

https://dspace.org/

Spectral reinforcement learning

Earle, Adam C.

2019-03
https://hdl.handle.net/10539/29594

Downloaded from a DSpace repository. https://dspace.org/

Spectral Reinforcement Learning

A THESIS PRESENTED
BY
ADAM C. EARLE
TO
THE SCHOOL OF COMPUTER SCIENCE AND APPLIED MATHEMATICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
APPLIED MATHEMATICS

UNIVERSITY OF THE WITWATERSRAND
JOHANNESBURG, SOUTH AFRICA
MARCH 2019

©2019 – Adam C. Earle
all rights reserved.

Thesis advisors: Dr Benjamin Rosman, Professor Montaz M. Ali,
Dr Andrew Saxe Adam C. Earle

Spectral Reinforcement Learning

Abstract

The widespread success of deep learning techniques in recent years has been nothing short of remarkable. These techniques have revolutionized many diverse fields, such as machine vision and natural language processing; driving breakthrough technologies such as driverless cars and language translation. Despite the broad applicability of deep learning methods, successes have been largely confined to perceptual domains. The application of deep learning techniques to control problems on the other hand, has been met with temperate success. Although deep reinforcement learning (RL) methods have certainly yielded a number of impressive results in specific domains such as chess, go, and StarCraft, these successes have been equally revealing of the limitations of these approaches. Current state of the art methods are known to be extremely sample inefficient, often requiring millions of times more samples than their human counterparts to solve these problems, and are often limited to solving one task at a time.

At the heart of the success of deep learning techniques is the recursive composition of basic features in the construction of ever more abstract representations. Yet this key insight is utilized only in a limited way in current RL methods; typically in the approximation of an auxiliary value function. Crucially, the action selection protocol for these methods remains ultimately flat. A richer manifestation of the key insights provided by deep learning, would see the agent invoke a hierarchical action

selection protocol in which primitive actions are recursively composed to yield complex behavioural abstractions.

In this thesis I propose a new RL method wherein the agent executes a recursive compositional action selection protocol. This is achieved through the construction of a hierarchical control architecture with a bidirectional execution paradigm in which higher level behaviours are derived from the recursively composition of lower level controllers, and inversely primitive actions arise from the concurrent execution of multiple high-level controllers. In addition, I develop a fully autonomous procedure for uncovering the hierarchical architecture directly from the agent's experience in the domain. The novel control architecture allows the agent to uncover and exploit the multiscale structure of real world tasks, demonstrating powerful multitasking capabilities.

The theory is developed in four phases. Firstly I present a formalism of multitask learning with compositional policies in the offline setting; highlighting the powerful transfer capabilities of the agent. Secondly, this methodology is extended to the online setting through the construction of an integrated learning problem in which the agent periodically adjusts the relative influence of its constituent policies in response to new experiences. The extension to the online setting requires a mapping between the base layer states and the constituent policies. Thirdly, I propose an autonomous procedure by which this mapping may be uncovered. This sets the stage for the recursive application of the abstraction procedure. Finally I define the recursive procedure by which the deep hierarchical control architecture may be constructed, and the corresponding execution model through which it is ultimately actioned.

Contents

1 Introduction 1
- 1.1 Reinforcement Learning and the Multitask Learning Paradigm . . . 1
- 1.2 Transfer Learning and Hierarchy . . . 6
- 1.3 Compositionality and Task Synthesis . . . 11
- 1.4 Overview of Contributions . . . 14

2 Preliminaries 17
- 2.1 Introduction . . . 17
- 2.2 Reinforcement Learning . . . 18
- 2.3 Linear Markov Decision Processes . . . 35

3 Multitask Reinforcement Learning with Compositional Policies 44
- 3.1 Introduction . . . 44
- 3.2 The Importance of Compositionality and Concurrency . . . 48
- 3.3 Inexact Composition and Optimal Subspaces . . . 63
- 3.4 Semantics and Limitations of Task Composition . . . 83

4 Hierarchical Abstraction: A Single Layer Hierarchy 87
- 4.1 Introduction . . . 87
- 4.2 Constructing an Integrated Learning Problem . . . 92
- 4.3 The Execution Model . . . 100
- 4.4 Learning Dynamics in the Hierarchy . . . 107

5 Autonomous Skills Discovery 113
- 5.1 Introduction . . . 113
- 5.2 Subtask Discovery with NNMF . . . 115
- 5.3 A Note on Subtask Equivalence . . . 122
- 5.4 Autonomous Construction of the Subtask Transition Structure . . . 125

6 Discovering and Using Deep Control Hierarchies 128
- 6.1 Introduction . . . 128
- 6.2 The Benefits of Depth . . . 131
- 6.3 Recursive Construction of the Control Hierarchy . . . 134
- 6.4 Deep Skills Discovery . . . 135
- 6.5 Learning with Deep Hierarchies . . . 141

7 Conclusion 146
7.1 Summary of Contributions . . . 147
7.2 The Applicability of Linear Models . . . 150
7.3 Possible Extensions . . . 152
7.4 A Final Word . . . 156

Appendix A 158
A.1 The LMDP Setup for Experimental Domains . . . 158
A.2 Initializations, Learning Rates Schedules, and Moving Averages . . . 160
A.3 Exact Multitasking in the Offline Setting . . . 162
A.4 Linear Convergence . . . 163
A.5 Multitasking in the Online Setting . . . 163
A.6 Multitasking with Deep Control Hierarchies . . . 165

References 177

I declare that this thesis is my own, unaided work. It is being submitted for the Degree of Doctor of Philosophy at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

Adam Christopher Earle
18 March 2019

1

Introduction

1.1 Reinforcement Learning and the Multitask Learning Paradigm

Reinforcement learning has a rich history in the study of trial-and-error learning, both in the fields of human and animal psychology, and in the fields of artificial intelligence and control theory. At the turn of the 20^{th} century, reinforcement learning was introduced to the field of psychology as a descriptive mechanism to explain behavioural changes brought about through a series of interac-

tions between an agent, and its environment. The resulting paradigm conceptualizes learning as a series of incremental, speculative, changes to behaviour in response to the environmental feedback. This served well as a descriptive tool for the classical behavioural conditioning famously described by Pavlov and Watson[1,2]; and the subsequent theories of operant conditioning expounded by Skinner and others[3]. A central tenet of the resulting learning paradigm is that when faced with a range of choices in similar situations, an agent, equipped with the necessary faculties, will choose more often the actions that were associated with a desirable outcome, than those associated with a poor outcome. As a true description of the mechanism underlying animal behaviour, the theory remains somewhat contentious[4,5]; nevertheless, the basic principle is widely regarded as an obvious driver underlying much of animal behaviour[6].

Even if reinforcement learning is in fact a suitable model for the description of some subsection of animal behaviour, it nevertheless leaves many questions unanswered. For instance, it remains unclear under what conditions an agent should consider two situations, necessarily distinct in time, to be *similar*. Similarly, the mechanism of reward attribution, the process of establishing a causal link between an outcome and a behaviour, is also somewhat opaque since a desired outcome is not simply attributable to the immediately preceding action, but rather the complex summation of all preceding actions. Many of these questions remain open, and are present in the technical application of the learning paradigm as well.

The underlying concepts of reinforcement learning were later refurbished and repurposed as a mathematical tool in control theory for directly performing adaptive control[7]. These advancements were spurred on by myriad applications of control theory in engineering. This technical setting

presents a more simple and structured backdrop against which some of the difficulties expressed above vanish. In the abstract mathematical setting for instance, two situations may be identified as being similar by explicitly comparing the full parameter space of the control system. Nevertheless, many issues persist. For example, the problem of reward attribution is in no way diminished, since a desired outcome is still the result of a complex summation of all preceding actions. Around the same time, trial-and-error learning systems were being described by researchers working on artificial intelligence. They too were grappling with many of the same issues which proved a boon to the field, with advancements brought about precisely because of the parallel development enacted across these diverse fields.

With the advent of computational technologies, machine learning has emerged as a preeminent branch of artificial intelligence, and reinforcement learning as a core component thereof. In contrast with the other major branches of machine learning, namely supervised and unsupervised learning, reinforcement learning sits 'atop the cake'[8] as describing a class of algorithms that are capable of solving complex sequential decision problems without the need for huge amounts of labelled training data. To date, reinforcement learning algorithms have successfully been applied to problems in robotic control[9,10,11,12]; automated stock trading[13,14]; and more recently in the playing of perfect information games, such as backgammon[15], chess[16] and go[17], and imperfect information games such as StarCraft 2[18,19].

While these are certainly impressive feats, current state of the art reinforcement learning methods are not without their short-comings. Foremost amongst these is the notorious sample inefficiency of current methods. Although they require only a single scalar reward as a learning signal, current al-

gorithms often require many millions of trials in order to solve complex tasks. In practice it is often simply not possible to run this many trials, and as such the scope of applicability of current methods has been largely limited to applications which can be simulated in silica. Simulation provides the means to generate many trials in a reproducible, and massively parallel, fashion such that our algorithms are able to converge in a reasonable amount of wall-clock time despite their sample inefficiency.

The sample inefficient nature of our current algorithms is such that they often require many orders of magnitude more samples than humans do to solve the same problem. In the popular Atari games environment[20], in which super-human performance has been achieved on many games[21], state-of-the-art algorithms still require many orders of magnitude more trials than even amateur human gamers. When samples are cheap and fast to generate, the data hungry nature of our algorithms may not be fundamentally limiting in engineering applications. However, for many domains of interest samples are expensive and time consuming to acquire. These domains demand greater sample efficiency of our methods. Moreover, this dependence on massive amounts of data would seem to be a critical barrier to any form of general-purpose AI, since an agent limited in this way would not be able to function in the real-world.

Contrasted with machine learning models, human learning is distinguished both by its richness and sample efficiency. Children are often able to classify a new class after the presentation of just a single sample[22]. The ability to perform this sort of 'one-shot' learning is an emergent property of human learning which persists into adulthood and is a striking hallmark of organic intelligence which is not generally exhibited by our algorithms (although recent work has explored few-shot learning

in specific domains[23,24,25]). Furthermore, humans are able to demonstrate this rapid learning over a wide range of tasks; even those seemingly unlike anything they have experienced before. It is unclear how we are able to perform this seemingly miraculous feat of learning given the paucity of the data. As we ponder the root cause of this discrepancy in learning efficiency, it bears mention that the human example is fundamentally situated in time. That is to say that, unlike an algorithm, a person is never truly able to start with a blank slate when presented with a new task. Instead, they bring with them the summation and synthesis of all their previous life experience, and indeed, bring this experience to bear on the new task at hand in a structured fashion.

Returning to the Atari domain, even though the human players have never played the specific game before, they have access to many of the critical 'modules' - they are able to recognize objects; they have some notion of intuitive physics and game dynamics; and they are able to assign credit to their actions in a temporally extended manner by virtue of an internal world-model[26]. Details aside, it would certainly seem to be the case that humans are somehow able to leverage diverse previous experiences, in a structured way, to dramatically improve their initial performance on new tasks. In addition, people are able to do this over a wide range of domains, and are able to execute these mechanics very rapidly.

It seems to be the case that seemingly disparate real-world tasks in fact share sufficient structure, such that the experience gained in learning to solve some tasks, may be leveraged to improve learning on others. This observation suggests that there may be value in studying trial-and-error learning in the context of solving a set of tasks, rather than a single task. In the machine learning parlance, this is know as the multitask setting.

1.2 Transfer Learning and Hierarchy

In the multitask setting the agent is faced with a whole set of tasks, rather than just a single task. This particular problem setting is well suited to the study of how knowledge should be represented within the agent, and the mechanisms by which it is retrieved and repurposed to aid learning on a new task. A subtle but important distinction is sometimes made between two common realizations of the multitask learning paradigm. In the first, the agent is presented with the full set of tasks to be solved from the beginning. Learning may then proceed in a parallel fashion on all of the constituent tasks at once. This realization is commonly referred to as *multitask learning*. In the second, the agent is instead presented with the tasks in a serial fashion with a new task being drawn from the full set of tasks only once the current task has been solved. In this setup, learning proceeds in a strictly incremental fashion, and is thus referred to as *incremental learning* or continual learning.

Both multitask learning and incremental learning refer to a particular context in which the process of learning may be analyzed. In both cases, the study the underlying mechanisms by which knowledge is stored, retrieved, and repurposed is referred to as the study of *transfer learning*.

It is of contextual interest that the study of transfer learning exists also in the realm of supervised learning. While there is certainly overlap in the research being conducted in these sibling fields, there are also some fundamental differences. An important distinction is to be made when neural architectures are the apparatus of choice. Neural networks traditionally store their knowledge in the weight space of the network. As such, incremental learning is synonymous with the study of weight dynamics in these networks. Researchers seek to understand how the network parameters should be

changed so as to solve a new task, understanding that as these parameters change knowledge of previous tasks may be degraded or destroyed[27,28]. The tendency of neural architectures to abruptly forget previously acquired knowledge when new information is received is known as catastrophic interference or catastrophic forgetting. On the other hand, in the reinforcement learning setting, one may be more permissive in how knowledge is represented within the agent, at times leveraging an explicit external memory bank from which information can be written and retrieved. In practice these distinctions are not quite so sharp, with reinforcement learning researchers investigating single network architectures capable of learning multiple tasks[29], while supervised learning researchers explore the possibility of external memory mechanisms.

Even when restricted to the reinforcement learning domain, transfer learning is nevertheless an incredibly broad field. To acquire a sense for this breadth, it is illustrative to view the disparate branches of transfer learning in the reinforcement learning context through a mathematical lens. Reinforcement learning problems are typically modeled as Markov Decision Processes (MDPs). An MDP may be represented as a 4-tuple $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$. Here $\mathcal{S}$ denotes a set of states in which the agent may reside, $\mathcal{A}$ denotes a set of actions which the agent may take, P denotes the probability of transitioning to a new state given the current state and the chosen action, and finally R represents a scalar reward for having executed the transition. The agent is then tasked with discovering which actions to take in which states in order to maximize some notion of cumulative reward. This is discussed in greater detail in Chapter.(2).

Notice that changing any one of the defining parameters yields a new MDP and consequently a new task. Consider a navigation task in which a robot is set to finding some special location in

the office: a charging station perhaps. It is easy to see how such a problem may be modeled as an MDP. Now suppose that we would like our robot to find a different charging station in the same office. Given that the office layout has not changed, this new task may be modeled as an MDP with all the same parameters, but with a different reward structure. Alternatively, we might want our agent to once again find the initial charging station, but forbid it from accessing the east corridor. This new task may be modeled as an MDP with the same parameters, but with a different transition structure. As a principled way of building up to a general theory of transfer learning in this context, one might sensibly begin by studying transfer between tasks which differ along just one of the parametric dimensions. Thereafter, one might study transfer between tasks which differ along multiple dimensions, or indeed between all dimensions, resulting in a general theory of transfer between MDPs. Historically, the goal of transfer learning has been more modest and pragmatic. Rather than attempting to formalize a theory of transfer between arbitrary MDPs, tasks are almost always assumed to have some shared structure.

The shared structure between tasks is often laid bare when the correct representation is uncovered. It follows from the compositional and multiscale nature of real-world tasks, that suitable representations of this sort are often hierarchical in nature. Returning to our example of a robot performing a navigation task in an office, suppose that the navigation task could be represented by specifying three distinct components: the floor of the building, the location of the office on the floor, and finally the wall in the office on which the charging station is mounted. The agent would then solve the task by executing the constituent policies, first navigating to the correct floor, then the correct office, and finally the correct wall. Of course, the second task, in which the agent seeks

an alternate charging station, also permits a description in terms of the three distinct components. Importantly, when some of these components are shared, the agent need not relearn them. In this way, knowledge gained in solving the first task is transferred to, and improves performance on, the second.

This observation suggests a rather general mechanism for transferring knowledge between tasks. First, the agent must find a suitable hierarchical representation over the tasks which have already been solved. This representation may then be used to improve learning on subsequent tasks. Intuitively, this approach seeks to distill the latent structure in the task ensemble which arises from regularities in the transition and reward structures. Indeed, control engineers have constructed these sorts of hierarchical architectures in control systems for decades. Unfortunately, from the machine learning perspective, this simply passes the proverbial buck down the line.

The problem of developing learning algorithms capable of transferring knowledge has been replaced with the problem of developing algorithms capable of extracting the latent structure between tasks. Interestingly, attempting to extract this shared structure between a set of tasks is a problem that has been tackled by the reinforcement learning community before; albeit motivated by slightly different goals. Research in this direction has been motivated by the simple observation that planning is made simpler in the presence of temporally extended behaviours[30]. By way of example, a human tasked with getting to the post office is likely to execute a plan involving a sequence of actions such as: *get in the car*, *drive to main street parking*, and *walk to the corner*. Critically, while each of these 'actions' themselves comprise many motor primitives, planning is simply not carried out at that fine-grained scale. Similarly, a robot tasked with getting to the post office is better able

to solve that task when planning is not carried out exclusively over primitive actions. Attempting to autonomously extract suitable temporally extended actions from a set of trajectories is a problem addressed in the literature under the banner of *skills discovery*[31,32,33,34]. In general, skills discovery is a notoriously difficult problem. One fruitful approach towards uncovering these skills, is precisely by analyzing the latent structure of a set of tasks, and constructing skills that capture the shared behaviour.

In this thesis we develop a method for transfer learning which incorporates many of these ideas. Firstly, we exploit ideas from transfer learning to develop an agent equipped with an external memory bank from which previous experience can be retrieved and synthesised as needed. Moreover, we show how new experiences can be written to the external memory in such a way as to avert catastrophic forgetting. Next, we show how the knowledge in the external memory might be represented. By leveraging ideas in the skills discovery literature, we extract a set of prototypical 'basis' tasks from a set of trajectories which together cover the full range of behaviours exhibited by the task ensemble. We show that these basis tasks constitute a compressed representation of the agent's experience to date, and are in fact themselves a good candidate for the knowledge representation. We are then in a position to extract further structure from the flat knowledge base, by imposing a hierarchical representation. This representation further improves the agent's ability to transfer knowledge between tasks by permitting transfer in a multiscale fashion.

