, an optimal policy π^* is one such that v_{π^*} is maximal for any possible state, i.e., $v_{\pi^*}(s) \geq v_\pi(s)$, $\forall s \in \mathcal{S}, \pi$, and the objective of the agent is to find one such optimal policy.[3]

[1] More generally, both the reward function and the discount factor can be functions of the current state and action, and also the next state. Nonetheless, the simpler definition is enough for our purpose of motivating the Laplacian.

[2] As common, we use $\mathcal{P}(\Omega)$ to denote the power set of Ω

[3] Note that the expected value, $\mathbb{E}^{\pi}_{M}$, and thus an optimal policy π^*, is independent of μ_0, because we are setting the initial state as the fixed state s.

2.1.4 The exploration-exploitation dilemma

While we mentioned that the agent is characterized by its policy π, the agent is not static. After all, its objective is to find an optimal policy given a task T. Thus, the agent is better understood as an algorithm A that, at a given time-step $t \in \mathbb{N}$, chooses a policy π_t from the space of possible policies $\Delta(\mathcal{A})^{\mathcal{S}}$, based on the finite history $\tau_t = (S_0, A_0, R_1, S_1, A_1, R_2 \cdots, S_t)$, where $A_k \sim \pi_k$, for $0 \leq k < t$.

In principle, any agent A that always finds an optimal policy given an arbitrary pair (M, T) is a "good" agent. However, in practice we not only care about being able to solve reinforcement learning tasks, but about solving them efficiently. That is, we would like the agent to find π^* with as few interactions as possible. This meta-objective is usually framed as a minimization of either regret or sample complexity. In the case of regret, the main idea is to minimize the expected difference between the sum of rewards, up to some horizon $H \in \mathbb{N}$, obtained by an oracle that knows an optimal policy, and the sum of rewards actually obtained by the agent A during its exploration of different policies. Alternatively, in the case of sample complexity, usually analyzed under the Probably-Approximately-Correct (PAC) framework, the idea is to minimize the number of times that the agent behaves suboptimally, i.e., when the expected difference $v_{\pi^*} - v_{\pi_t}$ is larger than some specified threshold $\varepsilon > 0$ (see Szepesvári, 2010; Brunskill & Li, 2014, respectively, for explicit definitions).

Both efficiency meta-objectives pose a trade-off that the agent needs to resolve: the agent needs to explore its possible behaviours (i.e., policies) to estimate their corresponding values and, at the same time, it needs to choose behaviours that maximize rewards, under its current knowledge. This trade-off is usually referred to as the *exploration-exploitation dilemma* and it is at the core of what "solving" reinforcement learning means. In general, successful algorithms are

those that visit all state-action pairs a reasonable number of times, as fast as possible, to estimate with some acceptable confidence how good they are, and then commit to behave in such a way that only high-value trajectories are likely. Some theoretically successful examples are E^3 (Kearns & Singh, 2002) and R-MAX (see Brafman & Tennenholtz, 2002; Kakade, 2003).

Despite the existence of efficient algorithms, there are two considerations that make achieving the exploration-exploitation balance challenging in practice: in general, the problem of interests i) are non-stationary, meaning that either the MDP M or the task T can change with time, which makes necessary a perpetual exploration; and ii) they have a large number of states or actions, which means that it is infeasible to visit all of them.

2.1.5 Reinforcement learning as continual learning

Usually, reinforcement learning is narrowly understood as the set of optimization problems and corresponding solution methods posed by the possible MDP-task pairs (M, T) (Sutton, 1997). However, as hinted in the previous paragraph, reinforcement learning can be more broadly understood as the problem of choosing appropriate actions, online, given a unique input stream of observations, as opposed to states (Sutton, 2020). These observations are lossy versions of the states, which are generated via the agent-environment interaction, as described earlier, but where the environment is potentially a non-stationary system. The agent, then, is forced to continually model and revise its knowledge about the world[4] and the possible ways to behave on it (see Ring, 1995, for an early take on reinforcement learning agents as continual learners).

Generally speaking, the agent could live in a tragic world where efficient

[4]Here, world is to be understood as the non-stationary agent-environment system.

learning is not possible. If the rate of and manner in which the world changes is too fast or arbitrary, the agent cannot interact enough times with the environment to make reasonable guesses about the optimal actions to take. One example is the case of non-stationary multi-armed bandits, where there is a single state, i.e., $|\mathcal{S}| = 1$. Under an efficient agent A, if during $H \in \mathbb{N}$ interactions the number of times that the optimal arm (action) changes is $0 \leq C \leq H$, then the dynamic regret of A is order $\sqrt{HC}$ (Abbasi-Yadkori et al., 2022). Thus, if C is proportional to H, meaning that the optimal action changes almost all times, the regret becomes linear in H. In other words, the agent will always remain regretful.

Following this observation, a reasonable stance is to assume that, while the environment might be changing all the time, there exist some invariances that partially determine the possible dynamics and tasks that the agent can interact with. Then, an ideal agent is one that can efficiently discover the invariant structure in the world and is able to exploit it to learn efficiently under the non-stationarity conditions. Consequently, much of the recent literature has focused on finding appropriate ways to represent and exploit invariances (e.g., see Sodhani et al., 2022; Gomez et al., 2023b). Two central and similar approaches rely on learning action and state abstractions (Sutton, 1998; Sutton et al., 2023). These abstractions allow the agent to represent and reason about the world in different temporal scales of behaviour, in such a way that it is possible to separate, at each scale, what is constant or informative from what is not. In what follows, we will introduce these two types of abstractions and the associated problems they address. Later on, we will relate the Laplacian representation with both of them.

2.1.6 Options and the exploration problem

Before focusing on any particular type of abstraction, let us revisit the problem of exploration in reinforcement learning. As mentioned earlier, the agent-environment interaction induces a probability measure $\mathbb{P}^{\pi}_{M,\mu_0}$ in the trajectory space. The agent has the power, then, to influence $\mathbb{P}^{\pi}_{M,\mu_0}$ by choosing its policy, but it is limited by the connectivity imposed by the MDP probability map P. In this way, the agent can choose to traverse more some trajectories than others while abiding by the topology of M. From this viewpoint, *the exploration problem* lies in the agent being able to traverse all possible trajectories (or at least all representative trajectories), to then identify which ones it should remain visiting. This can be particularly difficult for MDPs with numerous states and actions since the more decisions the agent needs to consider the more unlikely some trajectories become and the more concentration of measure happens for small regions of the trajectory space. Moreover, under the continual learning framework discussed, the agent has to retain the ability to continually explore all trajectories, i.e., it must be able to repeatedly traverse arbitrary trajectories.

A particularly successful approach to deal with the exploration problem consists in the use of temporally-extended action abstractions referred to as options (e.g., see Machado et al., 2023; Klissarov & Machado, 2023). An option $O = (\mathcal{S}_O, \pi_O, \mathsf{T}_O)$ is, formally speaking, a tuple that contains a set of states $\mathcal{S}_O \subset \mathcal{S}$, a policy $\pi_O : \mathcal{S} \to \Delta(\mathcal{A})$, and a probability density map, the termination function, $\mathsf{T}_O : \mathcal{S} \to \Delta(\{0, 1\})$ (for a detailed introduction, see Sutton et al., 1999). Intuitively, an option represents a temporally-extended action π_O that can be executed from any state contained in $\mathcal{S}_O$. In the call-and-return formalism, the way in which the option is executed is that, at each state encountered S, a primitive action A is sampled according to π_O, and then with some probability

$\mathsf{T}(S)$ the agent terminates the execution of π_O, after which a new option is chosen to be executed. One can notice how options provide a second decision-making level —or, in general, arbitrarily many levels of decision— where the agent only needs to reason about which options to select, instead of which actions, and where the intermediate transitions can be potentially ignored. The reason why options are desirable for exploration in particular is that they force the agent to commit to a coherent behaviour for an extended period of time. Such mechanism allows to shift the region of trajectories where the concentration of measure happens. Hence, if the agent chooses an appropriate set of diverse options, it can explore the possible different trajectories to then effectively choose an optimal policy. This becomes particularly relevant in the continual learning setting when the MDP M is assumed to be fixed while the task T fluctuates, given that effective exploration is not directly affected by the task. Under these considerations, how to find appropriate sets of options for exploration becomes a fundamental research problem in reinforcement learning (for a thorough discussion, see Machado, 2019).

2.1.7 State abstractions and the generalization problem

Even under the assumption that exploration is not a problem, be it because there is some mechanism that allows the agent to experience a diverse set of transitions, or that we do not care about finding an optimal policy efficiently, or that the environment and task are stationary, if the state space is immeasurably larger than the memory and computational capacities of the agent, then there is no guarantee that the agent will be able to learn in any meaningful way. The reason is that essentially any visited state will be a new state, and then no amount of exploration would lead the agent to increase its confidence about how good is executing an action at a particular state. Assuming that it is possible to learn

in such a state space then entails assuming that obtaining information about a state s reveals information about any other possible state s', to some extent or another. In this way, the *generalization problem* lies in the agent being able to maximally infer information about arbitrary states given its current experience.

The generalization problem is usually phrased as a state abstraction learning problem where the goal is to learn a map $\phi : \mathcal{S} \to \bar{\mathcal{S}}$ that maps states to an abstract state space $\bar{\mathcal{S}}$. In earlier works, the idea was to cluster together states such that either their dynamics, rewards, values, or optimal actions (e.g., Walsh et al., 2006) were *the same*, in a single or a multi-task setting (for an early review, see Li et al., 2006). Later on, natural generalizations introduced abstractions ϕ such that the abstract space is endowed with some type of metric D. In this case, the idea is to map states that are *similar* in some sense, but not necessarily equivalent —be it because they result in similar transitions (e.g., Zhang et al., 2021), rewards, values (e.g., Abel et al., 2020), or optimal actions (e.g., Gomez et al., 2023b) —to points in the abstract space that are close under the metric D (for a review, see Abel et al., 2018).

The problem of learning an adequate state abstraction is intimately related with the problem of option discovery. Both of these abstractions lead to hierarchies of decisions levels, and an appropriate interaction of them could naturally lead to the desired exploration-exploitation balance in a continual learning setting. In particular, recent works have proposed iterative cycles of learning where attaining maximal features given a representation leads to discover options, while using those options lead to explore novel states that induce changes on the agent's representation (e.g., see Sutton et al., 2023; Machado et al., 2023). In the next section we will see how the Laplacian representation is a viable candidate to be used as a state abstraction in such an iterative cycle.

2.2 The Laplacian representation

Now we introduce the state abstraction that will become the focus of this document, the Laplacian representation, and we provide the main arguments supporting its use.

2.2.1 Graph theory and the Laplacian representation

In graph theory, the object of study is a node set $\mathcal{V}$ whose elements are pairwise connected by edges. The edge between a pair of nodes $v, v' \in \mathcal{V}$ is quantified by a non-negative real number $w_{v,v'}$, which is 0 only if there is no edge between the nodes. The adjacency matrix, $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, stores the information of all edges such that $(\mathbf{W})_{v,v'} = w_{v,v'}$. The degree of a node v is the sum of the adjacency weights between v and all other nodes in $\mathcal{V}$. The degree matrix $\mathbf{D} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the diagonal matrix containing these degrees. The Laplacian of a graph $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, and, just as the adjacency matrix, it fully encodes the information of the graph.

If we consider the state space of an MDP M as the set of nodes, $\mathcal{V} = \mathcal{S}$, and $\mathbf{W}$ as determined by $\mathbf{P}_\pi$, then we might expect the graph Laplacian to encode useful temporal information about M, meaning the number of time steps required to go from one state to another. In accordance with Wu et al. (2019), we broadly define the *Laplacian* in the tabular reinforcement learning setting as any matrix $\mathbf{L} = \mathbf{I} - f(\mathbf{P}_\pi)$, where $f : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|} \to \mathrm{Sym}_{|\mathcal{S}|}(\mathbb{R})$ is some function that maps $\mathbf{P}_\pi$ to a symmetric stochastic matrix.[5] For example, if $\mathbf{P}_\pi$ is symmetric, the Laplacian is typically defined as either $\mathbf{L} = \mathbf{I} - \mathbf{P}_\pi$ or $\mathbf{L} = \mathbf{I} - (1-\lambda)\mathbf{\Phi}_\pi^\lambda$, where $\mathbf{\Phi}_\pi^\lambda = (\mathbf{I} - \lambda\mathbf{P}_\pi)^{-1}$ is a matrix referred to as the successor representation matrix

[5] The Laplacian has $|S|$ different **real** eigenvectors and corresponding eigenvalues only if it is symmetric.

(Dayan, 1993; Machado et al., 2018). In the case where $\mathbf{P}_\pi$ is not symmetric, $\mathbf{L}$ is usually defined as $\mathbf{L} = \mathbf{I} - \frac{1}{2}(\mathbf{P}_\pi + \mathbf{P}_\pi^\top)$ to ensure it is symmetric (Wu et al., 2019).

The *Laplacian representation*, $\phi : \mathcal{S} \to \mathbb{R}^d$, maps a state s to d corresponding entries in a set of $0 < d \leq |\mathcal{S}|$ chosen eigenvectors of $\mathbf{L}$, i.e., $\phi(s) = [\mathbf{e}_1[s], \cdots, \mathbf{e}_d[s]]^\top$, where $\mathbf{e}_i$ is the $i-$th smallest eigenvector of $\mathbf{L}$ and $\mathbf{e}_i[s]$, its $s-$th entry (Mahadevan & Maggioni, 2007; Stachenfeld et al., 2014; Machado et al., 2017).

2.2.2 Value approximation with the Laplacian representation

To motivate the use of the Laplacian representation, we need to understand its mathematical properties, in particular, the properties of its eigensystem. For this, we will follow closely derivations in Mahadevan (2005) and Machado (2019). As an initial approach, let us consider the case in which the state space is tabular, i.e., $\mathcal{S} = \{1, \cdots, |\mathcal{S}|\}$. In this case, the value function can be written as a vector $\mathbf{v}_\pi$ in $\mathbb{R}^{|\mathcal{S}|}$ that satisfies the Bellman equation (Bellman, 1952):

$$\mathbf{v}_\pi = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_\pi \,, \tag{2.1}$$

where $\mathbf{r}_\pi[s] = \sum_{a \in \mathcal{A}} \pi(a|s) r(s,a)$ is the expected reward at state s following policy π. If we introduce the successor representation matrix[6] $\mathbf{\Phi}_\pi^\gamma$, then the value function can be expressed as a linear function of the mean reward vector:

$$\mathbf{v}_\pi = \mathbf{\Phi}_\pi^\gamma \mathbf{r}_\pi \,. \tag{2.2}$$

[6]This matrix is guaranteed to exist provided that $\mathbf{P}_\pi$ is a stochastic matrix, which implies it has eigenvalues lower than or equal to 1.

Assuming that $\mathbf{\Phi}_\pi^\gamma$ is symmetric, we can express it as the matrix product $\mathbf{E}\mathbf{\Lambda}\mathbf{E}^\top$, where the columns $\mathbf{e}_1, \cdots, \mathbf{e}_{|\mathcal{S}|}$ of $\mathbf{E}$ are the eigenvectors of $\mathbf{\Phi}_\pi^\gamma$, and the diagonal of $\mathbf{\Lambda}$ contains the corresponding eigenvalues $\lambda_{\text{SR},1} \geq \cdots \geq \lambda_{\text{SR},|\mathcal{S}|}$ of $\mathbf{\Phi}_\pi^\gamma$. Thus, we can rewrite our value function in terms of inner-products with the eigenvectors:

$$\mathbf{v}_\pi = \sum_{i=1}^{|\mathcal{S}|} \lambda_{\text{SR},i} \langle \mathbf{e}_i, \mathbf{r}_\pi \rangle \, \mathbf{e}_i \, .$$

This sum can be approximated by truncating it up to some number of components d:

$$\mathbf{v}_\pi \approx \tilde{\mathbf{v}}_\pi = \sum_{i=1}^{d} \lambda_{\text{SR},i} \langle \mathbf{e}_i, \mathbf{r}_\pi \rangle \, \mathbf{e}_i \, .$$

If one considers an adversarial setting where $\mathbf{r}_\pi$ is chosen from a unitary sphere to increase the ℓ_2-distance between $\mathbf{v}_\pi$ and $\tilde{\mathbf{v}}_\pi$, then the components that should remain after truncation are those corresponding to the largest d eigenvalues $\lambda_{\text{SR},1} \geq \lambda_{\text{SR},2} \geq \cdots \geq \lambda_{\text{SR},d}$. So, we can conclude that the largest eigenvectors of $\mathbf{\Phi}_\pi^\gamma$ form a basis for the subspace of value functions that lie closest to the possible value functions resulting from a reward $\mathbf{r}_\pi$ with maximum ℓ_2-norm of 1. In other words, the largest eigenvectors provide a natural basis for the possible value functions that can result from the agent-environment interaction.