Unfortunately, we remain plagued by the curse of dimensionality. It is a well-known fact that, in discrete state and action spaces, the number of parameters that an agent must learn increases exponentially as the size of the state and action space grows[6] (sans some generalization). It is an un-

fortunately reality that this scaling demon affects not just the number of base parameters that the agent must learn, but also the number of skills that must be extracted. Given that our agent's external memory is ultimately populated by a set of skills, this immediately affects the algorithm's ability to scale, since the size of the required memory quickly becomes unwieldy.

1.3 Compositionality and Task Synthesis

In the face of daunting scaling concerns, we consider again the human example. Assuming that MDPs are a suitable framework for modeling real-world tasks in abstraction, what might be the missing ingredient that allows human learning to scale in a way that seems impossible to achieve with our current algorithms?

Careful consideration of the mechanisms underpinning human learning, have inspired a number of conceptual ideals that our learning algorithms might benefit from. Indeed there is a long history of fruitful interplay between models of human cognition and artificial intelligence - this is evidenced in the central concept of 'mind as machine' in the cognitive sciences[35,36], and similarly in the wide-spread use of terminology such as 'memory', 'reinforcement learning', and 'perception' in the machine learning communities. While human intelligence is most certainly realized as a cacophony of many concurrent factors, we argue that one of these ideals, namely the ability to reuse a number of elementary building blocks to construct complex new behaviours, is especially important for tackling sample inefficiency in learning. The ability to combine existing simple behaviours to build complex new behaviours is referred to as compositionality, and is a powerful approach to

combat the curse-of-dimensionality.

A similar form of composition is thought to underpin large portions of human motor control[37,38,39,40]. When faced with a new control task, a person is able to draw on their previous experience to speed up learning. Seen through the appropriate lens, signatures of composition are present: people are almost always able to perform significantly better than random, even in the first few trials of a new task; two stages of learning are present (rapid initial improvement followed by incremental refinement); and the duration of the first stage of learning is roughly in proportion to how much their experience relates to the task at hand. These phenomena can be seen to emerge from a two-phase learning procedure that involves first learning to map the new task into a set of previous learnt skills, and then iteratively fine-tuning performance through subsequent trial-and-error experience.

By way of example, consider a person learning to hit a golf ball for the first time. While the specific motor control policy for the task must be fine-tuned, there are nevertheless a set of existing primitive control policies for standing, swinging one's arms, grasping, and bending, that can be leveraged to achieve a strong baseline performance. Thereafter, improvements can be made through continued practice and learning, but performance gains are significantly slower[38]. The phenomenological experience is one of all the essential components being present, and composed; but the resulting compositional policy being suboptimal for the specific task at hand. A croquet player on the other hand may have at their disposal an additional control policy which is better suited for the task of swinging a golf club. Such a person would likely experience a jump-start effect in their learning, beginning from a better baseline after which learning is again slower.

Absent a formal theory of both composition and transfer, our algorithms remain critically ex-

posed to the curse of dimensionality. A robotic arm, for example, must learn to navigate in the product space of its joints, rather than learning to manipulate each join independently and then combining the resulting policies. Notice that transfer learning alone may not be enough to overcome this problem. While successful transfer may allow the agent to learn to manipulate individual joints more rapidly, this does not correspond to multiple joints acting concurrently in an integrated fashion. To overcome the curse of dimensionality the agent would need to be able to execute a compositional policy in which multiple constituent policies are executed in parallel and in a graded fashion. In addition, the compositional approach to learning naturally suggests the extension to a hierarchical architecture, in which composite behaviours are themselves constituents for yet more abstract behaviours.

It would seem to be the case that both composition and hierarchical structuring are requisite ingredients. In so much as the human example is to be followed, strong parallels may be drawn between the idealized compositional hierarchies we have described and the many perceptual hierarchies at work in human cognition [41,42,43,44]. Although we will not require any measure of biological plausibility of our algorithms, it certainly seems likely that some form of structured composition must emerge as a cornerstone of any computational approach which achieves human-level sample efficiency in learning.

Compositionality is an important intuition in our understanding of perceptual systems, and we have made a strong case for the value of incorporating compositionality into the building of our control systems as well. Nevertheless this is typically not a property exhibited by current reinforcement learning algorithms. This notable absence would appear to be due, ultimately, to the inherently

unstructured nature of the MDP formalism. In the general case, an MDP satisfies a non-linear optimality condition, which appears to be diametrically opposed to the sort of convex composition we have described.

1.4 Overview of Contributions

Adopting the view that compositionality is a fundamental component of any scalable learning algorithm, we adopt a simplified framework, the linearized MDP [45], in which a limited form of optimal composition is known to hold. It is within this particular framework that we develop our method for transfer learning. Explicitly, we develop an offline method for transfer learning by extending the linearized MDP framework to the multitask setting. In this setting, the agent is equipped with an external memory bank which contains a set of prototypical tasks and their corresponding optimal policies. By exploiting the inherent compositionality of the framework, we show that the agent may immediately perform graded blends of previously learnt tasks and jump-start learning on novel tasks by initializing its policy as an approximate blend of the policies for the prototypical tasks.

Next we recast a number of difficult questions in transfer learning as a set of problems in linear algebra for which solution methods exist and are well understood. Critically, we establish that the convexity of the objective function in the simplified framework implies that a linear relationship exists between the initialization error for the optimal policy, and the rate at which the agent is able to learn the new task. This relation implies that given a suitable set of basis tasks, the agent is immediately able to perform optimally on any possibly task in the domain, by suitably initializing its policy.

Moreover, we conclude that the value of any particular set of subtasks may be explicitly quantified by measuring the approximation error to the complete basis. As a corollary of this result, we infer a method for choosing an optimal subset of tasks by minimizes the approximation error to the set of basis tasks. In this way the question of the quality of transfer achievable with a given set of subtasks is recast as a question about matrix reconstruction error.

The new method is then extended to the online setting, in which the reward function for the new task is not provided up front. This extension is achieved by constructing an integrated learning problem in which the agent periodically queries the external memory bank as it obtains more experience in the domain. We find that the integrated problem both improves early exploration over the flat implementation, and allows the agent to rapidly hone in on goal states as they are encountered. The integrated learning problem is shown to constitute a linear transformation of the base problem to one which is amenable to expedient numerical solution [*].

Further leveraging the linear relationship between initialization error and learning rate, we propose a novel method for skills discovery in the linearized multitask framework based on finding a low-rank approximation to the task basis. The subtasks we uncover are shown to be highly intuitive and in some sense dual to the subtasks uncovered by popular existing methods [†]. Skills discovery is a notoriously difficult and unsolved problem in the field of reinforcement learning.

We conclude by presenting a hierarchical LMDP control architecture, and the associated execu-

[*] This work has been published as: Andrew M Saxe, Adam C Earle, and Benjamin Rosman. Hierarchy Through Composition with Multitask LMDPs. *Proceedings of the International Conference on Machine Learning (ICML)*, 70:3017–3026, 2017

[†] This work has been published as: Adam C Earle, Andrew M Saxe, and Benjamin Rosman. Hierarchical Subtask Discovery with Non-Negative Matrix Factorization. *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–11, 2018

tion model. The hierarchical architecture we present is unique in that it permits arbitrarily deep hierarchical structure in contrast with most prior methods which permit just one or two layers of depth. Moreover we develop a recursive procedure for constructing the deep compositional hierarchy. This recursive procedure is fully autonomous and the resulting control architecture may be constructed directly from the agent's experience in the domain. In so doing we present the first method for deep compositional decomposition of a domain, in support of a powerful multitasking RL agent.

The journey of a thousand miles begins with one step.

Lao Tzu

2

Preliminaries

2.1 Introduction

In this section we introduce a number of important concepts upon which much of the work in this thesis is built. Firstly, we introduce the reinforcement learning problem, in Section.(2.2.1), in the standard discrete formulation, along with some typical solution mechanisms. We then introduce an important extension to the standard framing in the form of the multitask learning paradigm in

Section.(2.2.2). Therein we consider how an agent might represent its experience internally, and subsequently transfer that experience to aid learning on new tasks.

Finally, in Section.(2.3), we introduce a special class of MDPs in which the Bellman equation is linearized and the optimal controller may be obtained through the solution of a linear eigenvalue problem. This special class of MDPs is a cornerstone for this work. This lesser known construction has a host of additional desirable properties which are used in the methods we develop throughout this thesis.

2.2 Reinforcement Learning

The term reinforcement learning has a broad range of potential interpretations. This proliferation stems, at its core, from the generality of the problem statement. Abstractly, reinforcement learning is often stated simply as the problem of learning, through iterative interaction with the environment, to achieve a stated goal, using only a global scalar reward signal and not detailed supervised feedback.

We will be concerned primarily with the interpretation of reinforcement learning as being the branch of optimal control theory in which the dynamics model for the system is unknown. Here, reinforcement learning takes on the role as a direct, adaptive method for finding the optimal controller[7]. The optimal control is directly estimated via a sequence of samples generated through interaction with the environment, rather than indirectly estimated by first constructing an explicit model of the system dynamics. This interpretation is also known as model-free reinforcement learning.

In addition, we will understand the term to cover the machine learning elements of the estima-

tion problem itself. In this sense, reinforcement learning differs from supervised learning in that the error signal is received only at the end of an episode, with each episode consisting of multiple sequential decisions. A central problem in reinforcement learning is therefore the so-called credit assignment problem, in which the error signal, obtained only at the end of an episode, must be suitably attributed to individual decisions within the sequence. Furthermore, samples are typically not independent and identically distributed (iid), which further complicates the estimation problem.

2.2.1 Standard Formulation

The reinforcement learning (RL) problem is typically modeled as an MDP, which may be specified as a 4-tuple $\langle \mathcal{S}, \mathcal{A}, P, \mathbf{r} \rangle$, where $\mathcal{S} = \{s_1, \ldots, s_N\}$ is a finite set of states, $\mathcal{A} = \{a_1, \ldots, a_M\}$ is a finite set of actions, $P \in \mathbb{R}^{(N \times N) \times M}$ is a tensor of state transition probabilities, such that $p_{ijk} = P(s_i | s_j, a_k)$ corresponds to the probability of transitioning into state s_i when choosing action a_k from state s_j, and finally $\mathbf{r} \in \mathbb{R}^N$ is a vector of real-valued state costs in which the j^{th} component corresponds to the scalar cost at state s_j. More generally, the scalar reward may depend on both the preceding state, and the action taken, i.e. $r(s_i, s_j, a_k)$. For simplicity we restrict our attention to the case in which rewards depend only on the final state. Wherever possible we will prefer the explicit array notation, as in $\mathbf{r}, p_{ijk}$, to the implicit functional notation, as in $R(s), P(s_i, s_j, a_k)$.

The Markov assumption tacit in the formulation is that the state of the stochastic process at some time t_n is independent of any preceding choices made at times $t_1, \ldots, t_{n-1}$. Formally, the conditional probability of future transitions, conditioned on all previous states, depends only on the current state. The RL problem can be stated for discrete or continuous state and action spaces; or indeed,

any combination thereof. Here we introduce the problem with both discrete state and discrete action spaces (but refer the reader to Bertsekas [48] for a formal description of the alternate formulations).

The solution to an MDP corresponds to finding a mapping from states to actions which maximizes some notion of cumulative reward (equivalently, minimizes some notion of cumulative cost). This state-action map is commonly referred to as a policy, and is typically denoted as $\pi : \mathcal{S} \rightarrow \mathcal{A}$. Note that the policy may be stochastic, instead mapping states to distributions over possible actions. A number of different formulations of the cumulative reward objective exist. In the finite-horizon case for example, the objective is to maximize the expected reward over a fixed number of steps. Alternatively, we may elect to optimize for the average expected reward over a fixed number of steps. In the infinite-horizon formulation, the objective is to maximize the cumulative reward over potentially infinite-length trajectories, with subsequent rewards being proportionally discounted so as to guaranteed asymptotic convergence. In the infinite-horizon setting MDPs are typically specified as a 5-tuple $\langle \mathcal{S}, \mathcal{A}, P, \mathbf{r}, \gamma \rangle$, where the additional term $\gamma \in (0, 1)$ is a discounting factor applied to future rewards [6,48].

Here we consider the first-exit formulation of the problem. We suppose that there exists some subset of the state space $\mathcal{S}_G \subset S$ which is *absorbing*. These correspond to states for which $p_{ijk} = \delta_i^j \ \forall \ a_k \in \mathcal{A}$, and $r_i = 0 \ \forall \ s_i \in \mathcal{S}_G$, where $\delta_i^j := (1 \ \textit{if} \ i = j \ \textit{else} \ 0)$ is the standard delta function. Once an agent enters such a state it is unable to move thereafter, and the agent incurs no additional cost for remaining in the state. This formulation is in fact a superordinate of the discounted infinite-horizon formulation [48]. These absorbing states are sometimes referred to as goal states in the

machine learning community.

The main approaches for learning in this context may be classified as being either model-based, or model-free. Model-based approaches first seek to estimate the unknown transition dynamics and reward structures $\hat{P}$, $\hat{\mathbf{r}}$, and then solve the resulting control problem using standard techniques. Model-free approaches on the other hand, seek to directly solve for the optimal controller without ever needing to model the transition or reward structure. Within the class of model-free solutions, a subsequent classification is useful. Methods may therein be identified as either directly searching policy space, such as policy gradient methods[49,50], or instead utilizing auxiliary functions to infer the optimal policy.

We are now in a position to formally define one such utility function - a useful quantity known as the value function. The value function $V^{\pi} : \mathcal{S} \to \mathbb{R}$ is defined for each state as the expected cumulative reward, for all future time steps, under a given policy π. This defines a natural notion of 'how good' a state is as:

$$v_s^{\pi} := V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} r^{t+k+1} \mid s^t = s\right], \tag{2.1}$$

where $\mathbb{E}_{\pi}$ denotes the expectation under the policy π, the superscript indices denote the time step, and $\mathbf{v}^{\pi}$ is the vector of scalar expectation values . If the absorbing set $\mathcal{S}_G$ can be reached with non-zero probability in a finite number of steps from any initial state, then the undiscounted infinite-horizon optimal value function exists, and is unique[48]. The optimal policy π^* is not necessarily unique, but all optimal policies share an optimal value function $\mathbf{v}^*$. Moreover, for deterministic

MDPs an optimal policy is necessarily greedy with respect to its value function. As such, many reinforcement learning algorithms are based on effectively estimating the value function at different states (or state-action pairs)[6]. With the optimal value function in hand, an optimal policy may then be easily extracted. The value function defined above is also known explicitly as the state-value function, highlighting its dependence only on the current state.

Another useful auxiliary function is the action-value function, more commonly known as the Q-value function. This action-value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined for each state-action pair as the expected reward for having taken action a, in state s under a given policy π. The Q-value function is explicitly defined as:

$$Q^\pi(s,a) = q^\pi_{s,a} = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} r^{t+k+1} \mid s^t = s, a^t = a \right]. \tag{2.2}$$

The definition of the Q-function is a useful way to expose the influence of particular actions on the state value function. Expressing this dependence explicitly is sometimes convenient for sampling methods. Note, however, that these quantities are trivially related:

$$v^\pi_s = \sum_{a \in \mathcal{A}} \pi(a|s) q^\pi_{s,a}, \tag{2.3}$$

where $\pi(a|s)$ is the probability of executing action $a \in \mathcal{A}$ from states $s \in \mathcal{S}$. Both the state value function and the action-value function satisfy the well-known Bellman Optimality condition; given

for the state value function below:

$$v_s^* = \max_{a \in A} \left\{ r_s + \sum_{s'} p_{s',s,a} v_{s'}^* \right\}. \tag{2.4}$$

The optimal value function can be obtained through a number of different methods. Direct Monte Carlo methods are the most obvious approach. These methods are based on the simple idea that averaging a sufficiently large number of random samples yields a sufficiently accurate estimate of the average value of the target quantity. Suppose that by running, or simulating, trajectories through the environment, we obtain k estimates for the value function at a given state $g_s^k = \sum_t r_t$. We may then estimate the value at state s as:

$$g_s = \frac{1}{k} \sum_{i=1}^{k} g_s^i, \tag{2.5}$$

where we are guaranteed that $g_s^k \to v_s^*$ as $k \to \infty$, under reasonable sampling assumptions[6,48].

When the transition dynamics P and reward structure $\mathbf{r}$ are known, the Bellman relation implies that the value iteration procedure $v_s = r_s + P_{s,s',\pi(s)} v_{s'}$ converges. An alternative approach to estimating the value function is to directly estimate this value iterate. By running or simulating trajectories through the environment, we acquire state transition samples of the form (s, r_s, s'). Notice that $r_s + g_{s'}$ is an unbiased estimate of the value iterate $r_s + P_{s,s',\pi(s)} v_{s'}$ in the sense that:

$$\mathbb{E}^{\pi} \left[r_s + g_{s'} | s \right] = r_s + \sum_{s'} P_{s,s',\pi(s)} v_{s'}. \tag{2.6}$$

This estimate for the value iterate is clearly noisy due to the stochastic selection of the sample values r_s, s'. In order to mitigate this effect, the value iteration is annealed by applying a multiplicative constant to the update procedure:

$$g_s = (1 - \alpha)g_s + \alpha(r_s + g_{s'}), \tag{2.7}$$

where $\alpha \in (0, 1)$ is referred to as the *gain* or *learning-rate* in the control theory, and RL communities respectively. Notice that since the estimate for v_s is dependent on another estimated quantity $g_{s'}$, this update procedure is a sort of bootstrapping method. This method is commonly known as temporal difference (TD) learning - more specifically TD(0) learning. The name highlights the fact that the sample estimate for the value iterate depends only on a single step look-ahead to s'. More generally the update may depend on multiple subsequent samples resulting in a range of update procedures collectively known as TD(λ) methods[6].

2.2.2 Mutlitasking and Transfer Learning

The multitask learning paradigm is a natural extension of the standard RL problem. Rather than being faced with a single task $\mathcal{M}_1$, the agent is instead faced with a set of tasks, $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$, each of which is specified as an independent MDP. The fundamental question is then whether or not the agent is able to improve learning on the set of tasks by sharing information between tasks. As such, the multitask learning paradigm, and the question of transfer learning are intimately linked. In one formulation of the problem, the tasks are presented incrementally, so that the agent has solved tasks

$\mathcal{M}_1, \ldots, \mathcal{M}_{k-1}$ before seeing task $\mathcal{M}_k$. In another formulation, the agent is presented with all k tasks simultaneously, and must attempt to solve them concurrently. In this way the multitask learning paradigm is also related to the multi-agent problem, in which each agent can be thought of as one parallel instantiation of a single agent executing multiple tasks.

The question of transfer learning is also central to the problem of lifelong learning. Indeed, lifelong learning is concerned precisely with how to incrementally adjust the internal representation of tasks so as to avoid catastrophic forgetting, while continuously freeing up new resources to continue to learn new tasks. At its core, the multitask learning problem is centered on a very general formulation designed to probe questions related to how information is stored within the agent, and the mechanisms by which that information is repurposed to improve performance on subsequent tasks.

In general, the tasks in the set are assumed to be related to each other in some way. By way of example, one might consider a set of locomotion tasks in which the agent operates with the same set of available actions, and with the same reward structure, but with different transition dynamics. This might correspond to a situation in which a robot is trained to navigate on a flat surface, and is subsequently tasked with navigating on a slanted surface. Similarly, one might imagine a set of tasks in which the agent operates with a different set of actions, but within the same state-space. This might correspond to a robot which is trained to perform a task with N limbs, and then is tasked with reaching the same goal state, but with one of its limbs removed.

In this work, we will be primarily concerned with the multitask formulation in which the tasks are specified in the same state and action space, but differ in their reward structures (see Taylor and Stone [51] for a more general review of transfer learning methods). More specifically, we will be con-

cerned with transfer methodologies based on hierarchical representations of the state, action, and task spaces.

2.2.3 Hierarchical RL and the Options Framework

As noted above, the estimation problem becomes exponentially more challenging as the dimensions of the state and action spaces grow. Hierarchy is one principled way to combat these issues, by directly tackling the curse of dimensionality. As a simple, illustrative example, suppose that an agent must choose between nine objects, being rewarded for selecting the correct one. Each object is specified by a tuple (colour, shape), where colour can be one of (red, blue, green) and shapes can be one of (circle, square, triangle). The reward associated with each object is computed as the sum $r(object) = r(color) + r(shape)$. One can now imagine two approaches for solving this task. The agent may either ignore the compositional manner in which the nine objects are constructed, and solve a single learning problem in the product space. Or, the agent may utilize the compositional manner in which the objects were constructed, and instead solve two simpler learning problems; one in which the best shape is identified, and another in which the best colour is identified. In the former, flat, approach the agent must solve a task with nine parameters, whereas in the latter hierarchical approach, the agent must solve a task with $2 \times 3 = 6$ parameters. In general, when hierarchical structure is present in a task, utilizing this structure allows the agent to operate in the summed space, rather than in the product space of parameters. This experiment corresponds to a hierarchical multi-armed bandit problem[52].

While hierarchy is an always good-in-principle approach, it presupposes that we have some *a pri-*

ori knowledge of the particular structure of the task at hand. Of course, in practice we do not always know that tasks have some latent compositional structure, or what that structure may actually be. Conceptually, this leads to a two-step learning paradigm in which the agent must first learn the correct hierarchical representation of the task, and then leverage this representation to actually solve the task. The hope is that the two-step learning problem is nevertheless significantly easier than solving the task in the product space of parameters.

Hierarchy has also been suggested as a powerful form of transfer learning. Extending our example above, suppose that the agent was initially tasked with finding the (blue, triangle), and has now been asked to find some new object, say the (red, square). If the agent had initially elected to work in the product space, it is not immediately clear how the agent might transfer the knowledge embedded in the optimal policy $\pi^*_{(blue,triangle)}$, to the new task. However, if the agent had elected to initially work with the hierarchical representation, this structure is immediately transferable to the new task - i.e. solving the disjoint problems of finding the best colour, and separately the best shape, again yields the correct solution. This is an example in which knowledge of the hierarchical structure itself constitutes a powerful form of transfer. However, one can imagine still greater benefits if the new task differs from the previous task along only one of the hierarchical dimensions. Suppose instead that the new task was to find the (blue, square), rather than the (red, square). This new task differs from the initial task along the shape dimension only. The agent may now transfer not only the hierarchical structure, but also the optimal policy for the colour task.

This exploitation of hierarchical task structure is thought to underpin much of the sample efficiency of human learning[22]. Rather than needing to learn from scratch in exponentially large task

spaces, we instead leverage the compact representations we have learned over a life-time of experience, and the inherent compositional nature of real-world tasks, to rapidly jump-start learning. These concepts have been codified in some RL frameworks, a selection of which we briefly review below.

Options

In the options framework temporally extended skills, commonly referred to as options, are added to the agent's repertoire, while still allowing the agent to plan at the level of primitive actions[6,53]. Explicitly, the additional temporally extended actions are appended to the agent's available action set, inflating the action space. Each 'option' is defined as a 3-tuple $\langle \mathcal{I}, \mathcal{T}, \pi \rangle$; where $\mathcal{I} \subseteq \mathcal{S}$, the initialization set, defines the subset of the state space from which the option may be executed; $\mathcal{T} : \mathcal{S} \to [0, 1]$, the termination map, defines the probability of terminating the option policy from each state; and $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, the option policy, defines the probability over actions for each state while the option policy is in force. It is common practice for an option to be executable from each state in which the option policy can continue. As such the options policy need only be defined over the initialization set.

With the incorporation of options into the action space, the resulting integrated learning problem is no longer Markovian. However, utilizing the fact that the temporally extended policies are themselves Markovian, many of the desirable properties that follow from the Markov assumption can in fact be salvaged. These sorts of decision problems are modeled as Semi-Markov Decision Processes (SMDPs)[30]. While the options framework utilizes the mathematical SMDP framework to

support the notion of temporally extended reward structures, unless some primitive actions are restricted for some states in a way different from the base MDP, the space of realizable policies in the options framework is identical to the base MDP.

From a conceptual perspective it is worth noting that all primitive actions can be interpreted as one-step options, so that the agent can be thought of as learning exclusively over options. Furthermore, by expanding each option state in the resultant optimal policy for the integrated learning problem, we may recover a non-hierarchical policy of the base MDP, however, the recovered 'flat' policy is generally not Markovian [6,53].

Hierarchies of Abstract Machines

Hierarchies of Abstract Machines (HAMs) [54], is an alternate approach to constructing hierarchical controllers that is conceptually quite different from the options framework. Where the options framework augments the action space of the agent to include temporally extended actions, resulting in a learning problem with more parameters; the focus in the HAMs setup is on simplifying the base MDP by restricting the class of realizable policies. This is done by defining the policies of the base MDP as 'programmes' which execute based both on the state of the base MDP, and their own current state. As such, the policy for the base MDP operates only over the available programmes, rather than primitive actions.

Each programme is realized as a finite state machine, and has four types of states; **ACTION**, **CALL**, **CHOICE**, **STOP**. When in the **ACTION** state, the machine determines a primitive action for the base MDP. When in the **CALL** state the machine suspends its execution and

passes control over to another machine; one which is available to it. From the **CHOICE** state, the machine stochastically transitions to its own next state. And finally, from the **STOP** state, the machine terminates its execution and returns control to the machine which initiated it. The machine to which control is returned is initialized to the state at which its execution was suspended. To ensure that the hierarchy is sensible, it is commonly assumed that the initial machine has no **STOP** action, such that the base MDP is always hung. Furthermore we assume that no loop through the hierarchy exists with execution probability 1 that contains no **ACTION** state. This would result in the base MDP receiving no primitive actions[54].

To illustrate this concept consider, in Fig.(2.1), an example similar to that initially proposed by Parr[54] in which a HAM is set up to solve a grid-world navigation task. In this simplistic setup, a designer has constructed two state machines, one of which is designed to follow walls, and another which is designed to retreat from them (back-off).

In Fig.(2.1) the initial machine is deterministically set to **CALL** the **FOLLOW − WALL** machine. This machine then determines a set of primitive actions for the base MDP until the base MDP either reaches an obstacle, at which point the **FOLLOW − WALL** machine enters a **CHOICE** state, or reaches an intersection, at which point the **FOLLOW − WALL** machine enters a **STOP** state and is terminated; passing control back to the initial machine.

A HAM can be thought of as a way of precisely defining a, possibly drastic, restriction on the class of realizable policies. This restriction is determined by a designer whose prior knowledge is used to construct the hierarchy, by specifying both the state machines and their relations. It has been observed that the only relevant states to determine the optimal HAM policy are the so called 'choice-

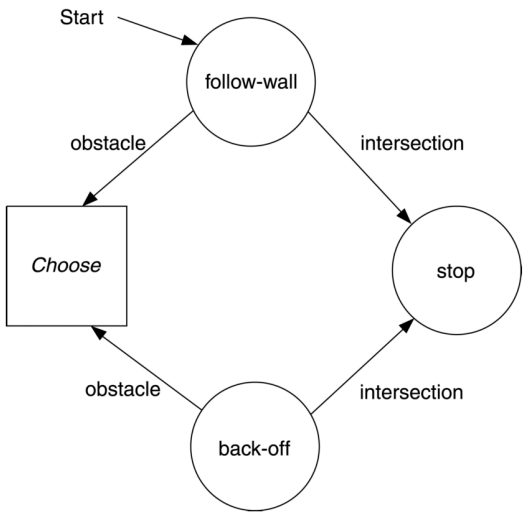


Figure 2.1: A depiction of the state-transition function for a simple HAM set up to solve a grid-world navigation task (reproduced from Barto and Mahadevan [53]). Here the designer has specified just two machines, which, executed suitably, are sufficient to solve the base navigation task.

points'[53]. In this way the HAM setup constitutes a potentially significant state abstraction as well. The optimal HAM policy implies a policy for the base MDP. However, there is no guarantee that this policy is close to the true optimal policy for the base MDP. The extent to which it will be a good proxy depends on the quality of the HAM and, in turn, the prior knowledge of the designer.

MAXQ Value Function Decomposition

Like the HAMs approach, the MAXQ algorithm also decomposes the base MDP into a set of abstract building blocks. However, where the constituents of the HAMs decomposition correspond to something akin to abstract actions, like $\mathbf{FOLLOW-WALL}$, and are therefore used recursively, the constituents of the MAXQ decomposition correspond to subtasks. The subtask decomposi-

tion is such that by performing the subtasks in the correct order, the base MDP is also solved. This approach to task decomposition is well aligned with the way in which people articulate task decomposition. By way of example, we may specify that the task of making a cup of tea is achievable by performing the subtasks, $\mathbf{GET-CUP}$, $\mathbf{ADD-TEABAG}$, $\mathbf{POUR-WATER}$, in order.

A number of natural properties emerge from this approach. Firstly, the subtasks themselves may be decomposed into ever finer-grained tasks. In this way the MAXQ decomposition allows for deeper control hierarchies to be naturally constructed. Continuing our example above, the subtask $\mathbf{GET-CUP}$, may itself be decomposed into the sequence of subtasks $\mathbf{OPEN-CUPBOARD}$, $\mathbf{GRASP-CUP}$, $\mathbf{PLACE-ON-TABLE}$. The resultant hierarchical subtask structure is depicted in a so-called task-graph. Once this task-graph has been constructed, the agent's action space is augmented to include execution of policies that solve the set of subtasks available to that level of the hierarchy. As such all of the constituent subtasks can be learned in parallel.

Formally the base MDP is decomposed into a set of subtasks $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$, each of which is specified as an SMDP. Typically, $\mathcal{M}_1$ denotes the root of the subtask-graph, such that solving the $\mathcal{M}_1$ subtask corresponds to solving the base MDP. To illustrate the concept of a task-graph decomposition, we consider the standard Taxi domain in which the agent (a taxi) navigates in a grid world which contains a number of predetermined special locations (stations) from which a passenger may be collected or dropped off. The agent is then tasked with picking a passenger up at one station, and then dropping them off at another. Here, a single episode of the task can be decomposed into:

1. picking the passenger up

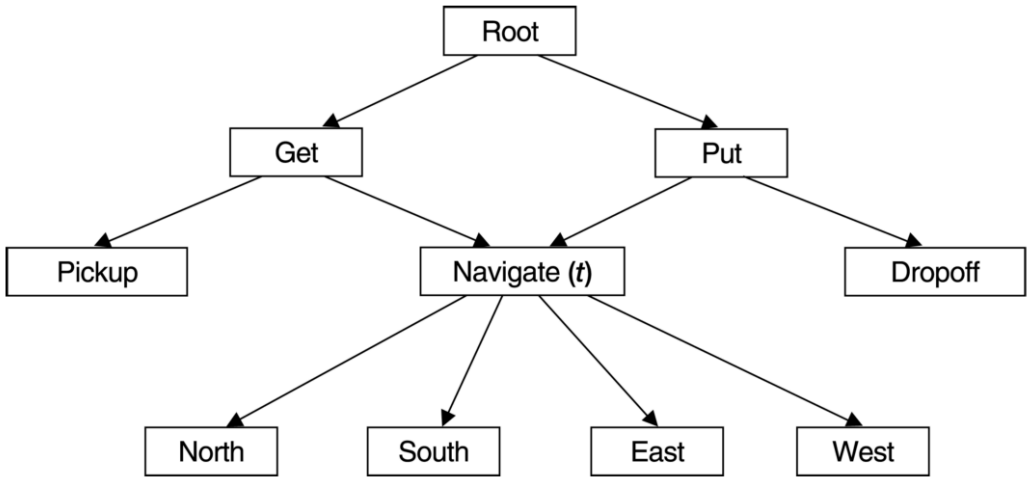


Figure 2.2: The task graph for the canonical Taxi domain (reproduced from [53]). At the highest level, the primary task can be decomposed into the subtasks **GET** and **PUT**, each of which can be further decomposed into the subtasks **NAVIGATE** and **PICKUP / DROPOFF**. Note that the leaf nodes in the graph correspond to the primitive actions which the agent must ultimately execute in the domain. Each of internal (and root) nodes can be solved in isolation as a separate MDP.

2. dropping the passenger off

Each of those subtask 'actions' can be further decomposed into:

1. pickup
2. navigate
3. drop off

The 'navigate' subtask can itself be further decomposed into primitive actions in the cardinal directions. The task graph for this problem is reproduced from Dietterich's initial concept[55] in Fig.(2.2).

Each subtask is defined by the 4-tuple $\mathcal{M}_i = \langle \mathcal{I}, \mathcal{T}, \pi, \mathbf{r}_{\mathcal{T}} \rangle$; where π, the subtask policy, defines the probability distribution over actions, which may include lower-level subtasks or primitives;

$\mathcal{I} \subset \mathcal{S}$, the initialization set, is the subset of the state space in which the subtask policy may be executed and continued, and $\mathcal{T} \subset \mathcal{S}$, the termination set, is the subset of states in which the subtask policy terminates. In the original MAXQ implementation, the initialization and termination sets constitute a complete decomposition of state space such that $\mathcal{I} \cup \mathcal{T} = \mathcal{S}$, and the subtask policy terminates with probability 1 when $s \in \mathcal{T}$. And, $\mathbf{r}_{\mathcal{T}}$, is the *pseudo-reward* function which assigns reward values to the subtask states, $\mathcal{T}$.

Each subtask $\mathcal{M}_i$ can be though of as a hierarchical option, but with an additional pseudo-reward $\mathbf{r}_{\mathcal{T}}$[53]. This pseudo-reward structure must be specified by the designer, and is similar to the auxiliary-reward function sometimes specified in the options framework to encourage learning of the options policies. The MAXQ decomposition has the additional property that primitives can be considered as a special case of subtasks so that an agent can be considered to learn exclusively against a hierarchy of subtasks. By appending to each base state an additional component which defines the 'execution stack', the hierarchy of calling subtasks, while the subtasks are non-Markovian with respect to the base state, they are Markovian with respect to the augmented states.

The methods we have considered thus far all take a slightly different approach to the construction of a hierarchical control architecture. Nevertheless, they all satisfy some version of the non-linear Bellman objective. Importantly, this means that none of these approaches support the sort of policy composition we argued for in Section.(1.3). In particular, supposing that the agent was provided with two optimal policies, it is not immediately clear how a third policy may be derived which captures some sort of composite behaviour. In the next section we consider a formulation of the control problem in which a form of optimal compositionality is permitted.

2.3 Linear Markov Decision Processes

In recent years, many hard problems have been effectively solved by finding procedures that transform them into structurally easier problems, which can then be solved using linear methods or convex optimization techniques. Even when exact transformations are not possible, good approximate solutions are often achievable. Historically, this approach has not seemed viable for reinforcement learning. Indeed, the discrete and unstructured nature of general MDPs seems diametrically opposed to the smoothing assumptions typically required to relax the problem constraints.

However, by carefully specifying the problem parameters, a specialized family of MDPs can be constructed for which the optimization over the control space is convex[45,56]. This simplification allows for a transformation in which the Bellman equation Eqn.(2.4) becomes linear, and the optimal control can be written in closed form. This specialized family of MDPs are consequently known as Linear MDPs (LMDPs).

Conceptually speaking, the key assumption that allows for this simplification, is that optimal policies are inherently stochastic. These stochastic optimal policies have the dual interpretation as being either stochastic policies over deterministic actions, or deterministic policies over stochastic actions. This ultimately has the effect of replacing the hard maximization in the Bellman equation over the discrete action space, with a differentiable soft-max over a continuous action space. The resulting transformation yields a differentiable objective that can be solved in closed form.

Explicitly, the resulting formulation is a control problem with a finite state space, and a continuous control space. There exists a general embedding procedure by which standard MDPs can be

described within the LMDP formalism. We will see that a wide range of control tasks permit a description in the LMDP formalism which yields a good approximate solution.

In addition, the linearization assumptions make the resulting construction much more analytically tractable. The additional structure present in this class of simple linear models may be exploited to develop analytical insights into a number of important areas, yielding powerful intuitions about the broader problem.