It is not difficult to note that the eigenvectors of $\mathbf{P}_\pi$, $\mathbf{\Phi}_\pi^\gamma$, and $\mathbf{L}$ are all shared when $\mathbf{P}_\pi$, and thus $\mathbf{\Phi}_\pi^\gamma$, is symmetric. To see this, assume $\mathbf{e}_i$ is an eigenvector of $\mathbf{P}_\pi$ with corresponding eigenvalue $\lambda_{P,i}$. Then, $\mathbf{P}_\pi^t \mathbf{e}_i = \mathbf{P}_\pi^{t-1}(\lambda_{P,i}\mathbf{e}_i) = \cdots = \lambda_{P,i}^t \mathbf{e}_i$. Since $\mathbf{\Phi}_\pi^\gamma = \sum_{t=0}^\infty \gamma^t \mathbf{P}_\pi^t$,[7] we obtain that $\mathbf{e}_i$ is an eigenvector of $\mathbf{\Phi}_\pi^\gamma$ with corresponding eigenvalue $\lambda_{\text{SR},i} = \frac{1}{1-\gamma\lambda_{P,i}}$, considering the geometric series for $\gamma\lambda_{P,i}$. Similarly, if we define the Laplacian as $\mathbf{L} = \mathbf{I} - \mathbf{P}_\pi$ or $\mathbf{L} = \mathbf{I} - (1-\gamma)\mathbf{\Phi}_\pi^\gamma$, we obtain that $\mathbf{e}_i$ is an eigenvector with corresponding eigenvalue $\lambda_i = 1 - \lambda_{P,i}$

[7]The Neumann series $\sum_{t=0}^\infty \mathbf{M}^t$ always exists for a matrix $\mathbf{M}$ with eigenvalues of magnitude less than or equal to 1 and it is equal to $(\mathbf{I} - \mathbf{M})^{-1}$.

or $\lambda_i = 1 - (1-\gamma)\lambda_{\text{SR},i} = \frac{\gamma(1-\lambda_{P,i})}{1-\gamma\lambda_{P,i}}$, respectively. Hence, we can conclude that the eigenvectors of the Laplacian associated to the smallest[8] $d-$eigenvalues $\lambda_1 \leq \cdots \leq \lambda_d$ provide a natural basis for the possible value functions.

2.2.3 A basis of behaviour

In view of the above, one could argue that each coordinate of the Laplacian representation, i.e., each eigenvector $\mathbf{e}_i$, captures a particularly different aspect, or mode, of all the possible value functions. On top of this, Equation (2.2) tells us that the value function space is a rotated and scaled version of the reward space. Hence, the coordinates $\mathbf{e}_i$ capture particular reward modes as well. Lastly, considering the reward hypothesis, i.e., that any goal corresponds to the accumulative maximization of a reward function (see Bowling et al., 2023), we reach the natural conclusion that each Laplacian representation coordinate encodes a distinct mode of behaviour, and that, as a whole, the Laplacian representation covers the main modes of behaviour that an agent can display. That is, the Laplacian representation is a principled state abstraction candidate to solve the option discovery problem laid out in the previous section.

There are multiple technical details that make the previous line of argumentation imprecise and even untrue. However, in what follows we will provide additional arguments for why in a general sense it holds true.

A main caveat of the Laplacian representation is that, as defined here, the Laplacian $\mathbf{L}$ is policy dependent. That is, its eigenvectors are functions of the policy π chosen by the agent. The fact that the particular eigenvectors for a policy π capture distinct reward modes is not affected by this. However, the ordering of these modes do depend on the specific policy, meaning that the first d modes

[8]Notice that the largest eigenvalues of $\mathbf{\Phi}_\pi^\gamma$ correspond to the smallest eigenvalues of $\mathbf{L}$.

given a policy π do not necessarily correspond to "main" modes of behaviour in any meaningful way. Furthermore, while one eigenvector $\mathbf{e}_i$ may capture a **reward mode** different to the reward mode captured by another eigenvector $\mathbf{e}_j$, it is questionable whether the **behaviour modes** captured by them are different.

While in general it is true that, given a specific P, a policy π might lead to arbitrary reward modes, there are two properties of the modes that always hold and that make them relevant in a general sense. To identify these properties, let us consider how the rewards propagate following the topology of the state space $\mathcal{S}$. In Figure 2.1 we can see a sketch that depicts a tabular state space with 2D lattice topology, meaning that each state corresponds to a tile in a 2D lattice and therefore has four contiguous neighbours. The arrows pointing up represent a reward of 1 and the arrow pointing down a reward of -1. In this manner, if we imagine that the reward in the state s_j propagates to its neighbour s_i with the same probability as the transition probability $p_{ij} = (\mathbf{P}_\pi)_{s_i,s_j}$, then the expected reward entering state s_i is $\sum_{j\in\mathcal{N}(i)} p_{ij} r_j$, where $\mathcal{N}(i)$ corresponds to the set of the four neighbour states. If we assume that in this case the transition probabilities are uniform, the incoming reward is $\frac{1}{2}$. If in addition we multiply the entering reward by the reward of the state, we obtain a measure C_i of how correlated s_i is with its neighbours. Hence, we have $C_i = \sum_{j\in\mathcal{N}(i)} p_{ij} r_i r_j$, which in our example is $\frac{1}{2}$ and it would be $-\frac{1}{2}$ if the arrow was the opposite. Since $p_{ij} = 0$ for any $j \notin \mathcal{N}(i)$, we can express the mean correlation for s_i as a sum over all states, i.e., $C_i = \sum_{j=1}^{|\mathcal{S}|} p_{ij} r_i r_j$. So, in the end, if we add up all the

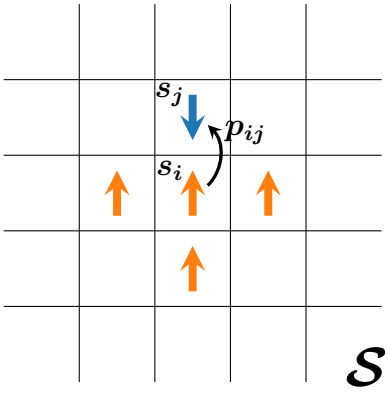


Figure 2.1: State space with 2D lattice topology

mean correlations, we obtain a single mean correlation measure C for the reward vector $\mathbf{r}$: $C(\mathbf{r}) = \sum_{i,j=1}^{|\mathcal{S}|} p_{ij} r_i r_j = \mathbf{r}^\top \mathbf{P}_\pi \mathbf{r}$. We can then see that the quadratic form $\mathbf{r}^\top \mathbf{L} \mathbf{r}$ is equal to the difference between the norm of $\mathbf{r}$, its auto-correlation, and the mean correlation with the neighbours: $\mathbf{r}^\top \mathbf{L} \mathbf{r} = \|\mathbf{r}\|_2^2 - C(\mathbf{r})$. The reason why this equality is interesting is the following. If we apply it to one of the eigenvectors $\mathbf{e}_k$, we can see that $\mathbf{e}_k^\top \mathbf{L} \mathbf{e}_k = \mathbf{e}_k^\top (\lambda_k \mathbf{e}_k) = \lambda_k = 1 - C(\mathbf{e}_k)$. Hence, the eigenvalue provides a negative measure of mean correlation given the dynamics. Moreover, the state space we introduced here is exactly the same as the 2D Ising lattice model in statistical physics where the rewards correspond to spins, and the expression $\sum_{i=1}^{|\mathcal{S}|} r_i r_i - \sum_{i,j=1}^{|\mathcal{S}|} p_{ij} r_i r_j$ is the energy of a spin configuration.[9] That is, **the eigenvalue $\boldsymbol{\lambda_k}$ is the energy of the reward mode $\mathbf{e_k}$**. Then, the lower the energy λ_k, the higher the correlation, and the more unlikely to observe local changes in the reward. We can conclude that if two eigenvectors $\mathbf{e}_i$ and $\mathbf{e}_j$ do not share their eigenvalue, then they will present a different level of variability and, in particular, a different number of local maxima. As a result, if we use them as intrinsic reward functions, the associated optimal policies, usually called eigenoptions (see the definition by Machado et al., 2017), present a different number of sink states. Consequently, the length of the trajectories spanned by the different policies vary and so the obtained behaviours correspond to different time-scales (to see experiments confirming this phenomena, refer to the survey by Machado et al., 2023). Moreover, since the eigenvectors are orthogonal, the local maxima for two eigenvectors has to occur in different states (otherwise their inner product would not cancel). In particular, for low energy eigenvectors, i.e., those corresponding to the Laplacian representation, the few existing local maxima will be typically distant between each other, owing to the

[9] To be precise, this is the energy of the configuration when there is an external magnetic field equal to the configuration and the couplings between the spins are given by the transition probabilities.

large correlation around the maxima.

In summary, irrespective of the policy π, the Laplacian representation induces **behaviour modes** that are maximally different in a meaningful way. Specifically, the behaviours correspond to the visitation of maximally distant regions, under the distance induced by the topology of $\mathcal{S}$, and they correspond to different temporal scales. In other words, **the Laplacian representation presents a general approach to discover options for exploration**.

2.3 Optimization approach to the Laplacian

Now that we have defined and motivated the use of the Laplacian representation, we are interested in techniques that reliably allow to approximate it in a general setting (i.e., where the state space can be arbitrarily large). For this, we introduce the optimization approach to the Laplacian. The main idea is to propose an optimization problem such that its solution corresponds to the Laplacian representation. Then, any iterative optimization technique, like stochastic gradient descent, becomes a plausible algorithm to obtain the Laplacian representation.

2.3.1 The Graph Drawing Objective

Given the graph Laplacian $\mathbf{L}$, the spectral graph drawing optimization problem (Koren, 2003) is defined as follows:[10]

$$\begin{aligned} \min_{\mathbf{u}_1,\cdots,\mathbf{u}_d \in \mathbb{R}^{|\mathcal{S}|}} \quad & \sum_{i=1}^{d} \langle \mathbf{u}_i, \mathbf{L}\mathbf{u}_i \rangle \\ \text{such that} \quad & \langle \mathbf{u}_j, \mathbf{u}_k \rangle = \delta_{jk} \ , \ 1 \leq k \leq j \leq d \, , \end{aligned} \tag{2.3}$$

[10]Note that, under the previously introduced interpretation of the eigenvalues, this problem corresponds to energy minimization.

where $\langle \cdot, \cdot \rangle$ is the inner product in $\mathbb{R}^{|\mathcal{S}|}$ and δ_{jk} is the Kronecker delta. This optimization problem has two desirable properties. The first one is that the d smallest eigenvectors of $\mathbf{L}$ are a global optimizer.[11] Hence, the Laplacian representation ϕ associated with $\mathbf{L}$ is a solution to this problem. The second property is that both objective and constraints can be expressed as expectations, making the problem amenable to stochastic gradient descent. In particular, the original *constrained* optimization problem (2.3) can be approximated by the unconstrained *graph drawing objective* (GDO):

$$\min_{\mathbf{u} \in \mathbb{R}^{d|\mathcal{S}|}} \quad \sum_{i=1}^{d} \langle \mathbf{u}_i, \mathbf{L}\mathbf{u}_i \rangle + b \sum_{j=1}^{d} \sum_{k=1}^{d} \left(\langle \mathbf{u}_j, \mathbf{u}_k \rangle - \delta_{jk} \right)^2 , \tag{2.4}$$

where $b \in (0, \infty)$ is a scalar hyperparameter and $\mathbf{u} = [\mathbf{u}_1^\top, \cdots, \mathbf{u}_d^\top]^\top$ is the vector that results from concatenating the vectors $(\mathbf{u}_i)_{i=1}^d$ (Wu et al., 2019).

2.3.2 The Generalized Graph Drawing Objective

As mentioned before, any rotation of the smallest eigenvectors of the Laplacian $\mathbf{L}$ is a global optimizer of the constrained optimization problem (2.3). Hence, even with an appropriate choice of hyperparameter b, GDO does not necessarily approximate the Laplacian representation ϕ. As a solution, Wang et al. (2021) present the generalized graph drawing optimization problem:

$$\begin{aligned} \min_{\mathbf{u} \in \mathbb{R}^{d|\mathcal{S}|}} \quad & \sum_{i=1}^{d} c_i \langle \mathbf{u}_i, \mathbf{L}\mathbf{u}_i \rangle \\ \text{such that} \quad & \langle \mathbf{u}_j, \mathbf{u}_k \rangle = \delta_{jk} \ , \ 1 \leq k \leq j \leq d \, , \end{aligned} \tag{2.5}$$

[11] This should be intuitive, given that these are the set of independent vectors with lowest energy. For proofs in the tabular setting, see the work by Koren (2003) for the case $d = 2$, and Lemma 1 for arbitrary d. For the abstract setting, see the work by Wang et al. (2021).

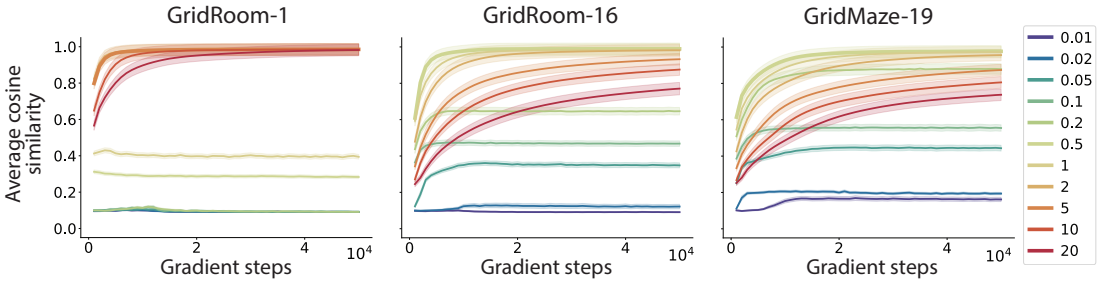


Figure 2.2: Average cosine similarity between the true Laplacian representation and GGDO. Each curve corresponds to a different value for a hyperparameter defining GGDO (named the barrier penalty coefficient), averaged over 60 seeds. The thickest curve correspond to the best coefficient and the shaded regions, to a 95% confidence interval.

where $c_1 > \cdots > c_d > 0$ is a monotonically decreasing sequence of d hyperparameters. Correspondingly, the unconstrained *generalized graph drawing objective* (GDDO) is defined as:

$$\min_{\mathbf{u}\in\mathbb{R}^{d|\mathcal{S}|}} \quad \sum_{i=1}^{d} c_i \langle \mathbf{u}_i, \mathbf{L}\mathbf{u}_i \rangle + b \sum_{j=1}^{d}\sum_{k=1}^{d} \min(c_j, c_k)\big(\langle \mathbf{u}_j, \mathbf{u}_k \rangle - \delta_{jk}\big)^2 . \tag{2.6}$$

Wang et al. (2021) prove that the optimization problem (2.5) has a unique global minimum that corresponds to the smallest eigenvectors of $\mathbf{L}$, for *any* possible choice of the hyperparameter sequence $(c_i)_{i=1}^d$. However, in the unconstrained setting (2.6), which is the setting used when training neural networks, these hyperparameters do affect both the dynamics and the quality of the final solution. In particular, Wang et al. (2021) found in their experiments that the linearly decreasing choice $c_i = d - i + 1$ performed best across different environments. More importantly, under gradient descent dynamics, the introduced coefficients are unable to break the symmetry among arbitrary rotations of the eigenvectors since they remain as equilibrium points (see Corollary (1) in Section 4).

These issues are particularly problematic because it is **impossible to tune**

the hyperparameters of GGDO without already having access to the problem solution: previous results, when sweeping hyperparameters, used the cosine similarity between the *true eigenvectors* and the approximated solution as a performance metric. To make matters worse, the best **hyperparameters are environment dependent**, as shown in Figure 2.2. Thus, when relying on GDO, or GGDO, *it is impossible to guarantee an accurate estimate of the eigenvectors of the Laplacian in environments where one does not know these eigenvectors in advance*, which obviously defeats the whole purpose.

2.4 Extension to abstract state spaces

The definition of the Laplacian representation and the associated graph drawing optimization problems (2.3) and (2.5) relied on the assumption that the state space was a finite set $\mathcal{S} = \{1, \cdots, |\mathcal{S}|\}$. In particular, the Laplacian $\mathbf{L}$ was defined as a matrix, and the notion of eigenvector depended on the standard Euclidean inner product. Since we are interested in the Laplacian representation being applicable to real-world problems, in general the size of the state space can be assumed, for all intents and purposes, infinite and, in particular, uncountable. This motivates the need to generalize the previous concepts and objectives to the case of abstract spaces. For this, we follow closely the work of Wu et al. (2019).

2.4.1 Reinforcement learning in abstract spaces

To work with a potentially uncountable state space $\mathcal{S}$, we have to revisit the definition of reward-agnostic Markov-decision process. In principle, the definition holds as is, but there are gaps to fill and details worth mentioning. First, the state distribution $P(s, a)$ corresponding to the state-action pair (s, a), now denoted

as $\mathbb{P}(\cdot|s,a)$, is not well-defined by itself. In the abstract setting, distributions are probability measures defined over a measurable set, which means that we need to choose an appropriate $\sigma-$algebra $\Sigma \subset \mathcal{P}(\mathcal{S})$ containing those sets of states that can be measured. In this way, the state space is now an abstract measurable space $(\mathcal{S}, \Sigma)$ and the simplex $\Delta(\mathcal{S})$ refers to the set of probability measures defined over $(\mathcal{S}, \Sigma)$. Moreover, the policy π remains the same, but the Markov process induced by it cannot be represented anymore by a matrix $\mathbf{P}_\pi$. In its place, we have now the transition probability map $\mathbb{P}_\pi : \mathcal{S} \to \Delta(\mathcal{S})$ defined by $\mathbb{P}_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\mathbb{P}(\cdot|s,a)$. Beyond this, the definitions of reward functions, tasks, trajectories, returns, and state value functions all remain the same. Hence, the objective given an MDP-task pair (M, T) is the same: solve the optimization problem $\arg\max_{\pi\in\Delta(\mathcal{A})^{\mathcal{S}}} v_\pi$.