2.3.1 Formulation

Formally, an LMDP is specified as a 3-tuple $\langle \mathcal{S}, P, \mathbf{r} \rangle$, where $\mathcal{S} = \{s_1, \ldots, s_N\}$ is a finite set of states, $P \in \mathbb{R}^{N \times N}$ defines the action-free state transitions, and $\mathbf{r} \in (-\infty, 0]^N$ defines the real-valued state costs. Here, we consider in detail the absorbing state formulation in which a subset of states $\mathcal{S}_\mathcal{B} \subset \mathcal{S}$ are absorbing and incur no state cost, $p_{ij} = \delta_i^j$ and $r_i = 0$ for all $s_i \in \mathcal{S}_\mathcal{B}$; although similar transformations exist for other problem formulations[45]. For ease of discussion, the states space is decomposed into the set of N_i *interior* states $\mathcal{S}_\mathcal{I}$ and the set of N_b absorbing *boundary* states $\mathcal{S}_\mathcal{B}$ such that $\mathcal{S}_\mathcal{I} \cup \mathcal{S}_\mathcal{B} = \mathcal{S}$.

In the LMDP, the control is a real-valued vector $\mathbf{u} \in \mathbb{R}^N$ which acts to shift the probability mass of the action-free state transitions, such that $\hat{p}_{ij}(\mathbf{u}) = p_{ij} \exp(u_i)$. We refer to the resulting Markov chain as the controlled dynamics; in contrast with the action-free transition probabilities, which we refer to as the passive dynamics. Notice that this specifies a very general notion of control; the controller is allowed to rescale the passive dynamics in any way it chooses, subject to the implicit constraints, $p_{ij} = 0 \implies \hat{p}_{ij}(\mathbf{u}) = 0$, and $\sum_i \hat{p}_{ij}(\mathbf{u}) = 1$.

In addition to the state cost $\mathbf{r}$, we specify an additional control cost which the agent must pay for shifting the transition dynamics. Since the control is real-valued and acts to continuously shift the underlying passive dynamics, a natural form for the control cost is as a divergence of the resulting transition probabilities. Specifically we take the control cost to be:

$$c(s, \mathbf{u}) = \mathrm{KL}\left(\hat{\mathbf{p}}_{s:}(\mathbf{u}) \;||\; \hat{\mathbf{p}}_{s:}(\mathbf{o})\right), \tag{2.8}$$

where *KL* represents the Kullback-Leibler divergence. It follows immediately from the properties of the KL-divergence that $c(s, \mathbf{u}) \geq 0$, and $c(s, \mathbf{u}) = 0 \iff \hat{p}_{s:}(\mathbf{u}) = \hat{p}_{s:}(\mathbf{o})$. A natural interpretation here is that the agent incurs an additional control cost when it wishes to command a transition structure different from the passive dynamics. Moreover, the agent will pay a greater cost when the commanded transitions differ significantly from the passive transition structure. Finally we have, $c(s, \mathbf{u}) = \infty$ when $\hat{p}_{s:}(\mathbf{o}) = 0$ and $\hat{p}_{s:}(\mathbf{u}) \neq 0$; thus the agent cannot command a transition that is prohibited by the passive structure. The total cost is then taken to be the sum of the state cost and the control cost:

$$l(s, \mathbf{u}) = r_s + \lambda c(s, \mathbf{u}), \tag{2.9}$$

where $\lambda \in \mathbb{R}$ is an inverse-temperature like regularization parameter, scaling the relative control cost.

The passive dynamics are specified as the resultant Markov chain under some reference policy,

and the passive transition structure is in fact better represented as $P_{\bar{\pi}}$ to explicitly capture this dependance, where $\bar{\pi}$ is the relevant reference policy. The reference policy is a free-parameter to be chosen by the designer, although the uniformly random policy is a typical choice. In so much as the optimal policy can be said to be regularized by the reference policy via Eqn.(2.8), the choice of the uniformly random policy as the reference policy corresponds to an entropy-regularized RL objective. This formulation has been the subject of some recent work[57,58,59]. More generally, the reference policy may correspond to some expert demonstration, or an alternate policy which is known *a priori* to exhibit some desired behaviour.

Under these conditions, the Bellman equation takes the following form:

$$v_s = min_u \left(r_s + \sum_{s'} p_{s,s'} \exp(u_{s'})(u_{s'} + v_{s'}) \right), \tag{2.10}$$

where $v(\cdot)$ is the standard state value function[45]. The fact that controls are real-valued and act to scale the transition structure multiplicatively makes the Bellman objective differentiable. Indeed, the objective specified in Eqn.(2.10), can be solved exactly using the method of Lagrangian multipliers, and the optimal control $\mathbf{u}^*$ can be written in closed form. Substituting the closed form optimal control into Eqn.(2.10) makes the Bellman equation linear. The optimal value function can then be expressed as:

$$\exp(-v_s) = \exp(-r_s) \sum_{s'} p_{s,s'} \exp(-v_{s'}). \tag{2.11}$$

Under the exponential transform $z_s = \exp(-v_s)$, the matrix form of the linear Bellman equation

may be written as:

$$\mathbf{z} = GP\mathbf{z}, \tag{2.12}$$

where $G = diag(\exp(-r_s))$, and $\mathbf{z} \in \mathbb{R}^N$ is the vector of the exponentiated value function. The exponentiated value function $\mathbf{z}$ is referred to as the desirability function. Finally, the resultant optimally controlled transition dynamics can be written in terms of the optimal value function as follows[45]:

$$\hat{p}_{s,s'}(\mathbf{u}_s^*) = \frac{p_{s,s'} \exp(-v_s)}{\sum_{s'} p_{s,s'} \exp(-v_{s'})}. \tag{2.13}$$

When a model for the continuous MDP is not available, the optimal desirability function must be stochastically approximated. In the typical online RL setting, at each step we observe a sample as the tuple (s, s', r_s), corresponding to the state transitions $s \to s'$ and the associated state reward r_s. A simple 1-step approximation is then:

$$\hat{z}_s = (1 - \alpha) z_s + \alpha \exp(-r_s) z_{s'}, \tag{2.14}$$

where the learning rate $\alpha \in \mathbb{R}$ is suitably annealed.

Taking stock: the specific assumptions made in the formulation, coupled with the special form of the control cost, has transformed the objective function for this class of optimal control problems into a linear eigenvalue problem. It can be shown that under the assumptions specified above, there exists an eigenvector solution to Eqn.(2.12), with eigenvalue 1, and that this solution is unique.

One important way in which the optimal policies for LMDPs differ from their standard MDP

counterparts is that the resultant optimal policies are inherently stochastic. This is an immediate consequence of the special form of the control cost specified in Eqn.(2.8), in which a deterministic optimal policy may incurs an infinite control cost[45]. Conceptually this corresponds to a relaxation over the discrete action space which yields the convex objection in Eqn.(2.10).

2.3.2 Properties

The class of LMDPs has a number of unique properties that are not present for general MDPs. We explore some notable examples here.

As mentioned above, the special form of the control cost specified in Eqn.(2.8) has the interpretation of regularizing the optimal policy by the reference policy implicit in the passive dynamics. In terms of modeling real-world phenomena via the LMDP formalism, the passive dynamics would likely correspond to the transition dynamics in the absence of an external control. In this setting the regularization has the interpretation of specifying a preference for *energy efficient* controls, and the optimal control may be interpreted as the minimum energy solution. This is a non-trivial connection. In so much as the physics of the real-world obeys the principle of least action (more accurately, stationary action), the LMDP formalism is an inherently good model for many of these sorts of tasks. Said differently, the discrete and unstructured nature of general MDPs does not utilize the fact that real-world control tasks are often, to good approximation, smooth and local. The LMDP formalism on the other hand, makes critical use of these properties, and may therefore be a good model despite its simplifying assumptions.

The temperature parameter in Eqn.(2.9) determines the relative weight of the control cost and

the state cost. Practically speaking, this parameter controls the relative stochasticity of the optimal policy. In the low temperature limit, $\lambda \rightarrow 0$, we recover the standard RL objective which depends on the state cost alone, and the corresponding optimal control nears determinism. Alternatively, for $\lambda >> 0$ the cost incurred by the agent is almost exclusively due to the divergence from the passive transition structure, and the resulting optimal policy is stochastic.

A natural question to ask is which MDPs can be represented as LMDPs - or more generally, whether a mapping exists between MDPs and LMDPs. Indeed, a general procedure for embedding MDPs within LMDPs was defined in Todorov [45]. Of course, the approximate solutions obtained from the LMDP are not always good approximations for the optimal solutions for the original MDPs. Clearly, when the origin MDPs violate the smoothing assumptions tacit in the LMDP, the approximate solutions may be poor. The resulting rule-of-thumb is that MDPs that have approximately local structure, can be embedded as LMDPs whose solutions will be good approximations to the origin MDP. The term local structure here refers to the property that states that are close in terms of expected transition lengths, are close in terms of value. As noted above, this property holds for many real-world control tasks.

One example of a control task which is suitable for LMDP embedding is the shortest path problem [45]. Here the agent operates in some discrete state space, and must uncover an optimal policy for reaching some goal state in the minimum number of transitions. Clearly this problem has the local property described above: if a state has high value, since it is close to the goal state, then all neighbouring states must also have relatively high value, since they too are close to the goal state. The LMDP has also been successfully applied to a number of non-navigation tasks, such as the classic

towers-of-hanoi problem[60], and the TAXI problem[46].

Similarly, there are any number of MDPs for which the approximate solution via the LMDP embedding is poor. Indeed, any MDP that does not satisfy the local assumption specified above, will suffer in this way. As a simple example, one might imagine a navigation task in which there exists a short trajectory which incurs some negative state cost, and some long trajectory which incurs zero state cost. The optimal solution to the original MDP is clearly to follow the long, but cheap trajectory. The approximate solution yielded by the LMDP would correspond to the short trajectory incurring the non-zero state cost. This happens due to the fact that the control cost associated with the long trajectory is non-zero, and can always be made to exceed the state cost incurred through through the short trajectory.

In terms of practical application, the LMDP also suffers from the fact that a passive transition structure must be known *a prior* for the task at hand. Fortunately, the Markov chain resulting from the uniformly random policy is often sufficient. Nevertheless, recent work has sought to overcome this limitation by applying the same smoothing assumptions, and resulting linearization to the state-action value function (also known as the Q-function) directly. This simplified formalism has been the subject of recent work under the banner of so-called soft-Q learning, or entropy-regularized RL[61,62].

In addition, the linearity of the Bellman equation allows for a number of compositional features which are not present for general MDPs[63]. This particular property is of central importance in the following chapter, and is discussed in greater detail therein.

In an exciting development, some recent work has sought to combine the unique properties of

the LMDP formulation with the benefits of a hierarchical task description. By formulating the constituent subtasks of a MAXQ decomposition as individual LMDPs, the authors are able to leverage the compositional properties of the LMDP to solve the subtasks in parallel. This is shown to be significantly more sample efficient on some standard RL domains[64]. Conceptually, the authors exploit the compositionality to solve a single task more rapidly, once the hierarchical structure has been uncovered via some other method.s

The whole is more than the sum of its parts

- Aristotle

3

Multitask Reinforcement Learning with Compositional Policies

3.1 Introduction

The standard reinforcement learning problem is concerned with finding a policy which maximizes the expected, cumulative, life-time reward on a particular task. In the multitask setting, the situa-

tion is considered in which the agent will face not just one, but many tasks. These tasks are usually assumed to be related to each other in some way - typically occurring within the same state space or action space, or having reward functions drawn from a shared distribution. A distinction is sometimes made between *multitask*-learning, in which the agent is aware of all of the tasks to be solved from the beginning, and *incremental*-learning, in which tasks are presented one at a time[65]. Here we understand the term 'multitask learning' to cover both paradigms. The multitask problem setting is well suited to explore concepts related to how information is stored and represented within the agent, and the mechanisms by which new information is assimilated, and existing information is repurposed to improve learning on new tasks. In line with these two distinct framings of the problem, two primary questions are often considered:

1. Knowing *a priori* that the agent will face a number of related tasks, can the learning process be augmented to achieve better performance on some global metric?
2. Having gained knowledge incrementally by solving related tasks, can this prior knowledge be leveraged to improve learning on future tasks?

We will be concerned primarily with the second of these questions, namely: how to utilize previous experience to improve learning on new tasks.

Our interest in understanding how previous experience might be brought to bear on new tasks is motivated, in part, by the fact that current reinforcement learning algorithms are notoriously sample inefficient. This is in stark contrast with human learners who, when faced with a novel task, are often able to immediately perform well. The sort of zero-shot learning of which humans are capable can be attributed to their ability to flexibly leverage previous experience, and repurpose

this experience to aid learning on the new task. In order for our algorithms to achieve comparable sample efficiency in learning, they will need to be imbued with similar capabilities.

When we consider the human ability to execute zero-shot learning on new tasks, we typical envision scenarios in which the learner is explicitly *instructed* to achieve some goal. This corresponds to the situation in which the reward structure for the task is provided up front. Note that this setup corresponds more to a probabilistic planning problem, rather than a strict reinforcement learning problem in which the agent typically has no up front knowledge of the reward structure. In this chapter we assume that the agent has access to the reward function for the target task, such that the goal is explicitly stated. The classical RL setup is considered in Chapter.(4).

Previous authors have utilized many diverse approaches in the construction of transfer learning methods[51,66]. Despite this diversity, existing methods remain critically exposed to the curse of dimensionality due to the fact that these methods do not exhibit the critical property of compositionality. Not only does compositionality allow for a more economical representation of tasks that contain compositional structure within the agent[67], but also it presents a principled way to tackle the innate scaling issues which arise as the state and action space increase.

While compositionality appears to be an always good-in-principle idea, current RL algorithms typically do not exhibit this property. It seems likely that this is due, fundamentally, to the inherently unstructured nature of the MDP formulation. Indeed in the standard setting, problems are typically specified as *general* MDPs, for which no additional structure is assumed. In the general case, MDPs which constitute a class of non-convex optimization problems which satisfy a non-linear optimality condition. This creates a backdrop against which composition would seem to be struc-

turally impossible.

Motivated by the intuition that compositionality is an essential component of any transfer learning mechanism which hopes to scale, we appeal to a class of MDPs in which a form of optimal composition is guaranteed to hold. This special class of MDPs corresponds to a convex relaxation over the actions space, and satisfy a linear optimality condition. Here, and throughout, we make extensive use of these so-called linear MDPs (described in detail in Section.(2.3)). In addition to supporting some form of compositionality, these linear models are also more analytic tractability.

In addition to utilizing the LMDP formulation, we make a further simplifying assumption. We consider a specific subset of multitask learning problems in which the constituent tasks occur in the same state and action space for the agent, but differ in their reward structures. A prototypical example of this class is a navigation problem in which the agent must reach a different set of states for each constituent task. Within this particular framing we develop a novel transfer method. This new method makes pivotal use of the compositional property inherited from the LMDP problem formulation by expressing new tasks as distributed representations of a set of basis tasks, and is shown to exhibit powerful transfer capabilities. Furthermore, leveraging the analytic tractability of the LMDP formulation, we derive a class of exact solutions to the transfer problem in terms of a set of the basis tasks, and show explicit bounds on the quality of transfer possible when the full basis is unavailable.

The new method exhibits a number of interesting properties which help to contextualize the method in terms of existing approaches to transfer learning, and indeed to human learning as well. Note that in the context of multitask learning, human learning is often observed to occur in two distinct phases. In the first phase, the learner makes very rapid progress on the task. This is followed

by a second phase of learning in which progress is incremental and occurs more slowly. The second phase of learning is characterized both by the slow and steady nature in which performance improves, and by the exploratory nature of the trial-to-trial variation. In order to build autonomous agents that are capable of the same sort of flexible adaptation in the face of changing objectives, we require learning algorithms that are ostensibly capable of both modes of learning. Many existing transfer learning methods can also be said to contain two analogous phases of learning. In the first, the agent attempts to uncover a good initial policy by relating the new task to those which it has seen before. In the second phase of learning, no further transfer takes place, and the agent improves its performance in an incremental way through the back propagation of error signals. The new method which we develop in this chapter will similarly exhibit such a bipartite execution paradigm.

In Section.(3.2) we exploit the tractability of the unusual mathematical construction to develop a formal mechanism for exact policy transfer, and introduce some central ideas which run throughout the chapter. In Section.(3.3) we consider a generalization to inexact composition, and finally we consider the semantic nature of the particular form of composition our method supports in Section.(3.4), and the tacit limitations therein.

3.2 The Importance of Compositionality and Concurrency

The need for compositional elements in our algorithms is not a new idea. In the field of visual scene recognition for example, scenes are often described as being composed of objects and their relations; while objects themselves are described as being composed of low-level features. Moreover, this com-

positional structure is tacitly present in the architectures of the state-of-the-art deep neural networks used to perform these classification tasks; wherein low-level neurons correspond to primitive features (like edge detectors), while high-level neurons correspond to specific composite features (like faces). Generally speaking, such architectures can be thought of as hierarchical feature detectors. There is much evidence to suggest that a similar form of feature compositions is present in human perceptual hierarchies [41,42,43,44].

Despite the evidence for a compositional approach to motor control in the brain, current state-of-the-art deep RL methods do not exhibit this compositional property, and research in this direction seems to be limited (but see, da Silva et al. [68], Haarnoja et al. [62], van Niekerk et al. [61]). Instead, deep learning techniques have largely been employed to solve other difficult aspects of control tasks; namely long time-horizons and sparse reward signals. While there are likely numerous reasons that compositional approaches to RL remain uncommon, we pause to call out just a few. Perhaps most importantly, within the standard RL formalism, a direct attempt at policy composition is immediately prohibited by the non-linearity of the Bellman optimality condition. Furthermore, the notion of compositional learning is most fruitful in a multitask setting - one might then argue that it is simply a step too far to attempt to solve adaptive multitask agents, when we can't yet robustly solve single task agents in large domains.