2.4.2 Linear operators and the abstract Laplacian

As matrices, both $\mathbf{L}$ and $\mathbf{P}_\pi$ mapped vectors in $\mathbb{R}^{|\mathcal{S}|}$ back to the same finite-dimensional vector space. This was the property that, in particular, allowed to connect the eigensystem of $\mathbf{L}$ with the (sub-)basis in the value space that best preserved the value functions given a fixed $\mathbf{P}_\pi$. For this to make sense in the abstract setting, and preserve the same relationship, we need to determine what is the abstract vector space $\mathcal{V}$ where the value functions live, and then redefine the Laplacian as a general linear operator in the set of linear operators $\text{End}(\mathcal{V})$.

In principle, we know that any value function v is a function from $\mathcal{S}$ to $\mathbb{R}$, i.e., $v \in \mathbb{R}^{\mathcal{S}}$. In addition, since value functions are still defined as expected values, given some policy π, we recover the Bellman equation:

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_M^\pi\left[\sum_{t=0}^{\infty}\gamma^t r(S_t,A_t)\,\Big|\,S_0=s\right]\\
&= \sum_{a\in\mathcal{A}}\pi(a|s)\left(r(s,a)+\gamma\mathbb{E}_M^\pi\left[\sum_{t=1}^{\infty}\gamma^t r(S_t,A_t)\,\Big|\,S_0=s,A_0=a\right]\right)\\
&= r_\pi(s)+\gamma\sum_{a\in\mathcal{A}}\pi(a|s)\mathbb{E}_M^\pi\left[\mathbb{E}_M^\pi\left[\sum_{t=0}^{\infty}\gamma^t r(S'_t,A'_t)\,\Big|\,S'_0=S_1\right]\,\Big|\,S_0=s,A_0=a\right]\\
&= r_\pi(s)+\gamma\sum_{a\in\mathcal{A}}\pi(a|s)\mathbb{E}_M^\pi\left[v_\pi(S_1)\,\Big|\,S_0=s,A_0=a\right]\\
&= r_\pi(s)+\gamma\sum_{a\in\mathcal{A}}\pi(a|s)\int_{\mathcal{S}}v_\pi(s')\mathbb{P}(ds'|s,a)\\
&= r_\pi(s)+\gamma\int_{\mathcal{S}}v_\pi(s')\mathbb{P}_\pi(ds'|s)\,,
\end{aligned}
$$

where, similarly as before, $r_\pi(s)=\sum_{a\in\mathcal{A}}\pi(a|s)r(s,a)$, and $\int_{\mathcal{S}}$ is used to denote the Lebesgue integral in $\mathcal{S}$.

To obtain the same expression (2.1) as before, we need to define the "product" $P_\pi v_\pi$. We can achieve this by choosing a common probability measure ρ in $(\mathcal{S},\Sigma)$ and introducing the probability densities $p_\pi(\cdot|s):\mathcal{S}\to[0,\infty)$ for all $s\in\mathcal{S}$, defined as the Radon-Nikodym derivatives $p_\pi(s'|s)=\frac{d\mathbb{P}_\pi(\cdot|s)}{d\rho}\Big|_{s'}$. This is the same to say that the probability of reaching a state in the set $\mathcal{B}\subset\mathcal{S}$, starting from state s, is the integral of the density in $\mathcal{B}$:

$$
\mathbb{P}_\pi(\mathcal{B}|s)=\int_{\mathcal{B}}p_\pi(s'|s)\rho(ds')\,.
$$

In this manner, we can introduce the transition operator $P_\pi:\mathbb{R}^{\mathcal{S}}\to\mathbb{R}^{\mathcal{S}}$,

defined by:

$$[P_\pi v](s) = \int_{\mathcal{S}} v(s') p_\pi(s'|s) \rho(ds') \,,$$

and, as a result, we obtain our desired Bellman equation:

$$v_\pi = r_\pi + \gamma P_\pi v_\pi \,.$$

Note, however, that the Lebesgue integral defining P_π could be infinite for some arbitrary function $v \in \mathbb{R}^{\mathcal{S}}$. This means that we need to constrain our vector space to those functions v such that $\int_{\mathcal{S}} v(s') p_\pi(s'|s) \rho(ds') < \infty$, for all s. More generally, if we want to define the Laplacian as the operator $L = I - f(P_\pi)$, for some arbitrary function f, we require that $\int_{\mathcal{S}} v(s) p(s) \rho(ds) < \infty$ for an arbitrary density p. One can prove that the square of $\int_{\mathcal{S}} v(s) p(s) \rho(ds)$ is upper-bounded by the product $\int_{\mathcal{S}} v(s)^2 \rho(ds) \cdot \int_{\mathcal{S}} p(s)^2 \rho(ds)$ (see Section 2.11 in the book by Bogachev & Ruas, 2007). Hence, if we require both value functions and densities to be square integrable with respect to our chosen metric ρ, then the Laplacian will be well-defined. In this manner, we restrict our abstract MDPs to have probability maps P such that the resultant transition densities p_π are square $\rho-$integrable and the vector space of value functions considered is $\mathcal{V} = \mathcal{L}^2(\rho) = \{v \in \mathbb{R}^{\mathcal{S}} : \int_{\mathcal{S}} v(s)^2 \rho(ds) < \infty\}$.

As a final step, we define the Laplacian in the abstract reinforcement learning setting, or *abstract Laplacian* for simplicity, as the square $\rho-$integrable linear operator $L = I - f(P_\pi)$ in $\mathrm{End}(\mathcal{V})$, where I is the identity linear operator and $f : \mathrm{End}(\mathcal{V}) \to \mathrm{End}(\mathcal{V})$ is a function that maps P_π to a *self-adjoint* square $\rho-$integrable linear operator $f(P_\pi)$. That a linear operator is self-adjoint means, essentially, that its corresponding density with respect to ρ is symmetric. Thus,

this restriction is equivalent to ask that $\mathbf{L}$ is a symmetric matrix in the finite-dimensional case. Correspondingly, when $p_\pi(s'|s) \neq p_\pi(s|s')$, we can define f in such a way that the density of $f(P_\pi)$ is $\frac{1}{2}(p_\pi(s|s') + p_\pi(s'|s))$ (see Equation (4) in Wu et al., 2019).

2.4.3 Graph drawing objectives revisited

The concept of eigenvector remains the same in the abstract setting, meaning that an eigenvector $e_i \in \mathcal{V}$ of the Laplacian L (or any other linear operator) and its corresponding eigenvalue $\lambda_i \in \mathbb{R}$ is a vector such that $Le_i = \lambda_i e_i$. We are interested, however, in finding a countable set of such eigenvectors with the property of being all "orthogonal". This is where things change in the abstract setting, since the concept of orthogonality depends on the explicit choice of an inner-product $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \to [0, \infty)$. Now, having already selected a measure ρ in the state space, a natural choice is the induced inner-product $\langle v_1, v_2 \rangle_\rho = \int_{\mathcal{S}} v_1(s) v_2(s) \rho(ds)$, which is the expected correlation of the input vectors under the measure ρ.

Under the previous considerations, the graph drawing optimization problem can be extended to the abstract setting as:

$$\begin{aligned} \min_{u_1, \cdots, u_d \in \mathcal{V}} \quad & \sum_{i=1}^{d} \langle u_i, \mathbf{L} u_i \rangle_\rho \\ \text{such that} \quad & \langle u_j, u_k \rangle_\rho = \delta_{jk} \ , \ 1 \leq k \leq j \leq d \, . \end{aligned} \tag{2.7}$$

We can notice that the only change with respect to the original problem (2.3) is that solutions are now functions in $\mathcal{V}$, instead of finite-dimensional vectors in $\mathbb{R}^{|\mathcal{S}|}$, and the inner-product is now the one induced by ρ, i.e., $\langle \cdot, \cdot \rangle_\rho$. Clearly, the remaining optimization problems (2.4-2.6) can be extended in the same manner.

2.4.4 Monte-Carlo approximations to the drawing objectives

What remains is to show that the abstract version of the GDO is an expectation. As mentioned earlier, this is very convenient, as it implies that the GDO, or its gradient, can be estimated via Monte-Carlo sampling techniques.

Without loss of generality, let us suppose that P_π is self-adjoint and that the Laplacian is defined as $L = I - P_\pi$. Then, considering that p_π is symmetric, in the sense that $p_\pi(s'|s) = p_\pi(s|s')$, we have:

$$
\begin{aligned}
\langle u_i, Lu_i \rangle_\rho &= \langle u_i, (I - P_\pi)u_i \rangle_\rho \\
&= \langle u_i, u_i \rangle_\rho - \langle u_i, P_\pi u_i \rangle_\rho \\
&= \int_{\mathcal{S}} u_i(s)^2 \rho(ds) - \int_{\mathcal{S}} u_i(s)[P_\pi u_i](s)\rho(ds) \\
&= \int_{\mathcal{S}} u_i(s)^2 \rho(ds) \underbrace{\int_{\mathcal{S}} p_\pi(s'|s)\rho(ds')}_{=1} - \int_{\mathcal{S}} u_i(s) \underbrace{\int_{\mathcal{S}} p_\pi(s'|s)u_i(s')\rho(ds')}_{=[P_\pi u_i](s)} \rho(ds) \\
&= \frac{1}{2}\left(2\int_{\mathcal{S}}\int_{\mathcal{S}} u_i(s)^2 p_\pi(s'|s)\rho(ds)\rho(ds') - 2\int_{\mathcal{S}}\int_{\mathcal{S}} u_i(s)p_\pi(s'|s)u_i(s')\rho(ds')\rho(ds) \right) \\
&= \frac{1}{2}\left(\int_{\mathcal{S}}\int_{\mathcal{S}} u_i(s)^2 p_\pi(s'|s)\rho(ds)\rho(ds') + \int_{\mathcal{S}}\int_{\mathcal{S}} u_i(s')^2 p_\pi(s'|s)\rho(ds')\rho(ds) \right. \\
&\qquad \left. - 2\int_{\mathcal{S}}\int_{\mathcal{S}} u_i(s)p_\pi(s'|s)u_i(s')\rho(ds')\rho(ds) \right) \\
&= \frac{1}{2}\int_{\mathcal{S}}\int_{\mathcal{S}} \left(u_i(s) - u_i(s')\right)^2 p_\pi(s'|s)\rho(ds)\rho(ds') \\
&= \frac{1}{2}\mathbb{E}_{S\sim\rho}\left[\mathbb{E}_{S'\sim\mathbb{P}_\pi(\cdot|S)}\left[\left(u_i(S) - u_i(S')\right)^2 \right] \right].
\end{aligned}
$$

Hence, we can conclude that each contributing term in the loss function of the graph drawing optimization problems (2.3) and (2.5) can be approximated by

sampling pairs of consecutive states (S, S'), where state S is sampled from the measure ρ, and the next state S' from the transition measure given S, i.e., $\mathbb{P}_\pi(\cdot|S)$.

Finally, taking into account that the chosen inner-product $\langle \cdot, \cdot \rangle_\rho$ is an expected-value in itself, we can write the abstract version of the GGDO as:

$$\min_{u_1,\cdots,u_d \in \mathcal{V}} \quad \mathop{\mathbb{E}}_{\substack{S\sim\rho \\ S'\sim\mathbb{P}_\pi(\cdot|S)}} \left[\frac{1}{2} \sum_{i=1}^{d} c_i \left(u_i(S) - u_i(S') \right)^2 \right] + \cdots \tag{2.8}$$

$$\cdots + b \mathop{\mathbb{E}}_{\substack{S\sim\rho \\ Z\sim\rho}} \left[\sum_{j,k=1}^{d} \min(c_j, c_k) \left(u_j(S) u_k(S) - \delta_{jk} \right) \left(u_j(Z) u_k(Z) - \delta_{jk} \right) \right].$$

2.4.5 Deep learning approximation

So far, we introduced a general definition of the Laplacian that works in arbitrary measure spaces. However, there is no way to compute the solution of an optimization problem where the feasible set is an abstract space $\mathcal{V}$. Instead, we have to approximate the solutions and we do so by means of parametric function approximators. In particular, we replace the d functions $(u_1, \cdots, u_d)$ in the problem (2.8) by the parametric model $\phi_{\boldsymbol{\theta}} : \mathcal{S} \to \mathbb{R}^d$, where $\boldsymbol{\theta}$ is a finite dimensional parameter vector. In our case, $\phi_{\boldsymbol{\theta}}$ represents a neural network, and $\boldsymbol{\theta}$ a vector containing the weights of the network. This choice allows us to find an approximate solution by iteratively sampling a transition batch, calculating the corresponding optimization objective, and propagating the gradients by means of any stochastic gradient descent-based optimizer. In the following chapters we will see why this gradient-descent procedure is incompatible with GGDO and how our proposed objective, ALLO, overcomes this incompatibility in theory and in practice.

Chapter 3

Augmented Lagrangian Laplacian Objective

In this chapter we introduce a method that retains the benefits of GGDO while avoiding its pitfalls. Specifically, we relax the goal of having a unique global minimum for a constrained optimization problem like (2.5). Instead, we modify the stability properties of the unconstrained dynamics to ensure that the only stable equilibrium point corresponds to the Laplacian representation.

3.1 Asymmetric Constraints as a Generalized Graph Drawing Alternative

We want to break the *dynamical symmetry* of the Laplacian eigenvectors that make any of their rotations an equilibrium point for GDO (2.4) and GGDO (2.6) while avoiding the use of hyperparameters. For this, let us consider the original graph drawing optimization problem (2.3). If we set $d = 1$, meaning we try

to approximate only the first eigenvector $\mathbf{e}_1$, it is clear that the only possible solution is $\mathbf{u}_1^* = \mathbf{e}_1$. This happens because the only possible rotations are $\pm\mathbf{e}_1$. If we then try to solve the optimization problem for $d = 2$, but fix $\mathbf{u}_1 = \mathbf{e}_1$, the solution will be $(\mathbf{u}_1^*, \mathbf{u}_2^*) = (\mathbf{e}_1, \mathbf{e}_2)$, as desired. Repeating this process d times, we can obtain ϕ. Thus, we can eliminate the need for the d hyperparameters introduced by GGDO by solving d separate optimization problems. To replicate this separation while maintaining a single unconstrained optimization objective, we introduce the stop-gradient operator $[\![\cdot]\!]$ in GDO. This operator does not affect the objective in any way, but it indicates that, when following gradient descent dynamics, the real gradient of the objective is not used. Instead, when calculating derivatives, whatever is inside the operator is assumed to be constant. Specifically, the objective becomes:

$$\min_{\mathbf{u}\in\mathbb{R}^{d|\mathcal{S}|}} \quad \sum_{i=1}^{d} \langle \mathbf{u}_i, \mathbf{L}\mathbf{u}_i \rangle + b \sum_{j=1}^{d} \sum_{k=1}^{j} \left(\langle \mathbf{u}_j, [\![\mathbf{u}_k]\!] \rangle - \delta_{jk} \right)^2 . \tag{3.1}$$

Note that in addition to the stop-gradient operators, the upper bound in the inner summation is now the variable j, instead of the constant d. These two modifications ensure that $\mathbf{u}_i$ changes only to satisfy the constraints associated to the previous vectors $(\mathbf{u}_j)_{j=1}^{i-1}$ and itself, but not the following ones, i.e., $(\mathbf{u}_j)_{j=i+1}^{d}$. Hence, the asymmetry in the descent direction achieves the same effect as having d separate optimization problems. In particular, as proved in Lemma 2 in the next section, the descent direction of the final objective, yet to be defined, becomes $\mathbf{0}$ only for permutations of a subset of the Laplacian eigenvectors, and not for any of its rotations.