We consider a specific class of multitask learning problems in which an agent operates in some discrete domain in which the state-space $\mathcal{S}$, action-space $\mathcal{A}$, and transition dynamics P are fixed, and the agent is tasked with finding a set of policies which maximize reward for a set of differing reward functions. In this setting a specific task is completely defined by its corresponding reward function

$\mathbf{r}$. Note that despite the fact that the state and action spaces are finite, an infinite number of possible tasks may be specified in this domain. To see this, suppose that there are $N \in \mathbb{Z}^+$ discrete states in the domain. A task with a single goal state can be constructed by placing a fixed non-zero scalar reward at just one of the states. There are of course N possible single-state tasks. Similarly, a task with two goal states can be constructed by placing a fixed non-zero scalar reward at two of the states. There are $\binom{N}{2}$ possible state-pairs. Continuing in this fashion, it is clear that 2^N possible tasks can be specified by placing a fixed scalar reward at some states, but not others. This construction enumerates the number of tasks by specifying the binary distinction between rewarded states and unrewarded states. Of course, more generally speaking, rewards are real valued scalars, and as such by varying the value of the scalar rewards at each state an infinite number of tasks can be specified.

This explosion in the number of possible tasks is a fact that often goes unmentioned. As a tangible example of this fact, an agent operating in a simple 10×10 grid-world domain, would need to solve a million trillion trillion individual policies in order to successfully solve all possible navigation tasks (since $2^{100} \approx 10^{30}$). This is a nauseating realization which flies in the face of our common sense reasoning. It would certainly seem to be the case that if we know how to navigate, separately, to location A, and to location B, that we should be able to immediately navigate to 'A-or-B'. At the heart of this intuition is our ability to discern the composite structure of these tasks, and the assumption that previous knowledge can be easily leveraged to achieve them.

In the multitask setting we consider, the agent is faced with some $N_t \in \mathbb{Z}^+$ tasks in the domain. Suppose that the tasks are presented sequentially, so that when the agent faces the N_t^{th} task, it has access to the reward function $\mathbf{r}^k$, optimal value function $\mathbf{v}^k$, and optimal policy π^k for each of the

preceding tasks $k \in \{1, \ldots, N_{t-1}\}$. In pursuing our primary question, we would then like to better understand how, when encountering a new task, the agent is best able to utilize its experience to improve its performance on the new task.

3.2.1 Common Questions and Existing Approaches

When the agent knows *a priori* that it will face a number of tasks in the domain, its learning procedure may be altered to leverage the shared structure in the value functions for the tasks it will face (see Borsa et al.[69] for a recent example). When tasks are specified incrementally, the agent must perform two conceptually disjoint operations. Firstly it needs to assess how similar the new task is to those it has experienced before, and secondly it needs to transfer its previous experience to the new task. To improve learning on the new task may mean to improve the agent's initial performance, improve the agent's asymptotic performance, or to improve the agent's rate of improvement (or indeed some combination of these)[65,66]. Unless otherwise specified, we will take the improvement through transfer to mean that the agent experiences a jump-start in its initial performance, and that its rate of improvement is increased.

The problem of transfer learning in RL has been approached in a number of broadly varying ways. Each of these approaches is based on the central observation that generalization occurs not just within tasks, but also across tasks. Some methods attempt to reformulate the base task as a hierarchy of abstracted tasks[70], arguing that the high-level structure is often better suited to transfer. Other approaches seek to augment the agent's action space by providing additional, temporally-extended, actions[30], again arguing that the abstracted structure is better suited to transfer. In perhaps a more

direct approach, a class of methods exist which attempt to directly augment the reward structure for the new task - essentially providing bread-crumbs for the agent to follow[71,72]. The reader is directed to Taylor and Stone[51], Lazaric[66] for a thorough overview of the field.

In order to call-out two fundamental and broadly applicable challenges in the space, we briefly consider the well-known *options* frameworks[30]. In this setup the agent is equipped with an additional set of subpolicies $\mathcal{O} = \{O_1, \ldots, O_k\}$, each of which is specified by a 3-tuple $O_i = (\pi_i, \mathcal{I}_i, \mathcal{T}_i)$. Here $\pi : \mathcal{S} \rightarrow \mathcal{A}$ defines the actual state-action map for the subpolicy, $\mathcal{I} \subseteq \mathcal{S}$ indicates the subset of the state space from which the subpolicy can be initiated, and $\mathcal{T} \subseteq \mathcal{S}$ defines the subset in which it may be terminated. In the simplest case the initialization set is taken to be the full state space $\mathcal{I} = \mathcal{S}$, and the termination set is taken to be a singleton goal state $\mathcal{T} = s \in \mathcal{S}$ - here, executing the subpolicy corresponds directly to the temporally-extended action 'go to goal'. When the agent elects to execute one of the subpolicies, it will follow the action-selection protocol specified by that policy until it is terminated. In this way the subpolicies can be thought of as multi-step actions, which are mathematically unified with the standard RL formalism through the framework of semi-Markov decision processes (SMDPs).

At any point in time the agent must choose an action from the augmented action space $\hat{\mathcal{A}} = \mathcal{A} \cup \{O_i | s \in \mathcal{I}_i\}$. This is the first important difficulty facing any approach whose transfer mechanism inflates the agent's action space. Since the number of parameters the agent must learn is exponential in the number of available actions, these approaches are fundamentally limited in their ability to scale. A stark example of this limitation is provided in[46], in which a robot must solve the task of finding the nearest charging station, given the optimal policy to reach each of the charging stations

individually. Even though the agent already possesses the optimal policy in its repertoire, learning to choose between these policies is slow, and scales poorly as the number of available policies increases.

The second important challenge faced by transfer methods is that of *negative transfer*. This is the observation that sometimes the effect of transfer can actually be to slow down learning on the target task. Importantly the agent is, in general, not able to assess up front whether or not transfer will result in a positive or negative effect. In the options framework for example, the agent may elect to execute a subpolicy that takes the agent further from the goal. Worse still, this subpolicy remains in effect until it is terminated (but see, Sutton et al.[30], Comanici and Precup[73] for approaches to 'early stopping'). Negative transfer is often attributable to a failure in the agent's ability to either accurately measure task similarity, and thereby transferring deleterious information, or to effectively structure transfer.

Much of the difficulty in measuring task similarity and transfer results from the complex interplay between the reward structure and the transition dynamics. A simple idea which follows from this observation is to find a decoupled representation of the value function in which the reward structure and transition dynamics are naturally separated. Consider a linear function approximation in which the value function is determined as a linear combination of a set of feature vectors, $\mathbf{v}^\pi = \mathcal{A}\mathbf{w}$. Since the value function at a given state corresponds to a biased sum of the value function at potential successor states, the set of features in $\mathcal{A}$ amounts to a representation of potential successor states. Given the linearity of the approximation, it follows that a good representation for states is one in which successor states have a similar representation in terms of euclidean distance between the feature vectors. Following this intuition, it was shown in Dayan and Hinton[71] that a

suitable state representation for linear approximation schemes is in terms of the so-called successor representation $\mathcal{M} = (I - P^{\pi})^{-1}$, which encodes the expected future visitation count for all states, where P^{π} is the resulting Markov chain underlying policy π. The value function is then determined as:

$$\mathbf{v}^{\pi} = \mathcal{M}^{\pi}\mathbf{w}, \tag{3.1}$$

where $\mathbf{w}$ can be thought of as the reward contribution to the value function, notably decoupled from the transition contribution captured in the successor matrix $\mathcal{M}$. This decoupled representation has been exploited in recent work to derive some important theoretically grounded approximate transfer results[74].

3.2.2 A New Approach to Transfer

Our approach to multitask learning is grounded in a linearized formulation of the standard RL problem. The linearized MDP provides us with an analytically tractable framework from which we are able to derive a number of explicit transfer results not available to other methods. We find that the unusual mathematical construction provides a very intuitive way to formalize the notion of task similarity, and transfer. Moreover, we are able to guarantee that negative transfer does not occur with the specified transfer mechanism. The specific mechanics of transfer are further exploited to answer a number of meta-questions pertaining to source task selection and experience distillation.

In contrast with the alternate approaches outlined thus far, the LMDP formulation implies a form of guaranteed optimal composition. Recall that the LMDP was described in Section.(2.3) in

terms of the absorbing state formulation in which a subset of the state space $\mathcal{S}_{\mathcal{B}} \subset \mathcal{S}$ (which we refer to as the set of *boundary* states) is absorbing and occur no additional state cost. The specific statement of the compositionality supposed in the LMDP formulation is then as follows: if the exponentiated boundary reward vector for the new task $\mathbf{q}^t_{\mathcal{B}} = \exp\left(\mathbf{r}^t_{\mathcal{B}}\right)$ can be written as the linear combination of the exponentiated boundary reward vectors for previous tasks $\{q^k_{\mathcal{B}}\}$, then the optimal desirability function for the new task $\mathbf{z}^t$ can be immediately obtained as the same linear combination of the optimal desirability functions for the previous tasks $\{\mathbf{z}^k\}$ [63]:

$$\mathbf{q}^t_{\mathcal{B}} = \mathcal{Q}_{\mathcal{B}}\mathbf{w} \quad \Rightarrow \quad \mathbf{z}^t = Z\mathbf{w}, \tag{3.2}$$

where $\mathcal{Q}_{\mathcal{B}} = \left[\mathbf{q}^1_{\mathcal{B}}, \ldots, \mathbf{q}^{N_t}_{\mathcal{B}}\right] \in \mathbb{R}^{N_b \times N_t}$ is the matrix whose columns are the exponentiated boundary reward vectors, $Z = \left[\mathbf{z}^1, \ldots, \mathbf{z}^{N_t}\right] \in \mathbb{R}^{N \times N_t}$ is the matrix of corresponding desirability functions, and $\mathbf{w} \in \mathbb{R}^{N_t}$ is a weight vector. Here again $N_b, N_t \in Z^+$ denote the number of boundary states and the number of subtask states respectively. To see explicitly why this relation holds, consider the specific case in which $\mathbf{q}^3_{\mathcal{B}} = \alpha\mathbf{q}^1_{\mathcal{B}} + \beta\mathbf{q}^2_{\mathcal{B}}$. Then from the Eqn.(2.12) we have:

$$\mathbf{z}^3_{\mathcal{I}} = M\mathbf{z}^3_{\mathcal{I}} + N\mathbf{z}^3_{\mathcal{B}},$$

$$\mathbf{z}^3_{\mathcal{B}} = \mathbf{q}^3_{\mathcal{B}},$$

where $\mathcal{M} = diag\left(\mathbf{q}^3_{\mathcal{I}}\right) P_{\mathcal{II}}$ and $\mathcal{N} = diag\left(\mathbf{q}^3_{\mathcal{B}}\right) P_{\mathcal{IB}}$. Since the boundary rewards $\mathbf{q}^3_{\mathcal{B}}$ are composite, we may write the solution for $\mathbf{z}^3_{\mathcal{I}}$ explicitly as:

$$\mathbf{z}^3_{\mathcal{I}} = \alpha \left(I - \mathcal{M}\right)^{-1} \mathcal{N} \mathbf{q}^1_{\mathcal{B}} + \beta \left(I - \mathcal{M}\right)^{-1} \mathcal{N} \mathbf{q}^2_{\mathcal{B}},$$

$$\mathbf{z}^3_{\mathcal{I}} = \alpha \mathbf{z}^1_{\mathcal{I}} + \beta \mathbf{z}^2_{\mathcal{I}}.$$

The agent is therefore able to immediately perform optimally on the new task with no further learning required. The ability to immediately perform well on a new task is sometimes referred to as 'zero-shot' learning. There are a few important conceptual features of the LMDP that deserve further mention.

Firstly, the optimal desirability function $\mathbf{z}^t$ is realized as a combination of multiple policies running *concurrently*. This behaviour is in stark contrast with the standard formalism in which actions must be selected in a mutually exclusive fashion, and must be executed sequentially. This behaviour is possible because the actions in the LMDP formalism corresponds to full distributions over next states, meaning that an action taken at a single point in time can be built up from multiple sub-actions taken in parallel[46]. A simplistic demonstration of this concurrency can be seen in Fig.(3.1 - Concurrent Execution). Here, the agent operates in a 1-dimensional corridor with uniform transition dynamics. The resultant desirability function clearly exhibits properties of both constituent's desirability functions simultaneously. At any point in time, the agent acts under the influence of both desirability functions which operate in parallel to produce the agent's overall behaviour.

In the options framework, these sub-policies are provided to the agent as additional actions[30]. This standard procedure has the effect of inflating the action space. Since the number of parameters to learn scales as the exponential of the number of actions, this can be crippling[75]. In contrast, the concurrent execution of actions in the LMDP formalism means that the agent does not need to *choose* between the sub-policies, instead executing all of them in parallel. Memory considerations aside, this implies that having access to more sub-policies is always beneficial to the agent. This is not a general phenomenon and is, for example, untrue in the options framework[46].

Secondly, the compositionality of the LMDP allows policies to simply be rescaled by element-wise scalar multiplication. This provides a simple way to fine-tune the relative importance of tasks when behaviours are not aligned, or rewards are unequal. Complex behaviours can begin to emerge as constituent policies are re-weighted. As can be seen in Fig.(3.1 - Linear Composition), the composed value function exhibits qualitatively different behaviours as the relative importance of the constituent policies are adjusted. Here, however, the composed policy nevertheless seems to admit some behavioural symmetry. This is an artefact of the underlying symmetry in the transition dynamics. When this symmetry is removed, the emergent complexity of compositional behaviours is more apparent. In Fig.(3.1 - Non-uniform Dynamics) we consider an agent operating in the same 1-dimensional corridor, but now the transition dynamics are left-biased with a non-zero drift velocity. The transition probabilities of the underlying Markov chain are $\texttt{LEFT} = 6/9$, $\texttt{STAY} = 2/9$, $\texttt{RIGHT} = 1/9$. In the general setting, the resultant compositional behaviours are rich and varied.

Finally, note that the LMDP addresses a pervasive concern for transfer learning methods in general, namely that of negative transfer. The guaranteed optimal composition of the LMDP immedi-

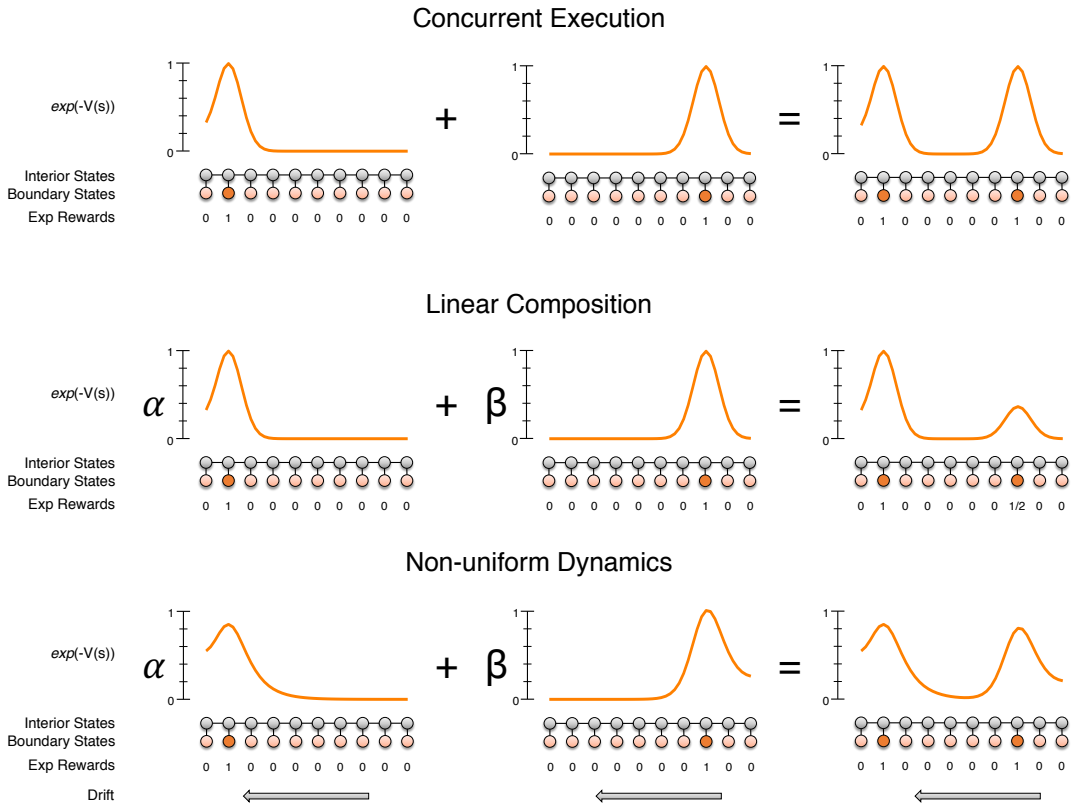


Figure 3.1: Compositional policies exhibit complex behaviours in the presence of non-uniform dynamics and weighted blends. Concurrent Execution: a linear combination of the exponentiated value functions yield an optimal policy when the boundary reward for the new task is a linear combination of the exponentiated boundary reward of the known tasks. Linear Composition: the relative importance of constituent tasks is easily adjusted through linear scaling of the exponentiated value function. This scaling affects global elements of the policy, and may lead to complex emergent behaviours in the agent. Non-uniform Dynamics: by composing the exponentiated value function, the task basis captures elements of both task specific information, in the form of the reward function, and environmental information, in the form of the passive transition structure. Here we consider the compositional policy resulting from a non-uniform underlying transition dynamic in which transitions are left-biased.

ately implies that no nonnegative transfer occurs, as the optimal policy is immediately realized.

3.2.3 A Basis for Action

The fact that policies may be executed concurrently, and may be composed through linear scaling allows us to naturally adopt the notion of a *task basis* - a library of known optimal policies from which new optimal polices can be constructed. If the task library is sufficiently rich, it constitutes a bona fide basis, in the mathematical sense, for the task space.