3.2 Augmented Lagrangian Dynamics for Exact Learning

The regularization term added in all of the previous objectives (2.4), (2.6), and (3.1) is typically referred to as a quadratic penalty with barrier coefficient b. This coefficient shifts the equilibrium point of the original optimization problems (2.3) and (2.5), and one can only guarantee that the desired solution is obtained in the limit $b \to \infty$ (see Chapter 17 by Nocedal & Wright, 2006). In practice, one can increase b until a satisfactory solution is found. However, not only is there no direct metric to tell how close one is to the true solution, but also an extremely large b is empirically bad for neural networks when optimizing GDO or GGDO. As a principled alternative, we propose the use of augmented Lagrangian methods. Specifically, we augment the objective (3.1) by adding the original constraints, multiplied by their corresponding dual variables, $(\beta_{jk})_{1\leq k\leq j\leq d}$. This turns the optimization problem into the following max-min objective, which we call the *augmented Lagrangian Laplacian objective* (ALLO):

$$\max_{\boldsymbol{\beta}} \min_{\mathbf{u}\in\mathbb{R}^{d|\mathcal{S}|}} \quad \sum_{i=1}^{d}\langle \mathbf{u}_i, \mathbf{L}\mathbf{u}_i\rangle + \sum_{j=1}^{d}\sum_{k=1}^{j}\beta_{jk}\big(\langle \mathbf{u}_j, [\![\mathbf{u}_k]\!]\rangle - \delta_{jk}\big) + \cdots \tag{3.2}$$

$$\cdots + b\sum_{j=1}^{d}\sum_{k=1}^{j}\big(\langle \mathbf{u}_j, [\![\mathbf{u}_k]\!]\rangle - \delta_{jk}\big)^2,$$

where $\boldsymbol{\beta} = [\beta_{1,1}, \beta_{2,1}, \beta_{2,2}, \cdots, \beta_{d,1}, \cdots, \beta_{d,d}] \in \mathbb{R}^{d(d+1)/2}$ is a vector containing all of the dual variables. There are two reasons to introduce the additional linear penalties, which at first glance do not seem to contribute anything that the quadratic one is not adding already. First, for an appropriately chosen b, the equilibria of the max-min objective (3.2) corresponds exactly to permutations of the smallest Laplacian eigenvectors, and only the sorted eigenvectors are a

stable solution under gradient ascent-descent dynamics. Second, the optimal dual variables $\boldsymbol{\beta}^*$ are proportional to the smallest Laplacian eigenvalues, meaning that with this single objective one can recover naturally **both** eigenvectors and eigenvalues of $\mathbf{L}$ (see the next section for the formalization of these claims).

Something to note is that the standard augmented Lagrangian has been discussed in the literature as a potential approach for learning eigenvectors of linear operators, but it was dismissed due to lack of empirical stability (Pfau et al., 2019). ALLO overcomes this problem through the introduction of the stop-gradient operators, which are responsible for breaking the symmetry of the Laplacian eigenvector rotations, in a similar way as how gradient masking is used in spectral inference networks (Pfau et al., 2019).

3.3 Barrier Dynamics

For the introduced max-min objective to work, in theory, b has to be larger than a **finite** value that depends on the specific Laplacian $\mathbf{L}$. Moreover, if $f(\mathbf{P}_\pi)$ in the definition of $\mathbf{L}$ is a stochastic matrix, which is the case for all of the typical definitions mentioned previously, one can exactly determine a lower bound for b, as proved in the next section. In practice, however, we found that b still needs to be increased. In our experiments, we do so in a gradient ascent fashion, just as with the dual variables.

Chapter 4

Theoretical Results

To prove the soundness of the proposed max-min objective, we need to show two things: 1) that the equilibria of this objective correspond to the desired eigensystem of the Laplacian, and 2) that this equilibria are stable under stochastic gradient ascent-descent dynamics.

4.1 Stationary solutions

As an initial motivation, the following Lemma deals with the first point in the *stationary setting*. While it is already known that the set of solutions to the graph drawing optimization problem (2.3) corresponds to the rotations of the smallest eigenvectors of $\mathbf{L}$, the Lemma considers a primal-dual perspective of the problem that allows one to relate the dual variables with the eigenvalues of $\mathbf{L}$. This identification is relevant since previous methods are not able to recover the eigenvalues.

Lemma 1. *Consider a symmetric matrix* $\mathbf{L} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ *with increasing, and possibly repeated, eigenvalues* $\lambda_1 \leq \cdots \leq \lambda_{|\mathcal{S}|}$*, and a corresponding sequence of*

eigenvectors $(\mathbf{e}_i)_{i=1}^{|\mathcal{S}|}$. *Then, given a number of components,* $1 \leq d \leq |\mathcal{S}|$, *the pair* $(\mathbf{u}_i^*)_{i=1}^d, (\beta_{jk}^*)_{1\leq k\leq j\leq d}$, *where* $\mathbf{u}_i^* = \mathbf{e}_i$ *and* $\beta_{jk}^* = -\lambda_j \delta_{jk}$, *is a solution to the primal-dual pair of optimization problems corresponding to the spectral graph drawing optimization problem (2.3). Furthermore, any other primal solution corresponds to a rotation of the eigenvectors* $(\mathbf{e}_i)_{i=1}^d$.

Proof. Let $\beta_{jk} \in \mathbb{R}$ denote the dual variables associated to the constraints of the optimization problem (2.3), and $\mathcal{L}$ be the corresponding Lagrangian function:

$$\mathcal{L}((\mathbf{u}_i)_i, (\beta_{jk})_{k\leq j}) := \sum_{i=1}^{d} \langle \mathbf{u}_i, \mathbf{L}\mathbf{u}_i \rangle + \sum_{j=1}^{d} \sum_{k=1}^{j} \beta_{jk}(\langle \mathbf{u}_j, \mathbf{u}_k \rangle - \delta_{jk}) \,.$$

Then, any pair of solutions $(\mathbf{u}_i^*)_i, (\beta_{jk}^*)_{k\leq j}$ must satisfy the Karush-Kunh-Tucker conditions. In particular, the gradient of the Lagrangian should be 0 for both primal and dual variables:

$$\nabla_{\mathbf{u}_i} \mathcal{L}((\mathbf{u}_i^*)_i, (\beta_{jk}^*)_{k\leq j}) = 2\mathbf{L}\mathbf{u}_i^* + \sum_{k=1}^{i} \beta_{ik}^* \mathbf{u}_k^* + \sum_{j=i}^{d} \beta_{ji}^* \mathbf{u}_j^* = 0 \,, \;\; \forall\, 1 \leq i \leq d \,; \quad (4.1)$$

$$\nabla_{\beta_{jk}} L((\mathbf{u}_i^*)_i, (\beta_{jk}^*)_{k\leq j}) = \langle \mathbf{u}_j^*, \mathbf{u}_k^* \rangle - \delta_{jk} = 0 \,, \;\; \forall\, 1 \leq k \leq j \leq d \,. \quad (4.2)$$

The Equation (4.2) does not introduce new information since it only asks again for the solution set $(\mathbf{u}_i^*)_{i=1}^d$ to form an orthonormal basis. Equation (4.1) is telling us something more interesting. It asks $\mathbf{L}\mathbf{u}_i^*$ to be a linear combination of the vectors $(\mathbf{u}_i^*)_{i=1}^d$, i.e., it implies that $\mathbf{L}$ always maps $\mathbf{u}_i^*$ back to the space spanned by the basis. Since this is true for all i, the span of $(\mathbf{u}_i^*)_{i=1}^d$ must coincide with the span of the eigenvectors $(\mathbf{e}_{\sigma(i)})_{i=1}^d$, for some permutation $\sigma : \mathcal{S} \to \mathcal{S}$, as proved in Proposition 1 in the Appendix. Intuitively, if this was not the case, then the scaling effect of λ_j along some $\mathbf{e}_{\sigma(j)}$ would take points that are originally in $\text{span}(\mathbf{u}_i^*)_{i=1}^d$ outside of this hyperplane.

Since we know that the span of the desired basis is $\text{span}(\mathbf{e}_{\sigma(i)})_{i=1}^{d}$, for some permutation $\sigma : \mathcal{S} \to \mathcal{S}$, we can restrict the solution to be a set of eigenvectors of $\mathbf{L}$. The function being minimized then becomes $\sum_{i=1}^{d} \lambda_{\sigma(i)}$, which implies that a primal solution is the set of d smallest eigenvectors. Now, any rotation of this minimizer results in the same loss and is also in $\text{span}(\mathbf{e}_i))_{i=1}^{d}$, which implies that any rotation of these eigenvectors is also a primal solution.

Considering the primal solution where $\mathbf{u}_i^* = \mathbf{e}_i$, Equation (4.1) becomes:

$$\nabla_{\mathbf{u}_i} \mathcal{L}((\mathbf{e}_i)_i, (\beta_{jk}^*)_{k \leq j}) = 2(\lambda_i + \beta_{ii}^*)\mathbf{e}_i + \sum_{k=1}^{i-1} \beta_{ik}^* \mathbf{e}_k + \sum_{j=i+1}^{d} \beta_{ji}^* \mathbf{e}_j = 0\,, \;\; \forall\, 1 \leq i \leq d\,.$$

Since the eigenvectors are normal to each other, the coefficients all must be 0, which implies that the corresponding dual solution is $\beta_{ii}^* = -\lambda_i$ and $\beta_{jk}^* = 0$ for $j \neq k$. □

4.2 Gradient ascent-descent equilibria

Now that we know that the primal-dual pair of optimization problems associated to (2.3) has as a solution the smallest eigensystem of the Laplacian, the following Lemma shows that the equilibria of the max-min objective (3.2) coincides only with this solution, up to a constant, and any possible permutation of the eigenvectors, **but not with its rotations**.

Lemma 2. *The pair $\mathbf{u}^*, \boldsymbol{\beta}^*$ is an equilibrium pair of the max-min objective (3.2), under gradient ascent-descent dynamics, if and only if $\mathbf{u}^*$ coincides with a subset of eigenvectors of the Laplacian $(\mathbf{e}_{\sigma(i)})_{i=1}^{d}$, for some permutation $\sigma : \mathcal{S} \to \mathcal{S}$, and $\beta_{jk}^* = -2\lambda_{\sigma(j)}\delta_{jk}$.*

Proof. Let us denote $\mathcal{L}$ the objective (3.2). Then, we have the following gradient

ascent-descent dynamical system:

$$\mathbf{u}_i[t+1] = \mathbf{u}_i[t] - \alpha_{\text{primal}} \cdot \mathbf{g}_{\mathbf{u}_i}(\mathbf{u}[t], \boldsymbol{\beta}[t])\,,\ \forall\, 1 \leq i \leq d, \tag{4.3}$$

$$\beta_{jk}[t+1] = \beta_{jk}[t] + \alpha_{\text{dual}} \cdot \frac{\partial \mathcal{L}}{\partial \beta_{jk}}(\mathbf{u}[t], \boldsymbol{\beta}[t])\,,\ \forall\, 1 \leq k \leq j \leq d\,, \tag{4.4}$$

where $t \in \mathbb{N}$ is the discrete time index, $\alpha_{\text{primal}}, \alpha_{\text{dual}} > 0$ are step sizes, and $\mathbf{g}_{\mathbf{u}_i}$ is the gradient of $\mathcal{L}$ with respect to $\mathbf{u}_i$, taking into account the stop-gradient operator. We avoid the notation $\nabla_{\mathbf{u}_i}\mathcal{L}$ to emphasize that $\mathbf{g}_{\mathbf{u}_i}$ is not a real gradient, but a chosen direction that ignores what is inside the stop-gradient operator.

The equilibria of our system corresponds to those points for which $\mathbf{u}_i^*[t+1] = \mathbf{u}_i^*[t]$ and $\beta_{jk}^*[t+1] = \beta_{jk}^*[t]$. Hence,

$$\mathbf{g}_{\mathbf{u}_i}(\mathbf{u}^*, \boldsymbol{\beta}^*) = 2\mathbf{L}\mathbf{u}_i^* + \sum_{j=1}^{i} \beta_{ij}\mathbf{u}_j^* + 2b\sum_{j=1}^{i}(\langle \mathbf{u}_i^*, \mathbf{u}_j^* \rangle - \delta_{ij})\mathbf{u}_j^* = \mathbf{0}\,,\ \forall 1 \leq i \leq d, \tag{4.5}$$

$$\frac{\partial \mathcal{L}}{\partial \beta_{jk}}(\mathbf{u}^*, \boldsymbol{\beta}^*) = \langle \mathbf{u}_j^*, \mathbf{u}_k^* \rangle - \delta_{jk} = 0\,,\ \forall 1 \leq k \leq j \leq d\,. \tag{4.6}$$

We proceed now by induction over i, considering that Equation (4.6) tells us that $\mathbf{u}^*$ corresponds to an orthonormal basis. For the base case $i = 1$ we have:

$$\mathbf{g}_{\mathbf{u}_1}(\mathbf{u}^*, \boldsymbol{\beta}^*) = 2\mathbf{L}\mathbf{u}_1^* + \beta_{1,1}\mathbf{u}_1^* = 0\,.$$

Thus, we can conclude that $\mathbf{u}_1$ is an eigenvector $\mathbf{e}_{\sigma(1)}$ of the Laplacian, and that $\beta_{1,1}$ corresponds to its eigenvalue, specifically $\beta_{1,1} = -2\lambda_{\sigma(1)}$, for some permutation $\sigma : \mathcal{S} \to \mathcal{S}$. Now let us suppose that $\mathbf{u}_j = \mathbf{e}_{\sigma(j)}$ and $\beta_{jk} = -2\lambda_{\sigma(j)}\delta_{jk}$

for $j < i$. Equation (4.5) for i then becomes:

$$\mathbf{g}_{\mathbf{u}_i}(\mathbf{u}^*, \boldsymbol{\beta}^*) = 2\mathbf{L}\mathbf{u}_i^* + \beta_{ii}\mathbf{u}_i^* + \sum_{j=1}^{i-1} \beta_{ij}\mathbf{e}_{\sigma(j)} = 0\,.$$

In general, we can express $\mathbf{u}_i^*$ as the linear combination $\mathbf{u}_i^* = \sum_{j=1}^{|\mathcal{S}|} c_{ij}\mathbf{e}_{\sigma(j)}$ since the eigenvectors of the Laplacian form a basis. Also, given that $\langle \mathbf{u}_j, \mathbf{u}_k \rangle = 0$, we have that $c_{ij} = 0$ for $j < i$. Hence,

$$2\sum_{j=i}^{|\mathcal{S}|} c_{ij}\mathbf{L}\mathbf{e}_{\sigma(j)} + \beta_{ii}\sum_{j=i}^{|\mathcal{S}|} c_{ij}\mathbf{e}_{\sigma(j)} + \sum_{j=1}^{i-1} \beta_{ij}\mathbf{e}_{\sigma(j)} = \sum_{j=i}^{|\mathcal{S}|} c_{ij}(2\lambda_{\sigma(j)} + \beta_{ii})\mathbf{e}_{\sigma(j)} + \sum_{j=1}^{i-1} \beta_{ij}\mathbf{e}_{\sigma(j)} = 0\,.$$

By orthogonality of the eigenvectors, we must have that each coefficient is 0, implying that $\beta_{ij} = 0$ and either $c_{ij} = 0$ or $\beta_{ii} = -2\lambda_{\sigma(j)}$. The last equation allows us to conclude that a pair (c_{ij}, c_{ik}) can only be different to 0 simultaneously for j, k such that $\lambda_{\sigma(j)} = \lambda_{\sigma(k)}$, i.e., $\mathbf{u}_i$ lies in the subspace of eigenvectors corresponding to the same eigenvalue, where each point is in itself an eigenvector. Thus, we can conclude, that $\mathbf{u}_i = \mathbf{e}_{\sigma(i)}$ and $\beta_{ij} = -2\lambda_i\delta_{ij}$, as desired. □

As a Corollary to Lemma 2, let us suppose that we fix all the dual variables to 0, i.e., $\beta_{jk} = 0$. Then, we will obtain that the constraints of the original optimization problem (2.3) must be violated for any possible equilibrium point. This explains why optimizing GGDO in Equation (2.6) may converge to undesirable rotations of the Laplacian eigenvectors, even when the smallest eigenvectors are the unique solution of the original associated constrained optimization problem.

Corollary 1. *The point* $\mathbf{u}^*$ *is an equilibrium point of the objective (2.4) or the objective (2.6), under gradient descent dynamics, if and only if for any* $1 \leq i \leq d$ *there exists a* $1 \leq j \leq d$ *such that* $\langle \mathbf{u}_i^*, \mathbf{u}_j^* \rangle \neq \delta_{ij}$. *That is, the equilibrium is guaranteed to be different to the eigenvectors of the Laplacian.*

4.3 Stable equilibrium

Finally, we prove that even when all permutations of the Laplacian eigenvectors are equilibrium points of the proposed objective (3.2), only the one corresponding to the ordered smallest eigenvectors and its eigenvalues is stable. This is in contrast with GGDO.

Theorem 1. *The only permutation in Lemma 2 that corresponds to an stable equilibrium point of the max-min objective (3.2) is the identity permutation, under an appropriate selection of the barrier coefficient b. That is, there exist a finite barrier coefficient such that* $\mathbf{u}_i^* = \mathbf{e}_i$ *and* $\beta_{jk}^* = -2\lambda_j\delta_{jk}$ *correspond to the only stable equilibrium pair, where* λ_i *is the* $i-th$ *smallest eigenvalue of the Laplacian and* $\mathbf{e}_i$ *its corresponding eigenvector. In particular, any* $b > 2$ *guarantees stability.*

The main idea of the proof is as follows. We have that $\mathbf{g}_{\mathbf{u}_i}$ in Equation (4.3) and $\partial\mathcal{L}/\partial\beta_{jk}$ in Equation (4.4) define the chosen ascent-descent direction. Concatenating these vectors and scalars in a single vector $\mathbf{g}(\mathbf{u}, \boldsymbol{\beta})$, the stability of the dynamics can be determined from the Jacobian matrix $J(\mathbf{g})$. Specifically, if all the eigenvalues of this matrix have a positive real part in the equilibrium pair $\mathbf{u}^*, \boldsymbol{\beta}^*$, we can conclude that the equilibrium is stable. If there is one eigenvalue with negative real part, then it is unstable (see Chicone, 2006; Sastry, 2013; Mazumdar et al., 2020). As proven below, for any pair $1 \leq i < j \leq |\mathcal{S}|$, there exists a real eigenvalue proportional to $\lambda_{\sigma(j)} - \lambda_i$. This means that, unless the σ permutation is the identity, there will be at least one negative eigenvalue and the equilibrium corresponding to this permutation will be unstable.