Proposition 1 *The task-library can be a basis for the task space.*

In a state space with N states, there are clearly N unique tasks with a singleton goal state. Suppose that we have for each of these tasks, the corresponding reward vector $\mathbf{r}^i$ and optimal desirability function $\mathbf{z}^i$. The exponentiated reward vectors for each task are then $\mathbf{q}^i = exp(\mathbf{r}^i) = \alpha^i \mathbf{e}_i$, where $\alpha^i = exp(r^i_i)$, the i^{th} component of the vector $\mathbf{r}^i$. Collecting the exponentiated rewards into a matrix we have:

$$\mathcal{Q}_{\mathcal{B}} = [\mathbf{q}^1, \ldots, \mathbf{q}^N] = [\alpha^1 \mathbf{e}_1, \ldots, \alpha^N \mathbf{e}_N].$$

Notice that a trivial rescaling yields the identity $I \in \mathbb{R}^{N \times N}$; the canonical basis. In particular, for any exponentiated reward vector $\mathbf{q}$, we have $\mathbf{q} = \mathcal{Q}_{\mathcal{B}}\mathbf{w}$ where $\mathbf{w} = diag(1/\alpha)\mathbf{q}$.

It follows from Eqn.(3.2) that the corresponding optimal desirability function is then $\mathbf{z} = Z\mathbf{w}$.

All possible tasks in the space, which may be identified by their boundary reward vectors $\mathbf{r}^k_{\mathcal{B}} \in \mathbb{R}^{N_b}$, therefore lie in the span of the columns of $\mathcal{Q}_{\mathcal{B}}$. Importantly this implies that by knowing just

N policies, the agent is immediately able to perform any task in the domain optimally with no further learning required; including, of course, all 2^N possible prototypical tasks. This is no small feat.

The ability to think of a basis for the task space has a host of additional benefits. For example, any library of tasks with the same span is functionally equivalent in this setting. With this in mind, the task library can be periodically pruned to remove redundant tasks, ensuring that the agent does not retain unnecessary information. This provides a natural way for thinking about optimal representations of the agent's experience. More generally, if an agent is faced with tasks drawn from some subspace of $W \subset \mathbb{R}^N$, the agent need only build and retain a task basis of $N_t = dim(W)$ policies whose corresponding reward vectors spans the subspace.

The structure of the task basis is particularly useful in that it formalizes a number of mechanisms by which important functions may be carried out. It defines how previous experience may be represented within the agent, and how new experiences might be integrated and distilled - namely by finding a minimal spanning set of policy vectors. In addition, the linear blending and parallel execution of policies provides a structured mechanism by which previous experience might be flexibly repurposed for new tasks.

The use of a shared basis for many tasks in a domain has certainly been explored in transfer learning before. These methods are predominantly parametric approaches which suppose that tasks can be represented by low-dimensional parameter vectors which can be modeled as a linear combination of shared latent model components[76,77,78]. Importantly, none of these methods guarantees nonnegative transfer, and certainly not direct optimal composition. As a paradigm, the spectral approach to transfer learning in RL has been reasonably successful. Notably, the method of proto-value func-

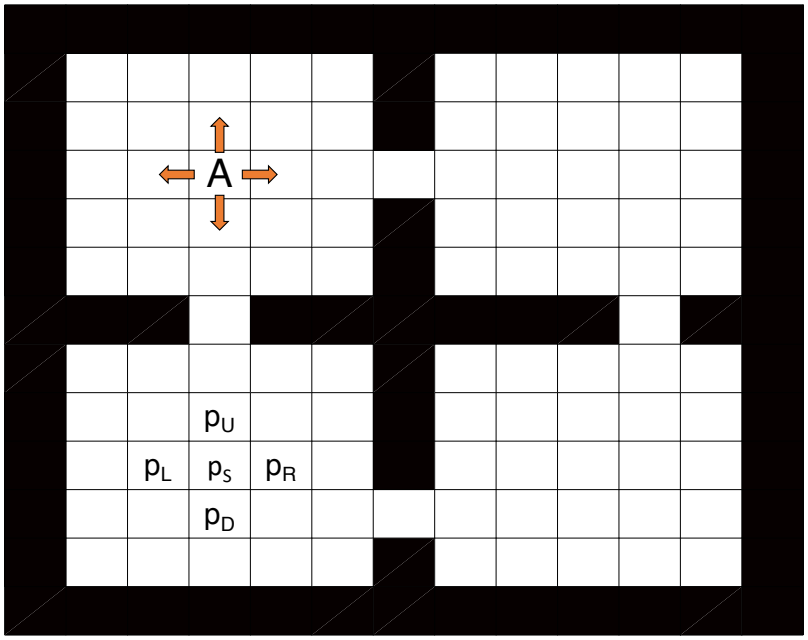


Figure 3.2: The 4-rooms domain: the prototypical discrete toy-environment. The domain is comprised of four identical rooms, each of which is a 5-by-5 grid. The rooms are connected to each other by a singleton doorway state. At each point in time the agent may move in one of the four cardinal directions - with probability p_U, p_D, p_L, p_R respectively, or remain stationary with probability p_S.

tions[76], which uncovers basis functions from a spectral analysis of a random-walk on the state-space graph, essentially captures task independent information in the form of the transition structure. This allows for useful transfer to occur even between tasks with very different goals - sidestepping somewhat the difficult problem of task-relatedness. In contrast our method captures elements of both the reward and transition information through the value function.

The execution protocol for the resultant multitasking method is described in Alg.(1). The method

is executed in two phases. In the first phase the agent constructs an improved initial policy by querying an external memory bank containing a representation of the library tasks. By attempting to represent the boundary reward structure for the new task in terms of the boundary reward structures for the tasks in the library, the agent uncovers a blend of library policies which it uses to initialize its behaviour on the new task. In the second phase of learning the agent makes incremental stochastic fine-tuned improvement to its policy. When the task library contains a complete basis for the task space, the initialized policy is also the optimal policy.

Algorithm 1 Off-line Multitask LMDP

Require: $Z, \mathcal{Q}_{\mathcal{B}}, \mathcal{S}_{\mathcal{B}}$

1: procedure MLMDP($\mathbf{r}_{\mathcal{B}}, \mathcal{N}$) # Given boundary reward $\mathbf{r}_{\mathcal{B}}$, and maximum step count $\mathcal{N}$
Initialize $\mathbf{z}$ *as blend of basis functions*
2: $\mathbf{q}_{\mathcal{B}} = exp(\mathbf{r}_{\mathcal{B}})$
3: $\mathbf{w} \leftarrow min_{\mathbf{w}} ||\mathcal{Q}_{\mathcal{B}}\mathbf{w} - \mathbf{q}_{\mathcal{B}}||_2$
4: $\mathbf{z} = Z\mathbf{w}$
Standard z-iteration loop
5: for $k \leftarrow 1$ to $\mathcal{N}$ do
6: $\mathbf{u} \leftarrow \mathcal{N}(\mathbf{p} \odot \mathbf{z})$ # Obtain normalized controlled dynamics
7: $(q_s, s') \sim \mathbf{u}$ # Draw next state
8: $z_s = \alpha z_s + (1 - \alpha) q_s z_{s'}$ # Perform stochastic update
9: if $s' \in \mathcal{S}_{\mathcal{B}}$ then
10: **Break** # Terminate at boundary states
11: end if
12: end for
13: return $\mathbf{z}$ # The optimal desirability function
14: end procedure

In order to demonstrate the new method in practice, we consider a simple multitask problem in the canonical 4-rooms domain. The agent is tasked with reaching states in the top-left, the bottom-

right corner, the bottom-left corner, and the top-right corner. The tasks are presented in a fixed order, and the agent is permitted to learn for 500 epochs of 100 steps, with early-exits on boundary state transitions. Results are presented in Fig.(3.3), with further details in Appendix.(A.3).

We have thus far considered the situation in which the target task is expressible as an exact linear combination of a library of source tasks. When this is possible, the LMDP formalism permits an exact optimal composition and no further learning is required. More generally however, one can imagine a library of tasks which is not an exact basis for the task space. This situation may arise in the incremental learning setting, in which the agent has only experienced tasks in some region of the state-space for example. Nevertheless, we would like our agents to benefit from this experience.

3.3 Inexact Composition and Optimal Subspaces

Our study of inexact composition is motivated by the fact that a full basis for the task space is, in practice, often not available. This may be due to the fact that the agent simply does not yet have the requisite experience (as in the incremental learning paradigm), or simply because the memory overhead of storing a full basis is too great. To be sure, the compositional nature of the LMDP allows for an incredibly compressed representation of the control space: requiring just N policies in order to optimally execute 2^N tasks. Indeed, the fact that the task library can be considered a basis for the task space shows that this is in some sense an optimal representation. Nevertheless, a full task basis requires that we store $\mathcal{O}(N^2)$ elements, which may be infeasible for large state spaces. This follows from fact that we require an explicit N-dimensional vector representation for each of the N basis

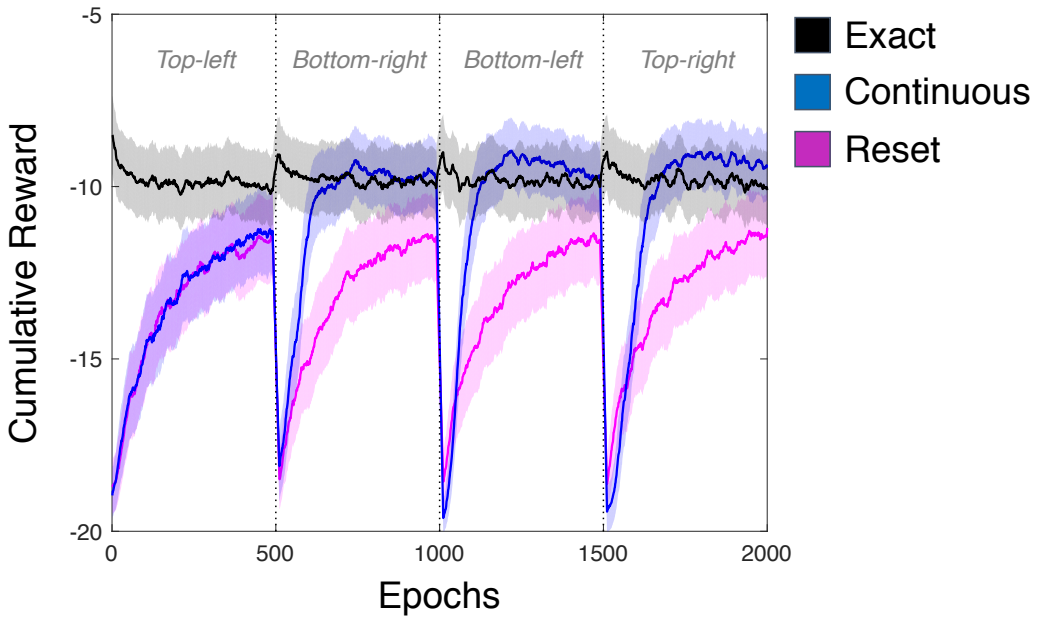


Figure 3.3: The offline multitask LMDP. Here we consider a multitask problem set in the canonical 4 rooms domain Fig.(3.2). The agent is faced with 4 navigation tasks presented incrementally. The agent must first navigate to the top left corner, then the bottom right corner, then the bottom left corner, and finally the top right corner. These tasks are presented to the agent every 500 epochs in the predefined order. As the new task is presented, the agent is provided the new boundary reward structure, and may elect to reinitialize its estimate of the desirability function for the target task; after which z-learning proceeds as usual. Exact Composition: when the agent is equipped with the full task basis, it is able to initialize its estimate of the desirability function to the optimal value. The minor performance drop in the first few epochs is due to the fact that the stochastic updates of the z-iteration procedure actually caused the optimal policy to degrade slightly. Performance is otherwise immediately optimal. Continuous Learning: in this implementation the agent simply rolls over its estimate of the desirability function when the new task is presented. This leads to worse than random initial performance (as can be seen most clearly for tasks 3 and 4), but given the simple structure of the domain, the agent is nevertheless able to leverage some previous experience to aid future learning (e.g. to move towards doorways from adjacent rooms). In general this does not hold; initializing the agent with an optimal policy for one task may well lead to worse performance for a second task than a blank slate initialization would. Reset Initialization: This implementation corresponds to the naive 'flat' agent that simply restarts learning on the new task with a blank slate. Learning is significantly slower here.

tasks.

A moment's consideration yields a simple, but important, observation - the choice of library tasks ought to be dependent on the distribution of targets tasks. Notice that an exact composition is possible precisely when $Q_{\mathcal{B}}\mathbf{w} = \mathbf{q}^t_{\mathcal{B}}$, and so, if the agent will only face tasks with boundary rewards in some specific region(s) of the state-space, the task library need only store the corresponding basis elements. This observation highlights the deep task-dependent nature of the composition, and hints at a more general framing of the compositional problem. Instead of requiring that new tasks be exactly represented by the library tasks, we instead seek an approximate representation; and a mechanism for translating the quality of the approximation to the expected value of the transfer.

Relaxing the assumption that an exact task basis is available, the problem may be naturally stated as:

$$min_w ||\hat{Q}_{\mathcal{B}}\mathbf{w} - \mathbf{q}^t_{\mathcal{B}}||_2 \quad \Rightarrow \quad \mathbf{z}^t \approx Z\mathbf{w}, \tag{3.3}$$

where the hat notation $\hat{\cdot}$ represents the incomplete 'basis'. The relationship between the approximation to the boundary reward and the task basis is not immediately obvious. To better understand how these are related, recall from Eqn.(2.12) that the desirability function may be written in closed form as:

$$\mathbf{z}_{\mathcal{I}} = M\mathbf{z}_{\mathcal{I}} + N\mathbf{z}_{\mathcal{B}}, \tag{3.4}$$

where $M = P_{\mathcal{I}\mathcal{I}} diag(\mathbf{q}_{\mathcal{I}})$ and $N = P_{\mathcal{I}\mathcal{B}} diag(\mathbf{q}_{\mathcal{I}})$. The closed form solution is then obtained as:

$$\mathbf{z}_{\mathcal{I}} = \Lambda\mathbf{z}_{\mathcal{B}} = \Lambda\mathbf{q}_{\mathcal{B}}, \tag{3.5}$$

where $\Lambda = (I - \mathcal{M})^{-1} N$. If we suppose that the exponentiated boundary reward is constrained to lie within $span(\hat{Q}_{\mathcal{B}})$, such that $\hat{\mathbf{z}}_{\mathcal{I}} = \Lambda \hat{Q}_{\mathcal{B}} \mathbf{w}$, we may formulate a new optimization problem as follows:

$$min||\mathbf{z}_{\mathcal{I}} - \hat{\mathbf{z}}_{\mathcal{I}}||_2 = min_w||\Lambda \mathbf{q}_{\mathcal{B}} - \Lambda \hat{Q}_{\mathcal{B}} \mathbf{w}||_2. \tag{3.6}$$

The explicit closed form solution may be written in terms of the pseudo-inverse as:

$$\mathbf{w} = \left(\left(\Lambda \hat{Q}_{\mathcal{B}} \right)^T \Lambda \hat{Q}_{\mathcal{B}} \right)^{-1} \Lambda \hat{Q}_{\mathcal{B}} \Lambda \mathbf{q}_{\mathcal{B}}. \tag{3.7}$$

When $\Lambda = I$, the solution collapses to the solution of Eqn.(3.3). Since Λ represents the steady-state dynamics, in practice Eqn.(3.7) means that the approximation to the exponentiated boundary reward results in an approximation to the desirability function that is progressively more degraded as the passive transition dynamics become more diffuse.

A critical property implicit in the guaranteed optimal composition of the LMDP is that of non-negative transfer - if the new task can be represented as a linear combination of previous tasks, then the agent is guaranteed to immediately perform optimally. We find that a similar statement is possible for the approximate formulation given in Eqn.(3.3). First, note that the LMDP specifies a class of MDPs for which the minimization over the control space is convex[56,45] (see Section.(2.3) for details).

In the standard optimization parlance, for some arbitrary objective function $J(x)$, we define the minimization error at iteration k to be $J(x^k) - J(x^*) = \epsilon^k$. Convergence rates are then stated in terms of this error. Results of this form are typically stated either in terms of the number of iterations

required to decrease the error below some fixed threshold , i.e. $min \quad k \quad s.t. \quad J(x^k) \leq J(x^*) + \delta$, or equivalently, the error threshold δ that is achievable for a fixed number of iterations k. It follows from the convexity of the Eqn.(3.3) that the minimization error is bounded above as follows:

$$\epsilon^k \leq \frac{c}{k}||x^* - x^\circ||^2, \tag{3.8}$$

where $c \in \mathbb{R}$ is a problem specific constant which depends on the Lipschitz continuity of the gradient of the objective[79]. The error bound in Eqn.(3.8) implies that for a fixed error threshold δ, the maximum number of iterations required to achieve an error below this value is monotonically reduced by improving the initial guess. Specifically, this implies that as the initial guess for $\mathbf{z}_\circ^t$ is improved, as measured by the normed error $||Z\mathbf{w} - \mathbf{z}^t||$, the maximum number of iterations to achieve ϵ-convergence is monotonically reduced.

The above result explicitly couples the approximation error, and by extension the task basis, with the convergence rate. This effectively constitutes a form of guaranteed non-negative transfer, since we are able to measure the approximation error *a priori* as the vector-norm. When a new task is encountered, we may compare the approximation error obtained with a representation within the task library, to a random initialization and choose the better approximation to initialize the learning problem on the new task. The error bound stated in Eqn.(3.8) holds when the gradient may be computed exactly. In our setting the gradient is not computed exactly, but may nevertheless be arbitrarily well approximated by sufficiently decreasing the learning rate.

This convergence bound is investigated numerically in Fig.(3.4). We consider a setup in which

the agent must repeatedly solve some fixed task in the domain over multiple trials. Each trial constitutes a full course of learning over multiple epochs. At the beginning of each trial, the agent is initialized with a random policy $\mathbf{z}^{\circ}$ drawn from the weighted sum $\lambda \mathbf{z}^{*} + (1 - \lambda)\hat{\mathbf{z}}$, where the interpolation parameter $\lambda \in (0, 1)$ is drawn uniformly at random, the random policy $\hat{\mathbf{z}}$ has elements drawn uniformly at random and suitably scaled, and $\mathbf{z}^{*}$ is the optimal policy for the task. The agent then proceeds to solve the task in an online fashion, by iterative drawing next-states s', from the controlled distribution defined in Eqn.(2.13), and performing a stochastic update to the policy as $z_s = \alpha z_s + (1 - \alpha) q_s z_{s'}$. The learning rate $\alpha \in (0, 1)$ is slowly decreased at the end of each epoch until the estimate for $\mathbf{z}$ stops changing (see Section.(2.3.1) for more details on the z-iteration procedure). For each trial we extract the initialization error $||\mathbf{z}^{*} - \mathbf{z}^{\circ}||_2$, and the average number of steps required to hit a goal state over the course of learning. See Appendix.(A.4) for more details on the experimental setup.