Proof. Let us define the following vectors defining the descent directions for $\mathbf{u}$

and $\boldsymbol{\beta}$:

$$\mathbf{g}_{\mathbf{u}} = \begin{bmatrix} \mathbf{g}_{\mathbf{u}_1} \\ \mathbf{g}_{\mathbf{u}_2} \\ \vdots \\ \mathbf{g}_{\mathbf{u}_d} \end{bmatrix}, \quad \mathbf{g}_{\boldsymbol{\beta}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \beta_{1,1}} \\ \frac{\partial \mathcal{L}}{\partial \beta_{2,1}} \\ \frac{\partial \mathcal{L}}{\partial \beta_{2,2}} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \beta_{d,1}} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \beta_{d,d}} \end{bmatrix}.$$

Then, the global ascent-descent direction can be represented by the vector

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}_{\mathbf{u}} \\ -\mathbf{g}_{\boldsymbol{\beta}} \end{bmatrix}.$$

To determine the stability of any equilibrium point of the ascent-descent dynamics introduced in Lemma 2, we only need to calculate the Jacobian of $\mathbf{g}$, the matrix $\mathbf{J} := J(\mathbf{g})$ whose rows correspond to the gradients of each entry of $\mathbf{g}$, and determine its eigenvalues (Chicone, 2006).

We proceed to take the gradients of Equation 4.5 and 4.6:

$$\mathbf{J}_{ij}(\mathbf{u}, \boldsymbol{\beta}) := (\nabla_{\mathbf{u}_i} \mathbf{g}_{\mathbf{u}_j}(\mathbf{u}, \boldsymbol{\beta})^\top)^\top$$

$$= \begin{cases} 2\mathbf{L} + \beta_{ii}\mathbf{I} + 2b\big[(\langle \mathbf{u}_i, \mathbf{u}_i \rangle - 1)\mathbf{I} + 2\mathbf{u}_i \otimes \mathbf{u}_i\big] + 2\sum_{k=1}^{i-1} b\mathbf{u}_k \otimes \mathbf{u}_k\,, & \text{if } i = j\,; \\ \beta_{ij}\mathbf{I} + 2b\big(\langle \mathbf{u}_i, \mathbf{u}_j \rangle \mathbf{I} + \mathbf{u}_i \otimes \mathbf{u}_j\big)\,, & \text{if } i > j\,; \\ \mathbf{0}\,, & \text{if } i < j\,; \end{cases}$$

$$\mathbf{J}_{k\beta_{ij}}(\mathbf{u},\boldsymbol{\beta}) := \left(-\nabla_{\mathbf{u}_k} \frac{\partial \mathcal{L}}{\partial \beta_{ij}}(\mathbf{u},\boldsymbol{\beta}) \right)^\top = -\mathbf{u}_j^\top \delta_{ik} - \mathbf{u}_i^\top \delta_{jk}\,;$$

$$\mathbf{J}_{\beta_{jk}i}(\mathbf{u},\boldsymbol{\beta}) := \frac{\partial}{\partial \beta_{jk}} \mathbf{g}_{\mathbf{u}_i}(\mathbf{u},\boldsymbol{\beta}) = \mathbf{u}_i \delta_{ij}\delta_{ik} + \mathbf{u}_k \delta_{ij}(1-\delta_{jk})\,;$$

$$\mathbf{J}_{\beta_{k\ell}\beta_{ij}}(\mathbf{u},\boldsymbol{\beta}) := \frac{\partial^2 \mathcal{L}}{\partial \beta_{k\ell}\partial \beta_{ij}}(\mathbf{u},\boldsymbol{\beta}) = 0\,.$$

Then, we have that in any equilibrium point $\mathbf{u}^*, \boldsymbol{\beta}^*$, i.e., in a permutation σ of the Laplacian eigensystem (as per Lemma 2), the Jacobian satisfies:

$$\mathbf{J}_{ij}(\mathbf{u}^*,\boldsymbol{\beta}^*) = \begin{cases} 2\mathbf{L} - 2\lambda_i \mathbf{I} + 4b\mathbf{e}_{\sigma(i)} \otimes \mathbf{e}_{\sigma(i)} + 2\sum_{k=1}^{i-1} b\mathbf{e}_{\sigma(k)} \otimes \mathbf{e}_{\sigma(k)}\,, & \text{if } i = j\,; \\ 2b\mathbf{e}_{\sigma(i)} \otimes \mathbf{e}_{\sigma(j)}\,, & \text{if } i > j\,; \\ \mathbf{0}\,, & \text{if } i < j\,; \end{cases}$$

$$\mathbf{J}_{k\beta_{ij}}(\mathbf{u}^*,\boldsymbol{\beta}^*) = -\mathbf{e}_{\sigma(j)}^\top \delta_{ik} - \mathbf{e}_{\sigma(i)}^\top \delta_{jk}\,;$$

$$\mathbf{J}_{\beta_{jk}i}(\mathbf{u}^*,\boldsymbol{\beta}^*) = \mathbf{e}_{\sigma(i)} \delta_{ij}\delta_{ik} + \mathbf{e}_{\sigma(k)} \delta_{ij}(1-\delta_{jk})\,;$$

$$\mathbf{J}_{\beta_{k\ell}\beta_{ij}}(\mathbf{u}^*,\boldsymbol{\beta}^*) = 0\,.$$

Now, we determine the eigenvalues of this Jacobian. For this, we need to solve the system:

$$\mathbf{J}\mathbf{v} = \eta \mathbf{v}\,, \tag{4.7}$$

where η denotes an eigenvalue of the Jacobian and $\mathbf{v}$ its corresponding eigenvector.

To facilitate the solution of this system, we use the following notation:

$$\mathbf{v} = \begin{bmatrix} \mathbf{w} \\ \boldsymbol{\nu} \end{bmatrix}, \quad \mathbf{v_u} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_d \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} \nu_{1,1} \\ \nu_{2,1} \\ \nu_{2,2} \\ \vdots \\ \nu_{d,1} \\ \vdots \\ \nu_{d,d} \end{bmatrix},$$

where $\mathbf{w}_i \in \mathbb{R}^{|\mathcal{S}|}$, for all $1 \leq i \leq d$, and $\nu_{jk} \in \mathbb{R}$, for all $1 \leq k \leq j \leq d$. With this, the eigenvalue system (4.7) becomes:

$$\begin{cases} \sum_{j=1}^{d} \mathbf{J}_{ji}\mathbf{w}_j + \sum_{j=1}^{d}\sum_{k=1}^{j} \mathbf{J}_{\beta_{jk}i}\nu_{jk} = \eta\mathbf{w}_i\,, & \forall\, 1 \leq i \leq d\,; \\ \sum_{k=1}^{d} \mathbf{J}_{k\beta_{ij}}\mathbf{w}_k = \eta\nu_{ij}\,, & \forall\, 1 \leq j \leq i \leq d\,. \end{cases} \tag{4.8}$$

Since the Laplacian eigenvectors form a basis, we have the decomposition $\mathbf{w}_i = \sum_{j=1}^{|\mathcal{S}|} c_{ij}\mathbf{e}_{\sigma(j)}$, for some sequence of reals $(c_{ij})_{j=1}^{|\mathcal{S}|}$. Hence, replacing the values of the Jacobian components in the upper equation of the system (4.8), we obtain:

$$\sum_{j=1}^{i-1} 2b(\mathbf{e}_{\sigma(i)} \otimes \mathbf{e}_{\sigma(j)})\mathbf{w}_j + \left(2\mathbf{L} - 2\lambda_i\mathbf{I} + 4b\mathbf{e}_{\sigma(i)} \otimes \mathbf{e}_{\sigma(i)} + 2\sum_{k=1}^{i-1} b\mathbf{e}_{\sigma(k)} \otimes \mathbf{e}_{\sigma(k)}\right)\mathbf{w}_i + \cdots$$

$$\cdots + \sum_{j=1}^{d}\sum_{k=1}^{j}\left(\mathbf{e}_{\sigma(i)}\delta_{ij}\delta_{ik} + \mathbf{e}_{\sigma(k)}\delta_{ij}(1-\delta_{jk})\right)\nu_{jk} - \eta\mathbf{w}_i = 0$$

$$\implies \quad 2\sum_{j=1}^{i-1} bc_{ji}\mathbf{e}_{\sigma(j)} + 2\sum_{j=1}^{|\mathcal{S}|}(\lambda_j - \lambda_i)c_{ij}\mathbf{e}_{\sigma(j)} + 4b\mathbf{e}_{\sigma(i)} + \sum_{j=1}^{i-1} bc_{ij}\mathbf{e}_{\sigma(j)} + \cdots$$

$$\cdots + \nu_{ii}\mathbf{e}_{\sigma(i)} + \sum_{k=1}^{i-1}\nu_{ik}\mathbf{e}_{\sigma(k)} - \eta\sum_{j=1}^{|\mathcal{S}|} c_{ij}\mathbf{e}_{\sigma(j)} = 0\,.$$

Since the eigenvectors form a basis, we have that each coefficient in the sum of terms must be 0. Hence, we obtain the following conditions:

$$\begin{cases} c_{ij}[2(\lambda_{\sigma(j)} - \lambda_i) + 2b - \eta] = -2bc_{ji} - \nu_{ij}\,, & \forall\, 1 \le j < i \le d\,; \\ \\ c_{ii}(4b - \eta) = -\nu_{ii}\,, & \forall\, 1 \le i \le d\,; \\ \\ c_{ij}[2(\lambda_{\sigma(j)} - \lambda_i) - \eta] = 0\,, & \forall\, 1 \le i < j \le |\mathcal{S}|\,. \end{cases} \tag{4.9}$$

Each of these conditions specify the possible eigenvalues of the Jacobian matrix $\mathbf{J}$. First and foremost, the third condition tells us that $\eta = 2(\lambda_{\sigma(j)} - \lambda_i)$ is an eigenvalue independent of b, for any possible pair $i \le j$. Since we are supposing the eigenvalues are increasing with their index, for the eigenvalues to be positive, the permutation $\sigma : \mathcal{S} \to \mathcal{S}$ must preserve the order for all indexes, which only can be true for the identity permutation. That is, all the Laplacian eigenvector permutations that are not sorted are unstable.

In addition, deriving the rest of the eigenvalues from the remaining two conditions in (4.9) and the second set of equations of the system (4.8), we obtain a

lower bound for b that guarantees the stability of the Laplacian representation. In particular, from the second set of equations of the system (4.8) we can obtain a relationship between the coefficients c_{ij} and c_{ji} with ν_{ij}, for all $1 \leq j \leq i \leq d$:

$$\begin{aligned}\sum_{k=1}^{d} \mathbf{J}_{j\beta_{ij}} \mathbf{w}_k &= \sum_{k=1}^{d} \Big(- \mathbf{e}_{\sigma(j)}^{\top} \delta_{ik} - \mathbf{e}_{\sigma(i)}^{\top} \delta_{jk} \Big) \Big(\sum_{\ell=1}^{|\mathcal{S}|} c_{k\ell} \mathbf{e}_{\sigma(\ell)} \Big) \\ &= -c_{ij} - c_{ji} = \eta \nu_{ij} \,. \end{aligned} \tag{4.10}$$

Replacing this into the second condition in (4.9) we get that $\eta = 2b \pm 2\sqrt{b^2 - \frac{1}{2}}$. These set of eigenvalues (two for each i) always have a positive real part, as long as b is strictly positive. In addition, if $b \geq \frac{1}{2}$, we get purely real eigenvalues, which are associated with a less oscillatory behavior (see Sastry, 2013).

Finally, if we assume that $\eta \neq 2(\lambda_{\sigma(i)} - \lambda_j)$ for $j < i$ (i.e., η is not an eigenvalue already considered), we must have that $c_{ji} = 0$, and so, by (4.10), $-c_{ij} = \eta \nu_{ij}$. Replacing this into the first condition in (4.9), we get that $\eta = (\lambda_{\sigma(j)} - \lambda_i) + b \pm \sqrt{[(\lambda_{\sigma(j)} - \lambda_i) + b]^2 - 1} \geq -2 + b - \sqrt{[(\lambda_{\sigma(j)} - \lambda_i) + b]^2 - 1}$. Thus, if b is larger than the maximal eigenvalue difference for the first d eigenvalues of $\mathbf{L}$, we have guaranteed that these eigenvalues of the Jacobian will be positive. Furthermore, since the eigenvalues are restricted to the range $[0, 2]$, we have that $b > 2$ ensures a strict stability of the Laplacian representation. □

Chapter 5

Experiments

We evaluate three different aspects of the proposed max-min objective: robustness, eigenvalue accuracy, and the necessity of each of the components of the proposed objective. For robustness, we determine the quality of the Laplacian representation for different initial values of the barrier coefficient and for environments of different sizes, and we contrast them against the baseline, GGDO. For eigenvalue accuracy, we compare the estimates obtained with the dual variables of our objective against those obtained from the original graph drawing loss function in (2.3), as described by Wang et al. (2023). Finally, we perform two different ablations to determine which components of our objective are empirically responsible for successful Laplacian representation learning.

5.1 Robustness

In this section, we cover the three sets of experiments performed to test ALLO's robustness.

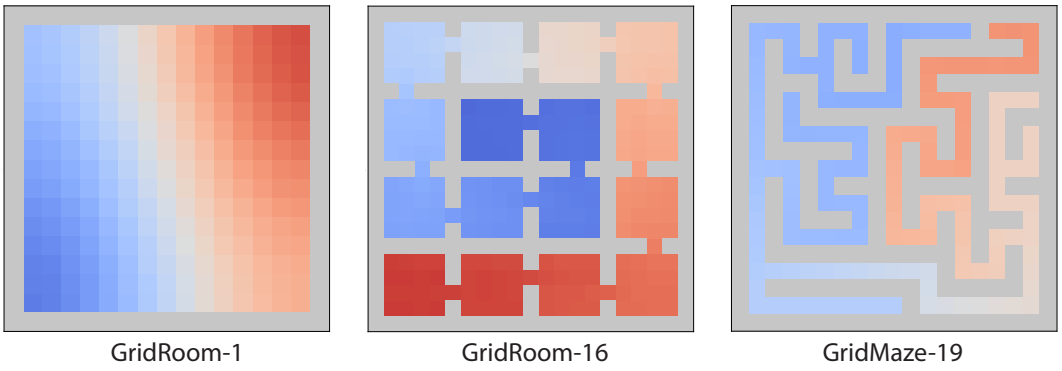


Figure 5.1: Examples of grid environments. The color corresponds to the second smallest eigenvector of the Laplacian learned by ALLO. Red represents negative values; and blue, positive ones.

5.1.1 Barrier coefficient sweep

We start by considering the grid environments shown in Figure 5.1. We generate $200,000$ transition samples in each of them from a uniform random policy and a uniform initial state distribution. We use the (x, y) coordinates of the agent's position as inputs to a fully-connected neural network $\phi_{\boldsymbol{\theta}} : \mathbb{R}^2 \to \mathbb{R}^d$, parameterized by $\boldsymbol{\theta}$, with 3 layers of 256 hidden units to approximate the $d-$dimensional Laplacian representation ϕ, where $d = 11$. The network is trained using stochastic gradient descent with our objective, as explained in Section 2.4.5. We repeat this process with different initial barrier coefficients, using the same values as in Figure 2.2.

Figure 5.2 shows the average cosine similarity of eigenvectors found using ALLO compared to the true Laplacian eigenvectors. In all three environments ALLO learns close approximations of the smallest $d-$eigenvectors in fewer gradient updates than GGDO (see Figure 2.2), without a strong dependence on the chosen initial barrier coefficients.

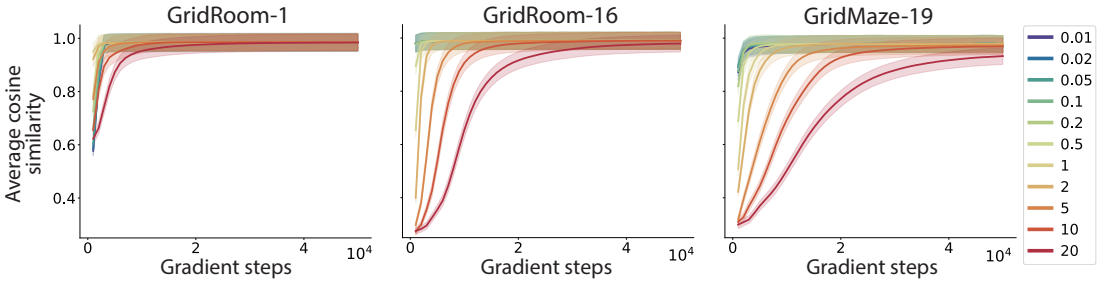


Figure 5.2: Average cosine similarity between the true Laplacian representation and ALLO. Each curve corresponds to a different initial barrier coefficient b, averaged over 60 seeds. The thickest curve corresponds to the best coefficient and the shaded regions, to a 95% confidence interval.