The linear relationship between the initialization error and the convergence rate predicted by Eqn.(3.8) is clearly evident in Fig(3.4). The breadth of the variance shown is primarily a function of the inherent stochasticity of the optimal policy and the predefined maximum number of steps the agent can take in a single epoch. Note that the slight deviation from linear behaviour at the boundaries is in fact an artefact of the moving average smoothing procedure. The value at these boundary states are averaged over a smaller number of samples, and as such the values on the left end are not up-weighted by higher valued sampled to the right producing the illusion of a down-tick. Similarly values on the right end are not down-weighted by lower valued samples to the left, producing the illusion of an uptick.

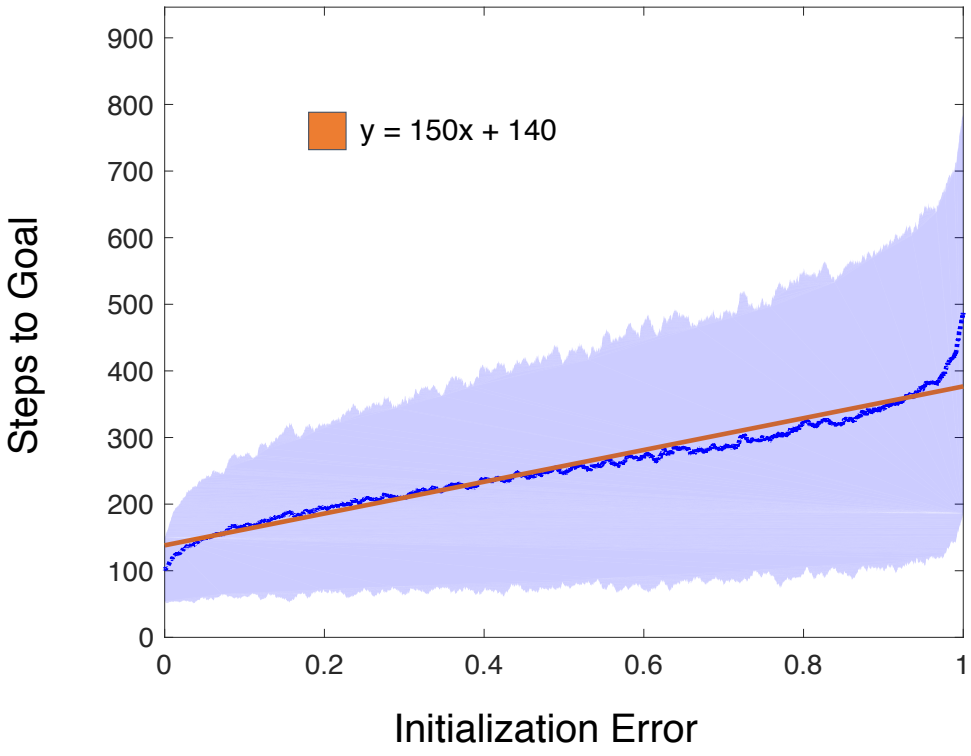


Figure 3.4: Convergence rate improves monotonically with improved initialization. Each individual data point corresponds to a full episode. For each episode the agent is initialized at a random state $s \in S$ with a random policy $\mathbf{z}^{\circ}$, and must solve some fixed task. Each episode comprises 50 epochs, each with a maximum step count of 1000. The normalized initialization error $||\mathbf{z}^{*} - \mathbf{z}^{\circ}||_{2}$ is plotted against the mean step count, with the shaded area corresponding to one standard deviation. A linear relationship is strongly evidenced, and aligns with the theoretical prediction. The gradient of this linear relationship is a function of the Lipschitz continuity of the objective function, and the intercept is a function of the inherent stochasticity of the optimal policy, mediated by the temperature parameter.

In addition, the convexity of the LMDP formalism provides some traction on a few important general concerns for transfer learning methods: specifically the problems of *task similarity* and *source task selection*.

The question of task similarity is concerned with how the new target task is related to the library tasks. The non-negative transfer result above suggests a specific metric for assessing task similarity as the normed error $||\hat{\mathcal{Q}}_{\mathcal{B}}\mathbf{w} - \mathbf{q}^t_{\mathcal{B}}||$ by coupling this metric with the convergence rate. Inversely, this similarity metric provides a useful measure for the quality of the task library by quantifying the extent to which target tasks can be represented in terms of library tasks. It is worth noting that this explicit metric is not available for most transfer methods which typically rely on proxy measures of task similarity[51]. The need for general purpose task similarity metrics to allow for robust transfer has certainly been noted by other authors[80], but they have remained elusive given the complex interplay between reward and transition dynamics in the nonlinear setting. This inability to quantify task similarity and the efficacy of transfer is at the heart of the problem of negative transfer.

Source task selection is concerned with how library tasks come to be in the first place. The problem is usually phrased either in terms of how library tasks are selected from an available set of tasks, or how they are distilled from experience. Tacit in the former phrasing, is the assumption that having too many library tasks may in fact be detrimental to the agent's performance, and so task selection is crucial. However, the non-negative transfer results imply that this assumption does not hold in the LMDP formalism - that is to say that having more tasks is necessarily no worse for the agent's performance than having fewer. This follows from the fact that constituent policies are run concurrently, and therefore do not require the agent to choose between a large set of policies, many of

which are harmful to the task at hand. Separately, in the latter phrasing, we suppose that the representational capacity is restricted due to external constraints. In this case a choice must be made as to how the agent's experience should be distilled and represented. Supposing that we have a fixed memory capacity and can store only $l < N$ policies in the library, it follows from the results above that we should choose to retain the l policies which are maximally able to represent new target tasks. Said differently, the representative tasks should capture the maximum variance in the distribution of new tasks.

When the new tasks are uniformly distributed through the state space, this corresponds directly to a decomposition of the desirability basis Z via principle components decomposition analysis (PCA). Of course, this approach is likely to succeed only when there exists some latent structure in the Z matrix. Fortunately, for many domains of interest, there is good reason to believe this is the case. Since the desirability matrix Z is a function of both the boundary reward structure and the passive transition dynamics, it reflects any additional structure in the environment, and so, for structured domains, it is likely that some low dimensional representation is available. If the target tasks are not uniformly distributed through the state space, a maximally variance capturing subspace is nevertheless easy to obtain. Further exploiting the compositionality of the LMDP, we note that since each individual task may be represented as $\mathbf{z}^i = Z\mathbf{w}^i$, the statistics of any distribution of tasks may be captured by a simple rescaling of the basis $\tilde{Z} = Z\ \mathrm{diag}(\tilde{\mathbf{w}}) = Z\left(\mathbb{E}_{i\sim\mathcal{D}}\left(\mathbf{w}_i\right)\right)$, where tasks are drawn from some distribution $\mathcal{D}$. The new desirability basis $\tilde{Z}$ is simply a modified version of the original basis Z, whose columns have been reweighted proportionally to how frequently they appear in the specific task distribution $\mathcal{D}$. When a given task appears more frequently, it is weighted

more heavily, and by simply applying PCA to the rescaled basis $\tilde{Z}$ a maximally variance capturing subspace for the specific task distribution is uncovered.

The problem of source task selection has thus been reduced to the problem of finding a maximally variance capturing subspace to the desirability basis Z. Note that the exponential transform which allowed us to linearize Eqn.(2.13) implies that all of the components of Z are non-negative. This fact implies an additional constraint on the matrix decomposition factors. With this additional constraint in mind, the source task selection problem may be stated as follows:

$$min_{D,W\geq 0}||Z - DW||_p, \tag{3.9}$$

for some *data*-matrix $D \in \mathbb{R}^{n\times l}$, some *weight*-matrix $W \in \mathbb{R}^{l\times n}$ and some matrix norm p. The constraint $D, W \geq 0$ may instead be written as $d_{ij}, w_{ij} \geq 0$ and requires that each component of the matrix factors is non-negative. The minimization problem stated in Eqn.(3.9) is commonly known as non-negative matrix factorization (NNMF) and has been studied extensively for its broad application to problems in interpretable dimensionality reduction[81]. Different choices for the matrix norm p yield qualitatively different solutions. When p is taken to be the Frobenius norm and the weight matrix is constrained to be orthogonal, the NNMF is equivalent to a form of k-means clustering, with the cluster centers specified by the columns of the data-matrix, and the cluster membership indicators specified by the rows of the weight-matrix[82].

In order to find an approximate solution to the problem $Z \approx DW$, some cost function must be specified. One candidate for the cost function is the matrix p-norm used in Eqn.(3.9) above. How-

ever, when the column vectors in the target matrix Z correspond to distributions, a more general class of cost functions are often used. In particular, the new source task selection problem can be stated as:

$$min_{D,W\geq 0} d_\beta(Z||DW), \tag{3.10}$$

where d_β denotes the β-divergence[83,84]. This class of divergences is commonly used in information theory to compare distributions. Importantly, unlike the matrix p-norm, β-divergences are not metrics since they are not symmetric with respect to their inputs. Nevertheless they are a common choice for the cost function for NNMF procedures over distributions since they reduce to more common divergences for specific values of β. For instance when $\beta = 0$, the divergence reduces to the Itakura-Saito divergence. Similarly, when $\beta = 2$ we recover the standard Euclidian distance above[85]. When $\beta = 1$ the divergence reduces to the generalized KL-divergence[86,85] and the problem is equivalent to probabilistic latent semantic analysis[87], which is a form of multinomial PCA. Explicitly we have:

$$D_1(A||B) = D_{KL}(A||B) = \sum_{ij}\left(a_{ij}\,\ln\left(\frac{a_{ij}}{b_{ij}}\right) - a_{ij} + b_{ij}\right). \tag{3.11}$$

Unless otherwise specified, we take the NNMF to be carried out under the cost function in Eqn.(3.10) with $\beta = 1$ corresponding to the generalized KL-divergence. This choice aligns both with our interpretation of the optimal subspace as being maximally variance capturing, and the specific form of the control cost taken in the LMDP formulation.

Although a variety of algorithms for NNMF are employed on a plethora of widely varying prob-

lems, the setup in Eqn.(3.9) may in fact be ill-posed. When the entries of the target matrix are strictly positive $z_{ij} > 0$, no unique solution exists and the various solutions are non-trivially different[88] (for example, they are not related via some permutation). Nevertheless, the interpretability and practical application of the resulting decompositions has kept the technique broadly in use.

In much the same way that the columns of Z correspond to individual basis policies, so the columns of the data matrix D may be interpreted as corresponding to a set of distilled policies. In order to interpret the distilled policies produced by the NNMF, it will be useful to recover their implied boundary reward structure. Assuming that the boundary transitions are uniform, this may be computed in closed form. As shown in[63], the linear Bellman equation can be written in the following form:

$$\begin{aligned}
\mathbf{z} &= GP\mathbf{z}, \\
\begin{bmatrix} \mathbf{z}_{\mathcal{I}} \\ \mathbf{z}_{\mathcal{B}} \end{bmatrix} &= \begin{bmatrix} \mathcal{M}, & \mathcal{N}, \\ 0, & I \end{bmatrix} \begin{bmatrix} \mathbf{z}_{\mathcal{I}} \\ \mathbf{z}_{\mathcal{B}} \end{bmatrix}, \\
(I - \mathcal{M})\, z_{\mathcal{I}} &= \mathcal{N} z_{\mathcal{B}}, \\
z_{\mathcal{B}} &= \mathcal{N}^{-1} (I - \mathcal{M})\, z_{\mathcal{I}},
\end{aligned}$$

where again, $\mathcal{M} = P_{\mathcal{I}\mathcal{I}} diag(\mathbf{q}_{\mathcal{I}})$ and $\mathcal{N} = P_{\mathcal{I}\mathcal{B}} diag(\mathbf{q}_{\mathcal{I}})$. It is then trivial to extract the exact

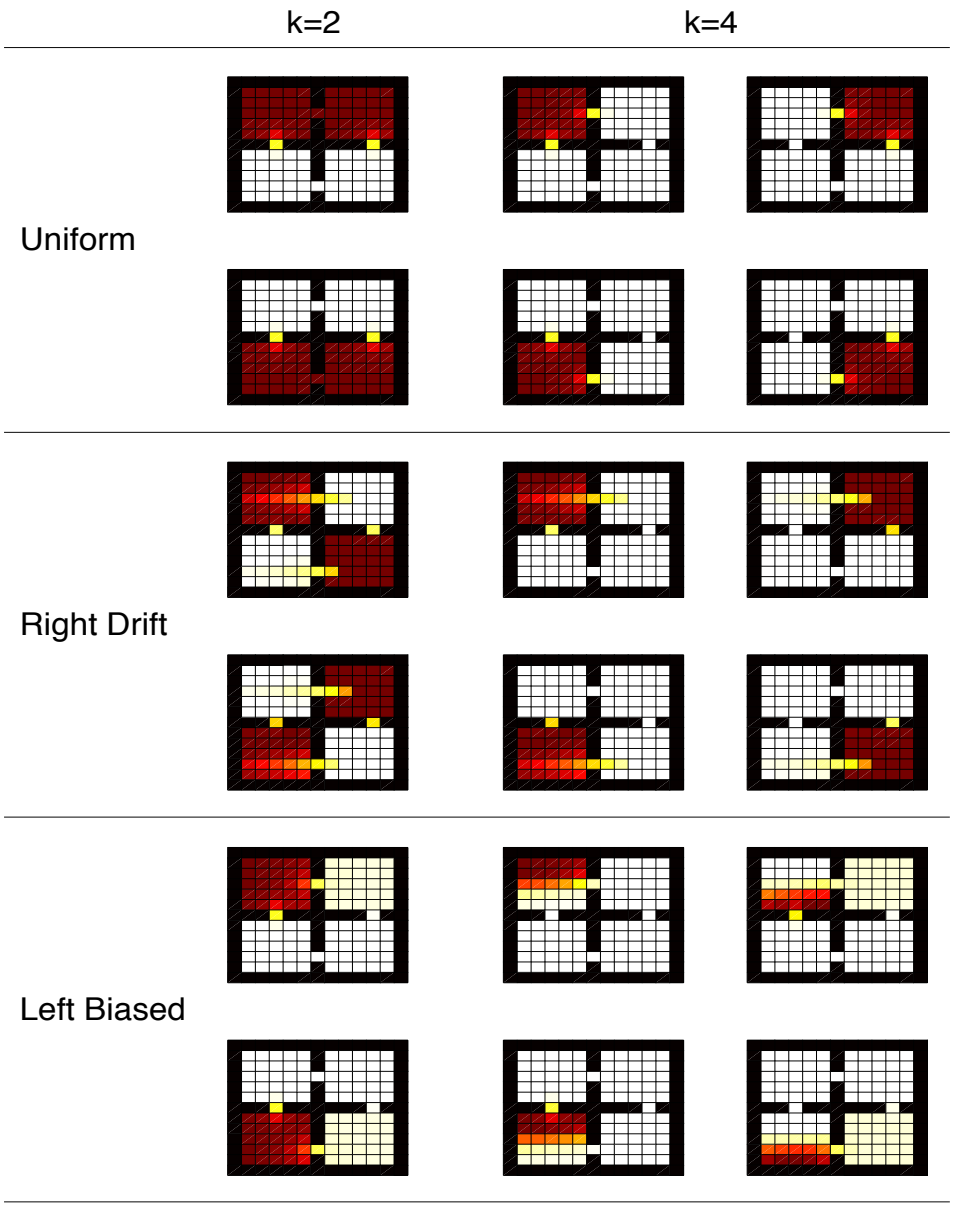


Figure 3.5: Intuitive decompositions: NNMF uncovers library tasks which capture the latent structure in the task basis Z. The distilled tasks, which we extract as the column vectors of the low-rank approximation D, correspond to entire rooms rather than singleton goal states. When the underlying transition dynamics contain a nonzero drift component, the symmetry breaking is reflected in the distilled policies in which the probability mass is notably shifted. When the tasks are not uniformly distributed through the space, we may nevertheless obtain a maximally variance capturing subspace, by simply reweighting the basis matrix.

boundary reward structure since, $\mathbf{z}_{\mathcal{B}} = \mathbf{q}_{\mathcal{B}}$. A cursory analysis of the distilled policies in Fig.(3.5) reveals that these tasks appear to have broader goal regions, rather than single goal states. This appears to be a general phenomenon when seeking out low rank approximations to the task basis [47]. Furthermore, the collection of distilled policies form an approximate cover for the task space.

These distributed policies permit an interesting interpretation. Instead of trying to reach some particular low-level state, the agent's behaviour is directed to a region of the state space, each configuration of which corresponds to a different way to achieve the same goal. In Fig.(3.5) the distilled policy places reward at any state in the first room. Semantically, this corresponds to the task 'get to the first room'. Remembering our would-be golfer for a moment, a distilled policy for the agent may correspond to any combination of joint angles which result in the club head being in the desired position. This analysis suggests that these distributed policies are best described in terms of high-level goals which specify no preference for how they are achieved at the lower level. This interpretation implies a powerful abstraction mechanism, and is pursued in the following chapters.

A sample of the distilled policies recovered through NNMF are presented in Fig(3.5). The full task basis $Z \in \mathbb{R}^{104 \times 104}$ corresponds to the matrix, each column of which is the optimal desirability function for a specific task with a singleton goal state. There are N such tasks, each one corresponding to placing a single boundary reward at an individual state. Three distinct situations are considered: in the first, both the transition dynamics and the task distribution are uniform; in the second, the task distribution is uniform but the transitions dynamics are biased so that the agent preferentially transitions to the adjacent state to the right; and thirdly, the transition dynamics are uniform, but the task distribution is such that goal states on the left half of the domain are 10-times

more likely than those on the right. In each case results are presented for two different values of the decomposition factors $k = \{2, 4\}$.