5.1.2 Environment robustness

As a second and more conclusive experiment, we select the barrier coefficient that displayed the best performance for GGDO across the three previous environments ($b = 2.0$), and the best barrier increasing rate, α_{barrier}, for our method across the same environments ($\alpha_{\text{barrier}} = 0.01$). Then, we use these values to learn the Laplacian representation in the 12 different grid environments shown in Figure 5.3. Note that each environment has a different number of states, ranging from $|\mathcal{S}| = 11$ (`GridMaze-7`) to $|\mathcal{S}| = 1088$ (`GridRoom-64`), and also a particular topology, presenting both totally disconnected regions (e.g., `GridMaze-32`), highly connected regions (e.g., `GridRoom-1`), and symmetries (compare `GridRoom-4` and `GridRoomSym-4`). Given that in this case some environments are considerably large, the numbers of transition samples used is 1 million.

Figure 5.4a compares the average cosine similarities obtained with each method. In particular, it shows the mean difference of the average cosine similarities across 60 seeds. Noticeably, the baseline fails completely in the two smallest environments (i.e., `GridMaze-7` and `GridMaze-9`), and it also fails partially in the two largest ones (i.e., `GridMaze-32` and `GridRoom-64`). In contrast, ALLO finds close

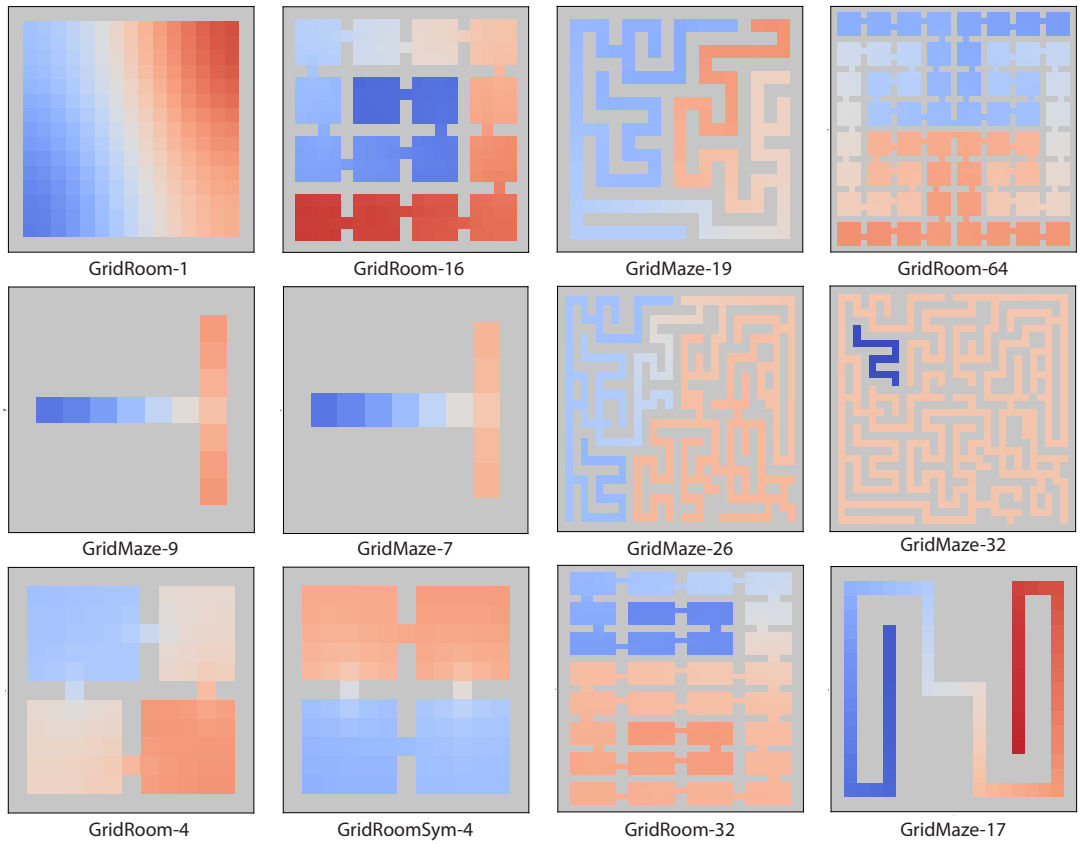
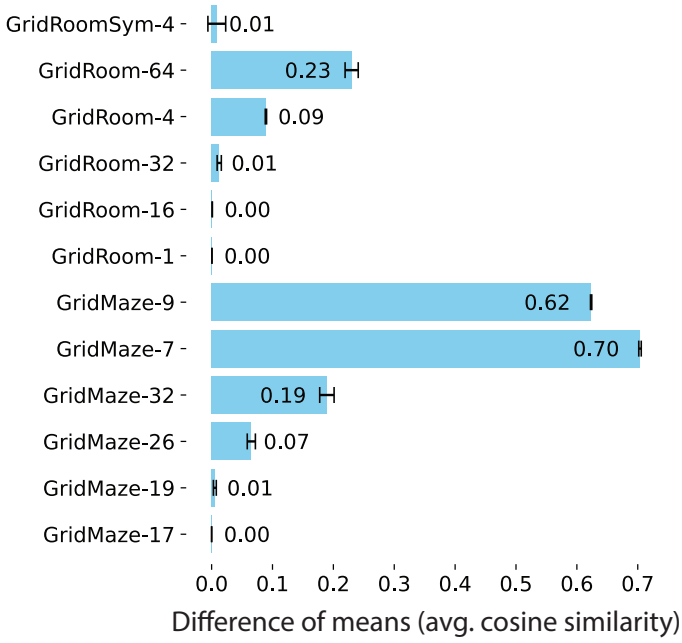
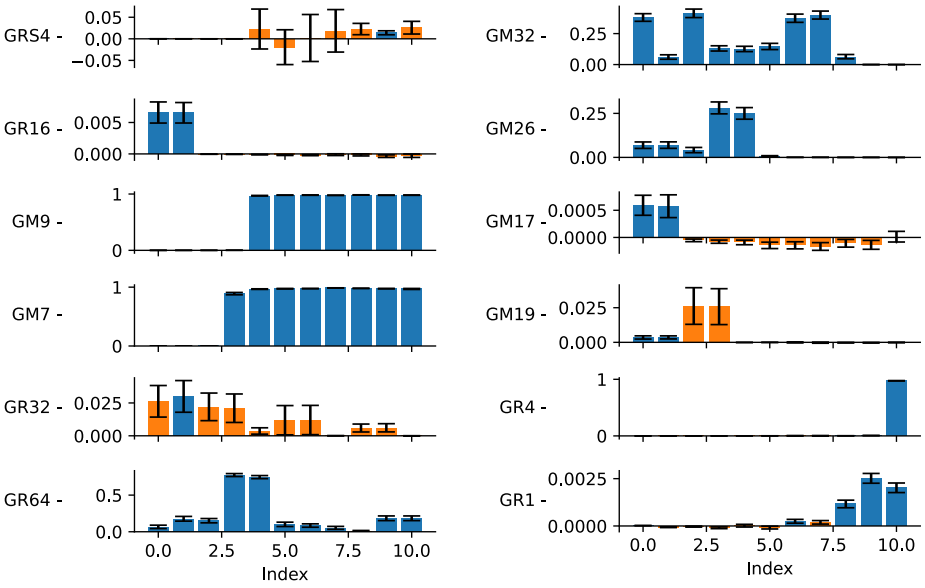


Figure 5.3: Grid environments where the Laplacian representation is learned with both GGDO and ALLO. Color corresponds to the second smallest eigenvector of the Laplacian learned by ALLO.

approximations of the true Laplacian representation across all environments, with the exception of `GridRoomSym-4`, where it still found a more accurate representation than GGDO. These results are statistically significant for 9 out of 12 environments, with a p-value threshold of 0.01 (see Table 6.1 in the Appendix). Again, this suggests that the proposed objective is successful in removing the untunable-hyperparameter dependence observed in GGDO. Moreover, Figure 5.4b shows the same cosine similarity comparison, but for each independent eigenvector. In the majority of the cases, the largest differences can be observed in the largest eigenvectors (e.g., in `GridMaze-9` and `GridRoom-4`), but there are instances in which the baseline has problems finding the smallest eigenvectors as well (e.g., in



(a) Average cosine similarity across the d components.



(b) Cosine similarity for each of the components. `GR` and `GM` stand for `GridRoom` and `GridMaze`. Blue bars correspond to p-values below 0.01.

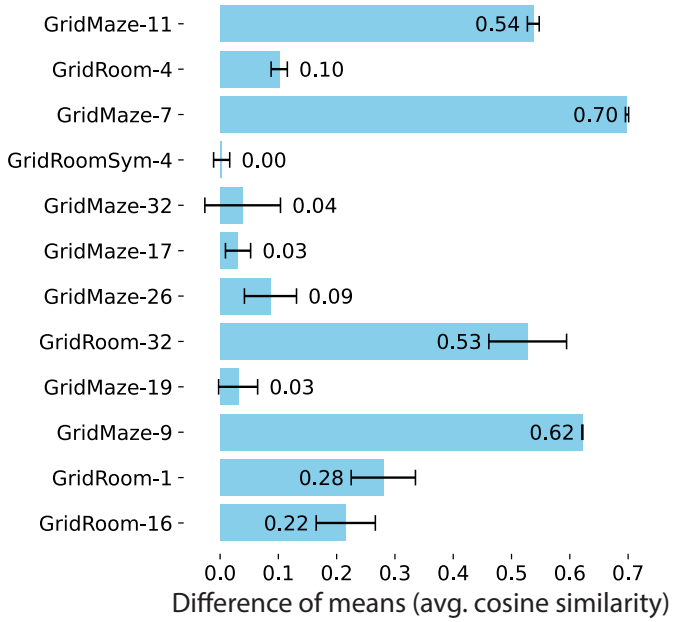
Figure 5.4: Cosine similarity difference between ALLO and GGDO when using the **(x, y) state representation**. Error bars show the standard deviation of the differences, defined as $\sqrt{\sigma^2_{\text{ALLO}}/n_{\text{ALLO}} + \sigma^2_{\text{GGDO}}/n_{\text{GGDO}}}$, where σ denotes the sample standard deviation, and n the number of seeds ($= 60$ in all cases).

`GridRoom-64` and `GridMaze-32`). These failures are particularly concerning since they indicate that the baseline could struggle in finding the best components for exploration in more complex environments, which is one of the main motivations of learning the Laplacian representation in the first place.

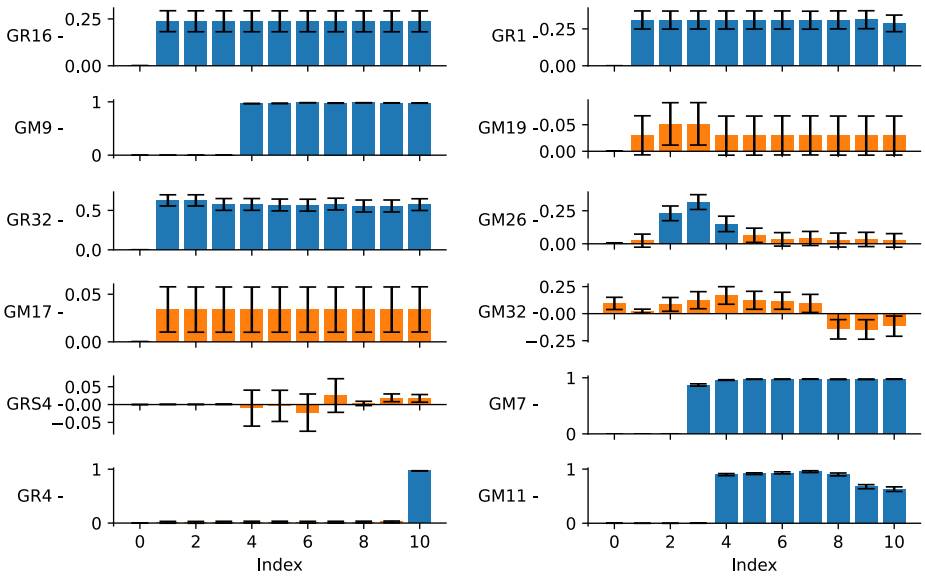
5.1.3 State representation robustness

As an additional robustness test, we changed the state representation used as input to the neural network. In particular, we used a pixel representation where each tile in the subplots of Figure 5.3 corresponds to a single pixel. In this way, the inputs ranged from a size of $7 \times 9 \times 3$ to a size of $41 \times 41 \times 3$. Correspondingly, we added two convolutional layers with no pooling, stride of 2, and kernel size of 3 to the previous fully connected network.

Figure 5.5 contains analogous cosine similarity comparisons to those observed in Figure 5.4 for the (x, y) state representation. There are three main differences with the previous results. First, the average cosine similarity difference (Figure 5.5a) is now only significant in 7 out of 12 environments, instead of 9 of 12 (see Tables 6.1 and 6.2 in the Appendix), but the difference is higher for several environments (e.g., `GridRoom-32` and `GridRoom-1`). Second, ALLO only finds the true Laplacian representation in 6 out of 12 environments, as opposed to 11 out of 12. From those 6 where perfect learning is not achieved, 4 have a cosine similarity above 0.9, one corresponds to the same environment where the Laplacian representation was not perfectly learned before, and there is one for which the similarity is only 0.61. Focusing on these results, we observed that ALLO was able to recover the Laplacian representation for some seeds, but for others the output of the neural network collapsed to a constant. This suggests that maintaining the same hyperparameters with the new state representation



(a) Average cosine similarity across the d components.



(b) Cosine similarity for each of the components. `GR` and `GM` stand for `GridRoom` and `GridMaze`. Blue bars correspond to p-values below 0.01.

Figure 5.5: Cosine similarity difference between ALLO and GGDO when using the **pixel state representation**. Error bars show the standard deviation of the differences, defined as $\sqrt{\sigma^2_{\text{ALLO}}/n_{\text{ALLO}} + \sigma^2_{\text{GGDO}}/n_{\text{GGDO}}}$, where σ denotes the sample standard deviation, and n the number of seeds ($\in [50, 60]$ in all cases).

and convolutional layers could have resulted in divergence problems unrelated to ALLO. Lastly, we can observe (see Figure 5.5b) that GGDO fails in finding the smallest non-constant eigenvector in multiple cases (e.g., in `GridRoom-16` and `GridRoom-1`), supporting the hypothesis that GGDO is not scalable to more complex settings, as opposed to ALLO, and preventing the use of the Laplacian for exploration.

5.2 Eigenvalue accuracy

The dual variables of ALLO should capture the eigenvalues of their associated eigenvectors. Here, we quantify how well they approximate the true eigenvalues in the same 12 grid environments as in Figure 5.4. In particular, we compare our eigenvalue accuracy against those found with a simple alternative method (Wang et al., 2023), based on GGDO and on Monte Carlo approximations. In short, this method uses the equivalence between the eigenvalues and the energy expression we derived in the Background, in Section 2.2.3. Figure 5.6 shows that the average relative error for the second to last eigenvalues, meaning all except one, is consistently larger across all environments when using the alternative approach, with a significance level of 0.01. This is not a surprising result given the poor results in eigenvector accuracy for GGDO. However, in several environments the error is high even for the smallest eigenvalues, despite GGDO approximations being relatively more accurate for the associated eigenvectors. Across environments and across the eigenspectrum, our proposed objective provides more accurate estimates of the eigenvalues (see Table 6.3 in the Appendix for the exact values obtained).

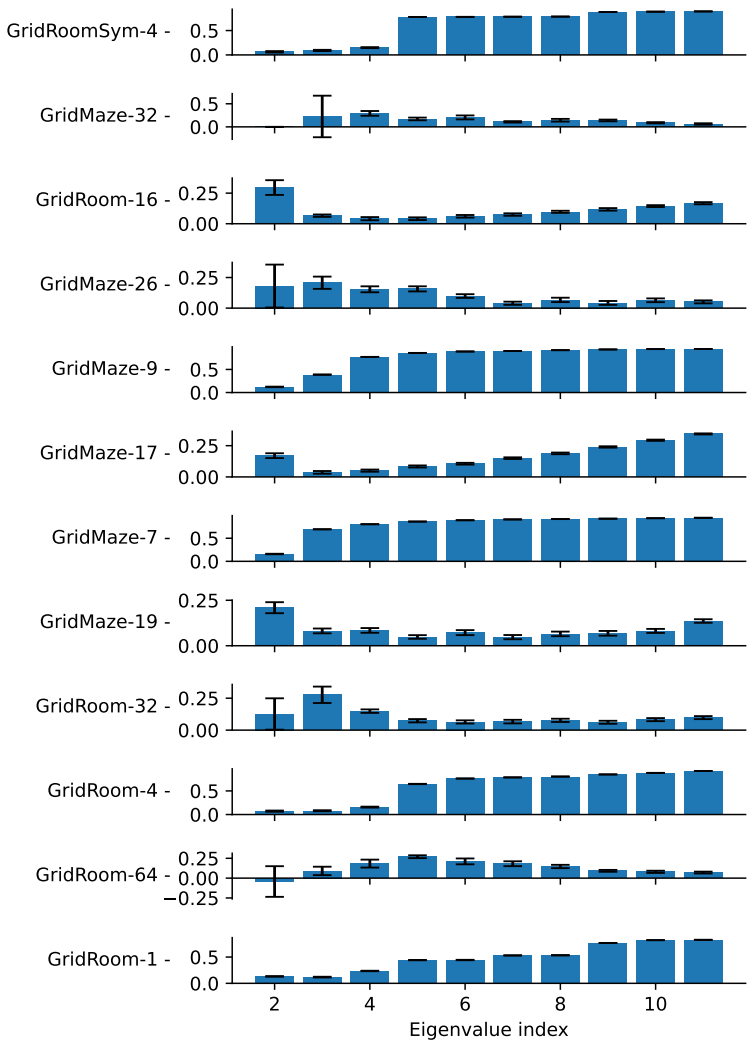


Figure 5.6: Relative errors for eigenvalue approximations between ALLO and GGDO when using the **(x, y) state representation**. Error bars show the standard deviation of the differences, corresponding to 60 different seeds. `GR` and `GM` stand for `GridRoom` and `GridMaze`. Blue bars correspond to p-values below 0.01.

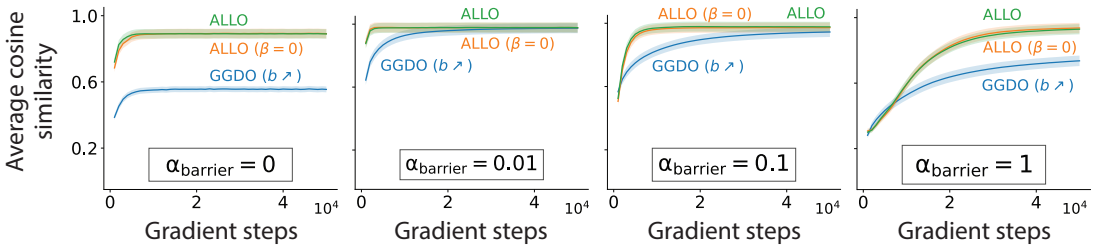


Figure 5.7: Average cosine similarity for different objectives in the environment `GridMaze-19`, for initial barrier coefficient $b = 0.1$, and for different barrier increase rates α_{barrier}.

5.3 Ablations

ALLO has three components that are different from GGDO: (1) the **stop-gradient** as a mechanism to break the symmetry, (2) the **dual variables** that penalize the linear constraints and from which we extract the eigenvalues of the graph Laplacian, and (3) the mechanism to monotonically **increase the barrier coefficient** that scales the quadratic penalty. Our theoretical results suggest that the stop-gradient operation and the dual variables are necessary, while increasing the barrier coefficient could be helpful, eventually eliminating the need for the dual variables if all one cared about was to approximate the eigenvectors of the graph Laplacian, not its eigenvalues. In this section, we perform ablation studies to validate whether these insights translate into practice when using neural networks to minimize our objective. Specifically, in `GridMaze-19`, we compare the average cosine similarity of ALLO, with the same objective but without dual variables, and with GGDO, which does not use dual variables, nor the stop gradient, nor the increasing coefficients. For completeness, we also evaluate GGDO objective with increasing coefficients.

The curves in each panel of Figure 5.7 represent the different methods we evaluate, while the different panels evaluate the impact of different rates of increase

of the barrier coefficient. Our results show that increasing the barrier coefficients is indeed important, and not increasing it, as in GDO and GGDO, actually prevents us from obtaining the true eigenvectors. It is also interesting to observe that the rate in which we increase the barrier coefficient matters empirically, but it does not prevent our solution to obtain the true eigenvectors. The importance of the stop gradient is evident when one looks at the difference in performance between GGDO and ALLO (and variants), particularly when not increasing the barrier coefficients. Finally, it is interesting to observe that the addition of the dual variables, which is essential to estimate the eigenvalues of the graph Laplacian, does not impact the performance of our approach. Based on our theoretical results, we conjecture the dual variables add stability to the learning process in larger environments, but we leave this for future work.

Chapter 6

Conclusions and Future Work

When introducing the generalized graph drawing optimization problem (2.5), Wang et al. (2021) proved theoretically that, by introducing a sequence of monotonically decreasing hyperparameters, the unique global minimizer corresponds to the desired eigensystem of the Laplacian. However, our results, exemplified by Figure 2.2, demonstrated how in practice the problem of obtaining arbitrary rotations of the smallest eigenvectors was not solved.

The shortcoming with the approach proposed by Wang et al. (2021) is that, while it is sound in the stationary setting, it does not correspond to the actual learning dynamics being used to find the Laplacian representation. Under this light, the main contributions of the work presented are the following:

1. We introduced a theoretically sound max-min objective and corresponding gradient ascent-descent dynamics for which the Laplacian representation is the unique stable equilibrium point. That is, we solved the problem of obtaining arbitrary rotations of the smallest eigenvectors in the learning setting (at least when there are no approximations), which is the setting

that is relevant in practice.

2. We shed light on how problematic it is for the customary deep learning approach to turn constrained optimization problems into unconstrained versions. In particular, introducing a constant hyperparameter that multiplies the constraints and turns them into a regularization component of the loss function (as in GGDO) can be arbitrarily harmful and invalidate the properties of the solutions to the original constrained optimization problem. In this way, our max-min approach can be potentially used in similar deep learning approximations to constrained optimization problems.

3. In line with the previous contribution, we demonstrated that it is useful to consider directly the dynamic properties of proposed learning algorithms, as opposed to the properties of possibly unreachable solutions.

4. We introduced the use of stop-gradient operators to break the symmetry between the smallest eigenvectors in the original graph drawing optimization problem (2.5). While simple, this was a novel and central element in the proof of Theorem 1.

5. Since the proofs of Lemma 2 and Theorem 1 do not depend on any particular property of the Laplacian, our proposed method allows us to learn the smallest eigenvectors of any self-adjoint linear operator whose application can be expressed as expectations. In this way, our method provides a simple alternative to the spectral inference networks proposed by Pfau et al. (2019).

As future work, there are three main directions to consider: obtaining a deeper theoretical understanding, exploiting the access to a proper approximation of the Laplacian representation, and exploring the use of eigenvalues of the graph Laplacian.

Advancing our theoretical understanding. While we proved that ALLO and its respective dynamics should converge to the Laplacian representation for any given barrier coefficient $b > 2$, empirically we saw that increasing the coefficient could be beneficial, even for the GGDO baseline. Thus, it would be valuable to better understand the impact of the change of this coefficient in the optimization process to accelerate learning. Also, our theoretical results do apply to the abstract setting and empirically hold, to a large extent, in the neural network-based function approximation setting. Nonetheless, it is still necessary to analyze the effect of introducing function approximators since the minima of ALLO could change and even become unstable, as observed in some cases when the pixel representation of the state was used. Moreover, we are assuming an offline learning setting where an agent collects transition samples and only then they are used to learn the Laplacian representation. It is relevant to understand how to make this process work in an online setting. Two reasons are that in the offline-learning setting we are completely avoiding the problem of exploration, by assuming that we can uniformly sample the initial state, and the continual learning problem, given that we are implicitly assuming the policy and the topology of the state space are fixed.

Exploiting the proper Laplacian representation. Now that we have a theoretically sound method to learn the Laplacian representation, it would be exciting to see the impact on algorithms that rely on it. In particular, there could be improvements in exploration thanks to better, i.e., more accurate, eigenoption learning (e.g., applying ALLO to the results of Klissarov & Machado, 2023). Additionally, there is potential in combining the deep learning Laplacian representation with tile-coding function approximators, and similar non-deep learning approximators that use localized basis functions, for value function estimation (e.g.,

see Chapter 9 in Sutton & Barto, 2018). The reason for this is that the Laplacian representation induces a vector space where "spatial" distance corresponds to temporal distance. Hence, it becomes reasonable to cluster together states that are close, since their values should not change too abruptly as a consequence of the Bellman equation.

Exploring the use of eigenvalues. Finally, since we can now obtain the eigenvalues of the graph Laplacian, it would be interesting to see how they could be leveraged, e.g., as an emphasis vector for feature representations or as a proxy for the duration of temporally-extended actions discovered from the Laplacian.

References

Yasin Abbasi-Yadkori, András György, and Nevena Lazic. A New Look at Dynamic Regret for Non-Stationary Stochastic Bandits. *arXiv*, 2022. URL https://arxiv.org/abs/2201.06532.

David Abel, Dilip Arumugam, Lucas Lehnert, and Michael L. Littman. State Abstractions for Lifelong Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2018. URL http://proceedings.mlr.press/v80/abel18a.html.

David Abel, Nate Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael L. Littman. Value Preserving State-Action Abstractions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020. URL http://proceedings.mlr.press/v108/abel20a.html.

Richard Bellman. On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences*, 1952. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1063639.

Vladimir Igorevich Bogachev and Maria Aparecida Soares Ruas. *Measure Theory*, 2007.

Michael Bowling, John D. Martin, David Abel, and Will Dabney. Settling the Re-

ward Hypothesis. In *International Conference on Machine Learning (ICML)*, 2023. URL https://proceedings.mlr.press/v202/bowling23a.html.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 2002. URL http://jmlr.org/papers/v3/brafman02a.html.

Emma Brunskill and Lihong Li. PAC-inspired Option Discovery in Lifelong Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2014. URL http://proceedings.mlr.press/v32/brunskill14.html.

Carmen Chicone. *Ordinary Differential Equations with Applications*, 2006.

Peter Dayan. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 1993. URL https://doi.org/10.1162/neco.1993.5.4.613.

Diego Gomez, Michael Bowling, and Marlos C. Machado. Proper Laplacian Representation Learning. *arXiv*, 2023a. URL https://doi.org/10.48550/arXiv.2310.10833.

Diego Gomez, Nicanor Quijano, and Luis Felipe Giraldo. Information Optimization and Transferable State Abstractions in Deep Reinforcement Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023b. URL https://doi.org/10.1109/TPAMI.2022.3200726.

Sham M. Kakade. On the Sample Complexity of Reinforcement Learning. PhD Thesis, University of College London, 2003. URL https://homes.cs.washington.edu/~sham/papers/thesis/sham_thesis.pdf.

Michael J. Kearns and Satinder Singh. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning*, 2002. URL https://doi.org/10.1023/A:1017984413808.

Martin Klissarov and Marlos C. Machado. Deep Laplacian-based Options for Temporally-Extended Exploration. In *International Conference on Machine Learning (ICML)*, 2023. URL https://proceedings.mlr.press/v202/klissarov23a.html.

Yehuda Koren. On Spectral Graph Drawing. In *International Computing and Combinatorics Conference (COCOON)*, 2003. URL https://doi.org/10.1007/3-540-45071-8_50.

Charline Le Lan, Stephen Tu, Adam Oberman, Rishabh Agarwal, and Marc G. Bellemare. On the Generalization of Representations in Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022. URL https://proceedings.mlr.press/v151/le-lan22a.html.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a Unified Theory of State Abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2006. URL http://anytime.cs.umass.edu/aimath06/proceedings/P21.pdf.

Marlos C. Machado. Efficient Exploration in Reinforcement Learning through Time-Based Representations. PhD Thesis, University of Alberta, 2019. URL https://era.library.ualberta.ca/items/581b87e0-a777-40a1-9776-f85a85864d6c.

Marlos C. Machado, Marc G. Bellemare, and Michael H. Bowling. A Laplacian Framework for Option Discovery in Reinforcement Learning. In *International*

Conference on Machine Learning (ICML), 2017. URL http://proceedings.mlr.press/v70/machado17a.html.

Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption Discovery through the Deep Successor Representation. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=Bk8ZcAxR-.

Marlos C. Machado, André Barreto, Doina Precup, and Michael Bowling. Temporal Abstraction in Reinforcement Learning with the Successor Representation. *Journal of Machine Learning Research*, 2023. URL http://jmlr.org/papers/v24/21-1213.html.

Sridhar Mahadevan. Proto-value Functions: Developmental Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2005. URL https://doi.org/10.1145/1102351.1102421.

Sridhar Mahadevan and Mauro Maggioni. Proto-value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes. *Journal of Machine Learning Research*, 2007. URL https://dl.acm.org/doi/10.5555/1314498.1314570.

Eric Mazumdar, Lillian J. Ratliff, and S. Shankar Sastry. On Gradient-Based Learning in Continuous Games. *SIAM Journal on Mathematics of Data Science*, 2020. URL https://doi.org/10.1137/18M1231298.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*, 2006.

David Pfau, Stig Petersen, Ashish Agarwal, David G. T. Barrett, and Kimberly L. Stachenfeld. Spectral Inference Networks: Unifying Deep and Spectral Learning. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=SJzqpj09YQ.

Mark B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, TX, USA, 1995. URL https://d-nb.info/945690320.

S. Shankar Sastry. *Nonlinear Systems: Analysis, Stability, and Control*, 2013.

Shagun Sodhani, Franziska Meier, Joelle Pineau, and Amy Zhang. Block Contextual MDPs for Continual Learning. In *Learning for Dynamics and Control Conference (L4DC)*, 2022. URL https://proceedings.mlr.press/v168/sodhani22a.html.

Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel Gershman. Design Principles of the Hippocampal Cognitive Map. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/dfd7468ac613286cdbb40872c8ef3b06-Abstract.html.

Richard S. Sutton. On the Significance of Markov Decision Processes. In *International Conference on Artificial Neural Networks (ICANN)*, 1997. URL https://doi.org/10.1007/BFb0020167.

Richard S. Sutton. Reinforcement Learning: Past, Present and Future. In *Asia-Pacific Conference on Simulated Evolution and Learning (SEAL)*, 1998. URL https://doi.org/10.1007/3-540-48873-1_26.

Richard S. Sutton. Toward a New Approach to Model-based Reinforcement Learning. [Unpublished manuscript], 2020. URL https://drive.google.com/drive/u/3/folders/1sCLuT2u82ifh2oaW9AWdPua6hXH69m4x.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*, 2018. URL http://incompleteideas.net/book/RLbook2020.pdf.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 1999. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Richard S. Sutton, Marlos C. Machado, G. Zacharias Holland, David Szepesvári, Finbarr Timbers, Brian Tanner, and Adam White. Reward-Respecting Subtasks for Model-Based Reinforcement Learning. *Artificial Intelligence*, 2023. URL https://doi.org/10.1016/j.artint.2023.104001.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*, 2010. URL https://doi.org/10.2200/S00268ED1V01Y201005AIM009.

Csaba Szepesvári. The Fundamental Theorem [Lecture Notes]. In Theoretical Foundations of Reinforcement Learning, 2023. URL https://rltheory.github.io/lecture-notes/planning-in-mdps/lec2/.

Thomas J. Walsh, Lihong Li, and Michael L. Littman. Transferring state abstractions between MDPs. In *International Conference on Machine Learning (ICML)*, 2006.

Han Wang, Archit Sakhadeo, Adam White, James Bell, Vincent Liu, Xutong Zhao, Puer Liu, Tadashi Kozuno, Alona Fyshe, and Martha White. No More Pesky Hyperparameters: Offline Hyperparameter Tuning for RL. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=AiOUi3440V.

Kaixin Wang, Kuangqi Zhou, Qixin Zhang, Jie Shao, Bryan Hooi, and Jiashi Feng. Towards Better Laplacian Representation in Reinforcement Learning with Generalized Graph Drawing. In *International Conference on Ma-*

chine Learning (ICML), 2021. URL http://proceedings.mlr.press/v139/wang21ae.html.

Kaixin Wang, Kuangqi Zhou, Jiashi Feng, Bryan Hooi, and Xinchao Wang. Reachability-Aware Laplacian Representation in Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2023. URL https://proceedings.mlr.press/v202/wang23av.html.

Martha White. Unifying task specification in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017. URL http://proceedings.mlr.press/v70/white17a.html.

Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning Representations with Efficient Approximations. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=HJlNpoA5YQ.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning Invariant Representations for Reinforcement Learning without Reconstruction. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

Appendix

6.1 Additional theoretical derivations

Proposition 1. *Let* $\mathbf{T} \in \mathbb{R}^{S\times S}$ *be a symmetric matrix,* $\mathbf{u}_1, \cdots, \mathbf{u}_S \in \mathbb{R}^S$ *and* $\lambda_1, \cdots, \lambda_S \in \mathbb{R}$ *be its eigenvectors and corresponding eigenvalues, and* $\mathbf{e}_1, \cdots, \mathbf{e}_d \in \mathbb{R}^S$ *be a* $d-$*dimensional orthonormal basis of the subspace* $\mathcal{E} := span((\mathbf{e}_i)_i)$*. Then, if* $\mathcal{E}$ *is closed under the operation of* $\mathbf{T}$*, i.e.,* $\mathbf{T}(\mathcal{E}) \subseteq \mathcal{E}$*, there must exist a* $d-$*dimensional subset of eigenvectors* $\{\mathbf{v}_1, \cdots, \mathbf{v}_d\} \subseteq \{\mathbf{u}_1, \cdots, \mathbf{u}_S\}$ *such that* $\mathcal{E}$ *coincides with their span, i.e.,* $\mathcal{E} = span((\mathbf{v}_i)_i) = span((\mathbf{e}_i)_i)$*.*

Proof. Let $\mathbf{w} \neq 0$ be a vector in $\mathcal{E}$. Then, by definition of $\mathcal{E}$, it can be expressed as a linear combination $\mathbf{w} = \sum_{i=1}^{d} \alpha_i \mathbf{e}_i$, where $\alpha_i \in \mathbb{R}$, and at least one of the coefficients is non-zero. Let us consider now the operation of $\mathbf{T}$ on $\mathbf{w}$ in terms of its eigenvectors. Specifically, we can express it as

$$\mathbf{T}\mathbf{w} = \sum_{j=1}^{S} \lambda_j \langle \mathbf{u}_j, \mathbf{w} \rangle \mathbf{u}_j \,,$$

which, by linearity of the inner-product, becomes

$$\mathbf{T}\mathbf{w} = \sum_{j=1}^{S} \left(\lambda_j \sum_{i=1}^{d} \alpha_i \langle \mathbf{u}_j, \mathbf{e}_i \rangle \right) \mathbf{u}_j \,.$$

Considering the hypothesis that $\mathcal{E}$ is closed under $\mathbf{T}$, we reach a necessary condition:

$$\sum_{j=1}^{S} \left(\lambda_j \sum_{i=1}^{d} \alpha_i \langle \mathbf{u}_j, \mathbf{e}_i \rangle \right) \mathbf{u}_j \stackrel{!}{=} \sum_{i=1}^{d} \beta_i \mathbf{e}_i \,, \tag{6.1}$$

where $\beta_i \in \mathbb{R}$, and at least one of them is non-zero.

We proceed by contradiction. Let us suppose that there does not exist a $d-$dimensional subset of eigenvectors $\{\mathbf{v}_1, \cdots, \mathbf{v}_d\}$ such that $\mathcal{E} = \text{span}((\mathbf{v}_i)_i)$. Since the eigenvectors form a basis of the whole space, we can express each $\mathbf{e}_i$ as linear combinations of the form

$$\mathbf{e}_i = \sum_{j=1}^{S} c_{ij} \mathbf{u}_i = \sum_{j=1}^{S} \langle \mathbf{u}_j, \mathbf{e}_i \rangle \mathbf{u}_j \,.$$

So, supposing that $\mathcal{E}$ does not correspond to any eigenvector subspace, there must exist $d' > d$ *different* indices $j_1, \cdots, j_{d'}$ and corresponding pairs (i_k, j_k) such that $c_{i_k j_k} \neq 0$. If this was not the case, this would imply that all the $\mathbf{e}_i$ lie in the span of some subset of d or fewer eigenvectors, and so $\mathcal{E}$ would correspond to this span.

Hence, we have that the coefficients α_i are arbitrary and that at least $d+1$ inner products are not zero. This implies that $\mathbf{w}$ lies in a subspace of dimension at least $d+1$ spanned by the d' eigenvectors $\mathbf{v}_1, \cdots, \mathbf{v}_{d'}$ with $\mathbf{v}_k = \mathbf{u}_{j_k}$. Now, the condition in Equation (6.1) requires this subspace to be the same as $\mathcal{E}$, but this is not possible since $\mathcal{E}$ is $d-$dimensional. Thus, we can conclude that there must exist a basis of d eigenvectors $\mathbf{v}_1, \cdots, \mathbf{v}_d$ of $\mathbf{T}$ such that $\mathcal{E} = \text{span}((\mathbf{v}_i)_i) = \text{span}((\mathbf{e}_i)_i)$. □

6.2 Average cosine similarity comparison

Env	ALLO	GGDO	t-statistic	p-value
GridMaze-17	0.9994 (0.0002)	0.9993 (0.0003)	0.641	0.262
GridMaze-19	0.9989 (0.0006)	0.9936 (0.0185)	2.218	0.015
GridMaze-26	**0.9984 (0.0007)**	0.9331 (0.0517)	9.770	0.000
GridMaze-32	**0.9908 (0.0161)**	0.8014 (0.0901)	16.018	0.000
GridMaze-7	**0.9996 (0.0002)**	0.2959 (0.0159)	343.724	0.000
GridMaze-9	**0.9989 (0.0007)**	0.3755 (0.0081)	596.775	0.000
GridRoom-1	**0.9912 (0.0003)**	0.9906 (0.0003)	9.691	0.000
GridRoom-16	**0.9990 (0.0004)**	0.9980 (0.0023)	3.297	0.001
GridRoom-32	**0.9982 (0.0010)**	0.9857 (0.0266)	3.647	0.000
GridRoom-4	**0.9965 (0.0052)**	0.9073 (0.0063)	84.136	0.000
GridRoom-64	**0.9917 (0.0059)**	0.7617 (0.0834)	21.326	0.000
GridRoomSym-4	0.8411 (0.0742)	0.8326 (0.0855)	0.581	0.281

Table 6.1: Average cosine similarities for ALLO and GGDO when using the **(x, y) state representation**. The sample average is calculated using 60 seeds. The standard deviation is shown in parenthesis and the maximal average is shown in boldface when the difference is statistically significant, meaning the associated p-value is smaller than 0.01.

Env	ALLO	GGDO	t-statistic	p-value
GridRoom-16	**0.9992 (0.0003)**	0.7836 (0.3897)	4.251	0.000
GridRoom-1	**0.9914 (0.0002)**	0.7114 (0.4199)	5.080	0.000
GridMaze-9	**0.9988 (0.0013)**	0.3773 (0.0064)	720.882	0.000
GridMaze-19	0.9789 (0.1354)	0.9480 (0.2032)	0.926	0.178
GridRoom-32	**0.9073 (0.2591)**	0.3795 (0.4183)	7.917	0.000
GridMaze-26	0.9330 (0.2214)	0.8466 (0.2358)	1.935	0.028
GridMaze-17	0.9995 (0.0001)	0.9687 (0.1658)	1.429	0.079
GridMaze-32	0.6198 (0.3056)	0.5810 (0.3745)	0.598	0.276
GridRoomSym-4	0.8434 (0.0743)	0.8407 (0.0686)	0.197	0.422
GridMaze-7	**0.9997 (0.0002)**	0.3015 (0.0211)	253.863	0.000
GridRoom-4	**0.9965 (0.0076)**	0.8950 (0.1065)	7.297	0.000
GridMaze-11	**0.9349 (0.0496)**	0.3975 (0.0592)	52.931	0.000

Table 6.2: Average cosine similarities for ALLO and GGDO when using the **pixel state representation**. The sample average is calculated using 60 seeds. The standard deviation is shown in parenthesis and the maximal average is shown in boldface when the difference is statistically significant, meaning the associated p-value is smaller than 0.01.

6.3 Average eigenvalues

Env	True	ALLO	GGDO
GridRoomSym-4	0.0000	0.0000 (0.0000)	0.0000 (0.0000)
	0.0540	0.0429 (0.0003)	0.0395 (0.0003)
	0.0540	0.0434 (0.0003)	0.0387 (0.0003)
	0.1068	0.0863 (0.0005)	0.0703 (0.0005)
	0.4502	0.4034 (0.0007)	0.0531 (0.0007)
	0.4507	0.4045 (0.0007)	0.0529 (0.0007)
	0.4507	0.4054 (0.0007)	0.0524 (0.0007)
	0.4512	0.4066 (0.0008)	0.0520 (0.0008)
	0.4915	0.4479 (0.0008)	0.0156 (0.0008)
	0.4951	0.4513 (0.0007)	0.0121 (0.0007)
	0.4951	0.4526 (0.0008)	0.0108 (0.0008)
GridRoom-16	0.0000	0.0000 (0.0000)	0.0000 (0.0000)
	0.0016	0.0017 (0.0000)	0.0016 (0.0000)
	0.0063	0.0055 (0.0000)	0.0051 (0.0000)
	0.0139	0.0116 (0.0001)	0.0111 (0.0001)
	0.0242	0.0199 (0.0002)	0.0189 (0.0002)
	0.0367	0.0301 (0.0002)	0.0279 (0.0002)
	0.0511	0.0420 (0.0002)	0.0382 (0.0002)
	0.0663	0.0546 (0.0003)	0.0482 (0.0003)
	0.0832	0.0688 (0.0005)	0.0591 (0.0005)
	0.1007	0.0836 (0.0006)	0.0691 (0.0006)
	0.1161	0.0969 (0.0005)	0.0774 (0.0005)
GridMaze-9	0.0000	0.0001 (0.0000)	0.0000 (0.0000)
	0.1582	0.1154 (0.0006)	0.0962 (0.0006)
	0.3083	0.2354 (0.0011)	0.1159 (0.0011)
	0.4899	0.3961 (0.0015)	0.0181 (0.0015)
	0.6613	0.5674 (0.0020)	0.0000 (0.0020)
	0.7529	0.6692 (0.0024)	0.0000 (0.0024)
	0.7777	0.6986 (0.0023)	0.0000 (0.0023)
	0.8266	0.7585 (0.0027)	0.0000 (0.0027)
	0.8613	0.8038 (0.0028)	0.0000 (0.0028)
	0.8768	0.8251 (0.0029)	0.0000 (0.0029)
	0.8796	0.8292 (0.0029)	0.0000 (0.0029)
GridMaze-7	0.0000	0.0001 (0.0000)	0.0000 (0.0000)
	0.1833	0.1325 (0.0006)	0.1034 (0.0006)
	0.4622	0.3645 (0.0011)	0.0437 (0.0011)
	0.5208	0.4186 (0.0010)	0.0000 (0.0010)
	0.7148	0.6158 (0.0012)	0.0000 (0.0012)
	0.7958	0.7084 (0.0014)	0.0000 (0.0014)
	0.8392	0.7621 (0.0014)	0.0000 (0.0014)
	0.8549	0.7826 (0.0017)	0.0000 (0.0017)
	0.8739	0.8076 (0.0015)	0.0000 (0.0015)
	0.8934	0.8347 (0.0016)	0.0000 (0.0016)
	0.9091	0.8572 (0.0018)	0.0000 (0.0018)
GridRoom-32	0.0000	0.0000 (0.0000)	0.0000 (0.0000)
	0.0008	0.0010 (0.0000)	0.0008 (0.0000)
	0.0018	0.0019 (0.0000)	0.0017 (0.0000)
	0.0039	0.0036 (0.0000)	0.0031 (0.0000)
	0.0065	0.0057 (0.0001)	0.0053 (0.0001)
	0.0135	0.0114 (0.0001)	0.0105 (0.0001)
	0.0161	0.0135 (0.0001)	0.0124 (0.0001)
	0.0200	0.0167 (0.0001)	0.0151 (0.0001)
	0.0270	0.0223 (0.0002)	0.0206 (0.0002)
	0.0284	0.0236 (0.0002)	0.0212 (0.0002)
	0.0364	0.0301 (0.0002)	0.0265 (0.0002)
GridRoom-64	0.0000	0.0000 (0.0000)	0.0001 (0.0000)
	0.0004	0.0007 (0.0000)	0.0006 (0.0000)
	0.0010	0.0012 (0.0000)	0.0009 (0.0000)
	0.0016	0.0017 (0.0000)	0.0017 (0.0000)
	0.0020	0.0021 (0.0000)	0.0015 (0.0000)
	0.0021	0.0022 (0.0000)	0.0018 (0.0000)
	0.0035	0.0033 (0.0000)	0.0031 (0.0000)
	0.0041	0.0038 (0.0000)	0.0034 (0.0000)
	0.0063	0.0055 (0.0000)	0.0051 (0.0000)
	0.0090	0.0078 (0.0001)	0.0072 (0.0001)
	0.0097	0.0084 (0.0001)	0.0078 (0.0001)

Env	True	ALLO	GGDO
GridMaze-32	0.0000	0.0000 (0.0000)	0.0001 (0.0000)
	0.0000	0.0007 (0.0001)	0.0004 (0.0001)
	0.0004	0.0008 (0.0000)	0.0006 (0.0000)
	0.0013	0.0015 (0.0000)	0.0010 (0.0000)
	0.0037	0.0035 (0.0000)	0.0033 (0.0000)
	0.0043	0.0040 (0.0000)	0.0038 (0.0000)
	0.0058	0.0053 (0.0001)	0.0050 (0.0001)
	0.0063	0.0056 (0.0001)	0.0053 (0.0001)
	0.0075	0.0067 (0.0001)	0.0059 (0.0001)
	0.0099	0.0085 (0.0001)	0.0080 (0.0001)
	0.0148	0.0126 (0.0001)	0.0118 (0.0001)
GridMaze-26	0.0000	0.0000 (0.0000)	0.0001 (0.0000)
	0.0005	0.0008 (0.0000)	0.0007 (0.0000)
	0.0034	0.0032 (0.0000)	0.0030 (0.0000)
	0.0039	0.0037 (0.0001)	0.0035 (0.0001)
	0.0048	0.0044 (0.0001)	0.0039 (0.0001)
	0.0058	0.0051 (0.0001)	0.0048 (0.0001)
	0.0094	0.0080 (0.0001)	0.0078 (0.0001)
	0.0129	0.0109 (0.0001)	0.0105 (0.0001)
	0.0156	0.0131 (0.0001)	0.0130 (0.0001)
	0.0195	0.0164 (0.0002)	0.0154 (0.0002)
	0.0255	0.0213 (0.0002)	0.0201 (0.0002)
GridMaze-17	0.0000	0.0000 (0.0000)	0.0000 (0.0000)
	0.0029	0.0027 (0.0000)	0.0026 (0.0000)
	0.0116	0.0097 (0.0001)	0.0094 (0.0001)
	0.0257	0.0212 (0.0001)	0.0199 (0.0001)
	0.0448	0.0368 (0.0002)	0.0331 (0.0002)
	0.0682	0.0562 (0.0003)	0.0490 (0.0003)
	0.0952	0.0790 (0.0004)	0.0647 (0.0004)
	0.1251	0.1045 (0.0005)	0.0809 (0.0005)
	0.1572	0.1327 (0.0005)	0.0950 (0.0005)
	0.1907	0.1624 (0.0006)	0.1065 (0.0006)
	0.2249	0.1937 (0.0007)	0.1163 (0.0007)
GridMaze-19	0.0000	0.0000 (0.0000)	0.0000 (0.0000)
	0.0016	0.0016 (0.0000)	0.0014 (0.0000)
	0.0058	0.0051 (0.0000)	0.0048 (0.0000)
	0.0068	0.0059 (0.0001)	0.0056 (0.0001)
	0.0140	0.0117 (0.0001)	0.0111 (0.0001)
	0.0232	0.0192 (0.0002)	0.0177 (0.0002)
	0.0365	0.0300 (0.0002)	0.0283 (0.0002)
	0.0403	0.0332 (0.0003)	0.0306 (0.0003)
	0.0516	0.0425 (0.0004)	0.0390 (0.0004)
	0.0559	0.0461 (0.0003)	0.0416 (0.0003)
	0.0821	0.0679 (0.0004)	0.0567 (0.0004)
GridRoom-4	0.0000	0.0000 (0.0000)	0.0000 (0.0000)
	0.0490	0.0392 (0.0003)	0.0357 (0.0003)
	0.0576	0.0462 (0.0003)	0.0416 (0.0003)
	0.1122	0.0908 (0.0005)	0.0732 (0.0005)
	0.3905	0.3448 (0.0014)	0.0923 (0.0014)
	0.4420	0.3963 (0.0009)	0.0592 (0.0009)
	0.4531	0.4080 (0.0010)	0.0517 (0.0010)
	0.4585	0.4139 (0.0010)	0.0451 (0.0010)
	0.4787	0.4348 (0.0008)	0.0283 (0.0008)
	0.4917	0.4489 (0.0010)	0.0152 (0.0010)
	0.5209	0.4802 (0.0008)	0.0000 (0.0008)
GridRoom-1	0.0000	0.0000 (0.0000)	0.0000 (0.0000)
	0.0895	0.0728 (0.0003)	0.0610 (0.0003)
	0.0895	0.0735 (0.0004)	0.0629 (0.0004)
	0.1643	0.1372 (0.0005)	0.0986 (0.0005)
	0.2801	0.2412 (0.0006)	0.1175 (0.0006)
	0.2801	0.2425 (0.0005)	0.1183 (0.0005)
	0.3277	0.2868 (0.0007)	0.1128 (0.0007)
	0.3277	0.2884 (0.0007)	0.1134 (0.0007)
	0.4376	0.3980 (0.0009)	0.0622 (0.0009)
	0.4622	0.4229 (0.0008)	0.0432 (0.0008)
	0.4622	0.4244 (0.0008)	0.0419 (0.0008)

Table 6.3: Average eigenvalues for ALLO and GGDO. For each environment, the true eigenvalues are shown in decreasing order, from the 2nd one up to the 11th. The sample averages are calculated using 60 seeds and in parenthesis are the standard deviations.