Notice that in all cases, the union of the distilled policies form an approximate cover for the space; while each policy individually corresponds to a broad distribution over states. This corresponds to the fact that $\texttt{span}(\hat{z})$ is a maximally variance capturing subspace. Moreover, as the decomposition factor increases, each constituent library task is required to cover a smaller region of the space, and thus exhibits more localized behaviour. When the transition dynamics and task distribution are uniform, the symmetries in the domain enforce symmetry on the decomposition. This can be seen in the decompositions presented in the first row of Fig.(3.5).

However, when there is asymmetry in either the task distribution or the underlying transition dynamics, this is reflected in the decomposition. Consider the case in which tasks with goal states on the left half of the domain are more likely than those with goal states on the right. This situation is depicted in the bottom row of Fig.(3.5). Notice how the probability mass of the distilled policies is shifted to the left for this particular distribution of tasks. This refactoring of the task library allows the agent to better represent those tasks with goal states on the left of the domain and in so doing minimizes the *expected* approximation error over tasks draw from the biased distribution. Similarly, when the underlying transition dynamics are biased such that transitioning to adjacent states to the right is more probable, the probability mass bleeds out to the right, allowing these new tasks to be better represented within the library.

An important observation is that this abstraction mechanism is internal to the agent, and can happen autonomously. As the agent explores, it may periodically distill this experience via Eqn.(3.9),

and in so doing uncover these abstracted behaviours. These representations can then be called upon to improve performance on future tasks via Eqn.(3.3). The design of such an 'online' process requires some nuanced handling, and is considered in Chapter.(4). There is also evidence of high-level behaviours emerging from compressed representations of previous experience in biological systems[89]. In a different but strongly related formalism, the so-called successor representation seeks to decouple reward and transition contributions to the agent's policies[71]. In so doing, the successor representation formalism also permits an explicit representation of the reward components for discrete states as a matrix (see, Eqn,(3.1)). The eigenvectors of this matrix are shown to exhibit a striking resemblance with grid cells observed in the entorhinal cortex[90].

Thus far we have assumed that the number of library tasks that can be stored l, is given *a priori* by an oracle or by some external consideration. Alternatively, we might ask what the best number of library tasks is to have in an attempt to optimally trade off storage capacity with expressivity. A valid, but somewhat naive, approach is to visually inspect the reconstruction error as a function of the number of stored tasks, and find the infliction point (often referred to as the 'elbow-joint' or 'knee' of the graph). The inflection points of the reconstruction error curve correspond to structural regularities in the domain. As an example, in a 4-rooms domain (Fig.(3.2)), the incremental value of increasing the library tasks from three to four is large, since we are better able to distinguish between each room of the domain (see Fig.(3.6)). Whereas the incremental value of increasing the number of library tasks from four to five, is significantly less for related reasons.

This is of course a somewhat unsatisfying solution. Fortunately, under some reasonable assumptions, a more theoretically grounded approach exists. Suppose that the basis matrix we are interested

in Z is observed as a noisy measurement obeying $Z = \hat{Z} + \sigma Y$, where Y is a noise matrix with zero mean and unit variance. Then in seeking a low-rank reconstruction to the ground truth signal Z, we need only keep the singular values of $\hat{Z}$ which are below the hard threshold $(4/\sqrt{3})\sqrt{n}\sigma$, where σ is the noise level[91] - or, when the noise level is unknown, simply below $2.858 y_{med}$, where y_{med} is the median singular value of $\hat{Z}$. Intuitively, singular values below this threshold are too noisy to be useful in reconstruction of the ground truth. Viewed through a biological lens, these noisy components may correspond to outlier experiences which are considered superfluous to the task of interest, and are quickly forgotten.

To probe this question computationally, in Fig.(3.6) we consider the reconstruction error to the full task basis for the 4-rooms domain. As expected, the reconstruction error is monotonically reduced as the decomposition factor is increased. The reconstruction error clearly undergoes two notable steps in which the error is significantly decrease by the inclusion of a single additional task. The first of these steps occurs at $k = 4$, and is an artefact of the strong structural regularities in the domain. The second step occurs at $k = 12$ as is precisely predicted by the hard thresholding result in[91], following from the fact that singular values are such that $\sigma_{13} < 2.858\sigma_{52}$. In either case, the ability to explicitly couple the reconstruction error with the convergence rate allows for a powerful analytic treatment of the question of how the agent should synthesize and discard information as it accumulates experience.

In Fig.(3.7) we investigate the performance of an agent equipped with a task library of varying size. The agent is once again tasked with navigating to the 4 corners of the domain, with each task being presented incrementally after 150 epoch. As before, when a new task is presented the agent

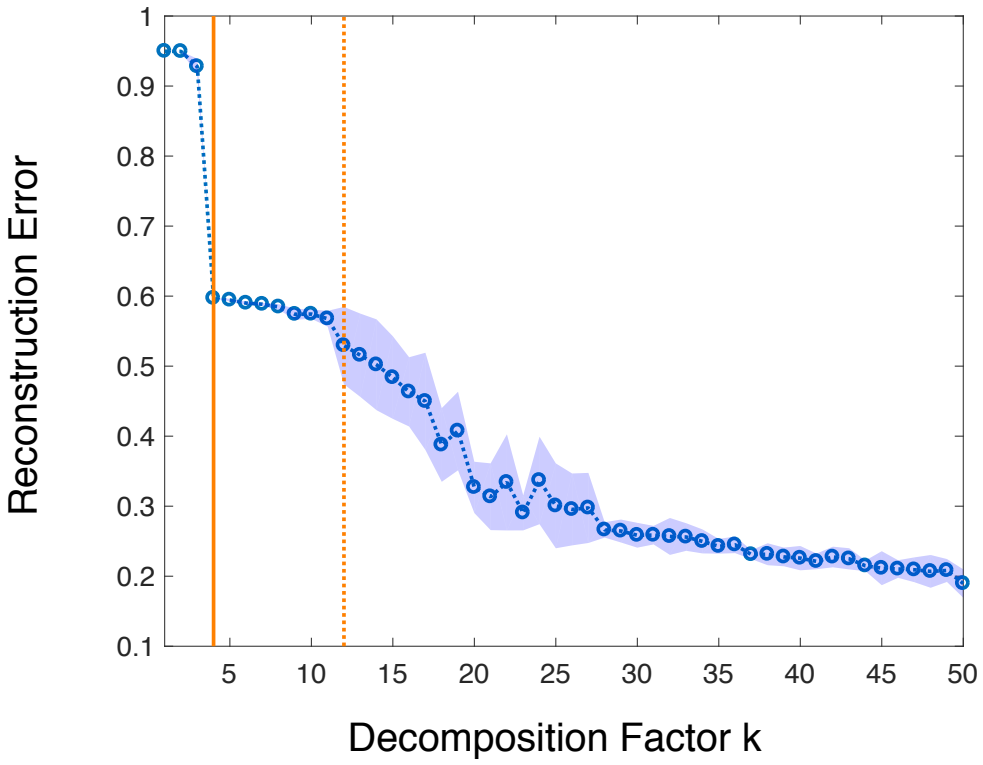


Figure 3.6: The trade-off between expressivity and storage capacity can be assessed by considering the reconstruction error. As expected, the reconstruction error decreases monotonically as the decomposition factor increases. The step-wise improvement in the reconstruction error occurs at two points: the first corresponding to structural regularities in the domain $k = 4$, and the second corresponding to low-information directions $k = 12$. These low-information directions are analytically predictable as a function of the distribution of singular values.

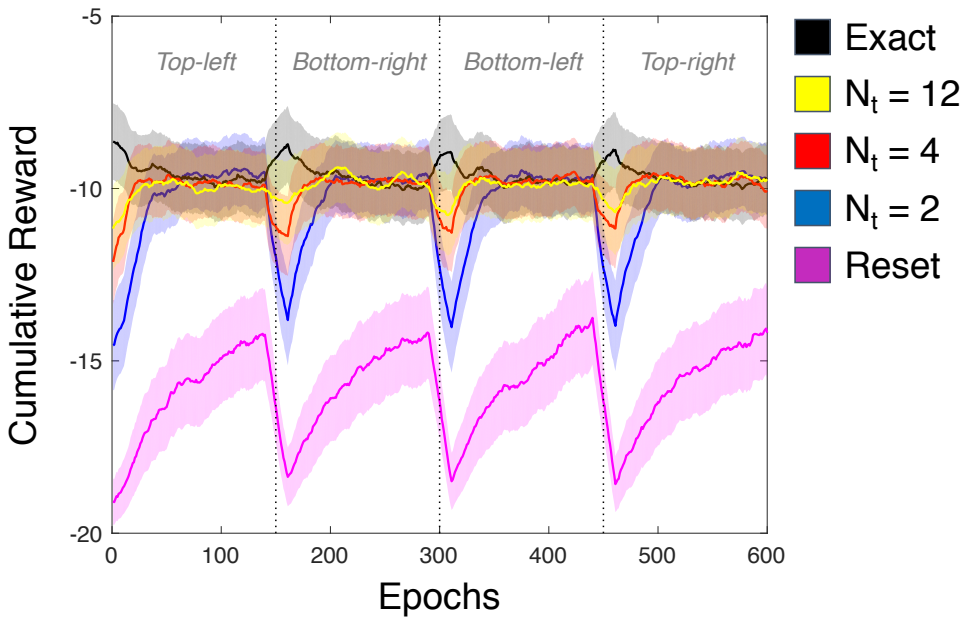


Figure 3.7: Approximate offline multitasking. In line with theoretical prediction the agent's performance is monotonically improved as the number of available subtasks is increased. The agent receives a significant boost in performance even with a very modest subtask library. This is due to the strong structural regularities in the domain. Notice, for example, that the optimal polices for reaching any state (or set of states) in the first room, are identical at all states not in the first room. As such, equipped with a task library of just 4 subtasks, the agent is initialized with a policy that is optimal at $3/4$ of the states, and the agent need only explore in the final room.

re-initializes its policy by finding an approximate representation of the new task as a distributed representation within the task library. As expected, the agent's performance is monotonically improved as the number of available subtasks in the library is increased. What is somewhat surprising however, is just how significant a jump-start the agent is able to received even with a modestly sized subtask library.

Taking stock: the inexact composition described in Eqn.(3.3) extends the applicability of the transfer mechanism to situations in which an exact task basis is not available, or the target task cannot

be precisely represented by the library tasks. This extension is critical for exploiting the property of composition in large state spaces, where storing $\mathcal{O}(N^2)$ elements is prohibitive. It is also important for the continuous implementation of the transfer procedure. In practice we would like for our agents to be able to utilize their experience to-date, whatever that may be, rather than requiring a complete description of the task space before value can be derived from transfer. By exploiting the convexity of the problem formulation, we are able to derive a non-negative transfer result in the inexact setting. In so doing, we established both a general metric for task similarity as the vector-normed approximation error, and a measure of the quality of a set of library tasks in terms of their ability to represent a set of target tasks.

By optimizing for the expressivity of the task library, we uncovered a mechanism for incorporating and distilling the agent's experience - retaining novel information, while discarding experiences that were similar to what the agent had seen before. This distillation process constitutes a continuous refactoring of the agent's internal representation. Moreover, we were able to define a measure for the incremental value of increasing the agent's representational capacity, and utilize this measure to suggest how the agent may make effective tradeoffs between efficiency and expressivity.

Finally, by interpreting the distilled policies as specifying complex distributions over states, rather than single goal states, we were able to describe the internal representation as corresponding to high-level task semantics. In the following section we discuss the general semantics provided by the specific form of compositionality we consider, and discuss the limitations thereof.

3.4 Semantics and Limitations of Task Composition

As demonstrated in Figs.(3.1, 3.5), the behaviours resulting from compositional policies can be complex, and difficult to describe. However, the effort to do so is well justified. Developing a language around the resulting behaviours will allow us not only to better describe the emergent phenomena exhibited by compositional policies, but also, inversely, to articulate in a suitable way the behaviours we would like our agents to exhibit. As such our new language has both a descriptive and constructive role to play.

Throughout this chapter we have been concerned with the undiscounted absorbing-state formulation of the RL problem. This particular setup endows the compositional policy with a natural `OR` semantic. That is to say that when our agent is operating under a policy composed of two constituent policies, and is thus seeking two sets of goal states, the agent is behaving as though directed to achieve either the one *or* the other of the constituent tasks. Note that this interpretation follows when there is little or no intersection between the boundary reward states of the constituent tasks (as is the case in the experiments we have considered thus far). When there is overlap between the boundary rewards of the constituent tasks, the composition is such that the agent will prefer states that have high value under all many. In some sense this corresponds to an *intersection of constraints* and an `AND` semantic.

At any point in time, the agent is driven to achieve the particular constituent task with the lowest expected cumulative cost. This single metric is derived from the state-cost, control cost, and boundary reward. While these factors are difficult to prise apart in general, in the simplest case a very spe-

cific interpretation is permitted. When the passive transition structure for the underlying Markov chain is uniformly random, the state-costs are all equal, and the boundary rewards are equal to each other in magnitude but otherwise disjoint, then the compositional policy corresponds to reaching the `NEAREST` goal state, as measured by number of discrete transitions.

In the strict sense, the `OR` semantic is the only one permitted by this form of composition, since boundary states are absorbing. However, if we are willing to slightly extend our execution paradigm, we are able to do a little better. To extend our execution paradigm, we briefly adopt the earlier meaning of task composition from Singh [92] to describe the temporal chaining of policies, rather than the concurrent execution of policies. In this setup the agent executes some policy until it terminates. The agent then continues the episode from the terminal state, operating under some new policy. In our case this corresponds to terminating just one of the constituent policies when the agent enters an absorbing state, and allowing the agent to continue thereafter.

Suppose that the agent is following a policy comprised of N_t constituent policies. The execution paradigm described above allows the agent to first pursue the nearest of the N_t targets. Once the agent has reached a goal state of the ${N_t}^{\text{th}}$ task, that task is terminated, and the agent is allowed to continue by following the new compositional policy with the ${N_t}^{\text{th}}$ constituent removed. The agent will then pursue the nearest of the $N_t - 1$ targets. In this way, our agent is immediately able to perform a greedy version of the traveling salesman problem. This particular execution mode endows the agent with an `AND` semantic, which we understand to mean that the agent can be made to solve a chained sequence of tasks albeit in a necessarily greedy fashion.

Thus far we have already accrued two powerful task semantics. Ideally, we would be able to de-

velop a full set of operative coordinating conjunctions in the task semantics. If this were possible, the compositional nature of the semantics would essentially permit a task algebra, allowing us to specify arbitrarily complex behaviours by manipulating the conjunctions. As an example, a task might then be specified by requiring that the agent visit $(A - \texttt{OR} - B) - \texttt{AND} - C - \texttt{AND} - (\texttt{NOT} - D)$. The sort of general composition described here is beyond the scope of this work, but has recently been pursued by other authors[93,94].

The multitasking agent we have described is capable of rapid adaptation to changes in the boundary reward structure, as new tasks are presented. However, this rapid adaptation is limited to precisely those changes in the boundary reward structure. When changes are made to either the internal reward structure, or the transition dynamics, the agent's adaptation can be slow.

We began this chapter by showing how the LMDP formulation allowed for compositional policies to be constructed through the re-weighting, and concurrent execution of multiple constituent policies in parallel. This property of the LMDP allows for the guaranteed optimal compositional of policies. This implies a form of zero-shot learning, and guaranteed non-negative transfer in the meta-learning and transfer learning parlance respectively. Extending this analysis, and further exploiting this special property of the formulation, we described a general method for assessing task similarity, and defined a mechanism for robust transfer between the target task and a set of source tasks even when an exact composition is not possible. We showed that a similar form of non-negative transfer was achievable in this realm. This result implied a strong coupling between the initialization error and the convergence rate, which allowed us to formalize the notion of an optimal task library as one that is maximally variance capturing of distributions in the task space. This in turn allowed us to de-

fine a procedure for distilling previous experience, and trading off storage with expressivity through the non-negative matrix factorization. Finally, we considered a semantic description of the emergent behaviours for compositional policies.

Throughout this analysis we have assumed that the reward structure for the target task was available. This describes a planning task in which the agent is given the intended goal, and must use its experience in lieu of a dynamics model, to jump-start its performance. In the next chapter we consider the situation in which the agent faces some new task in the domain, but is not given the reward structure *a priori*. This situation is closer in spirit to the standard RL picture. We will leverage the interpretation of the distilled policies as constituting abstractions over the task space, to improve the agent's exploration capabilities, and its ability to quickly hone in on reward signals when they are encountered. To achieve this seamless interaction between the agent operating in the real-world and this internal abstracted representation, we augment the agents control task to incorporate the distilled policies as additional states, and formalize a mechanism for consistently mapping between these.

The real voyage of discovery consists, not in seeking new landscapes, but in having new eyes.

-Marcel Proust

4

Hierarchical Abstraction: A Single Layer Hierarchy

4.1 Introduction

In the previous chapter it was assumed that when the agent was faced with a new task, it had access to the reward structure for that task. This corresponds to a probabilistic planning paradigm in

which the agent must utilize its experience to date, in lieu of a model for the transition dynamics, to jump-start learning on the new task. We now consider the case in which the agent is not provided the reward structure when faced with a new task, but nevertheless has access to the task library [*]. This scenario is closer in spirit to the standard RL problem formulation.

One possible approach to this problem would see the agent explore each state in the domain, recording the state-reward values as it went along. With the full knowledge of this reward structure in hand, we are returned to the previous planning paradigm, and the agent may simply act as before - by initializing its policy via Eqn.(3.3). This approach is not wholly naive given that the agent need only see each state once. This requires just $\mathcal{O}(N)$ observations, rather than the $\mathcal{O}(N \times |\mathcal{A}|)$ observations required to see all state-action pairs, as would be requisite for standard Q-learning. It is not enough that the Q-learning agent know the reward value at each individual state, instead it must further sample all actions available from each state in order to learning about the transition structure. In contrast, the LMDP agent need only learn the state reward values to perform optimally by exploiting the guaranteed optimal compositional of the LMDP formalism. In this way the agent would nevertheless make use of the task library even though every state is still visited.

Of course, we must hope to do better than this. It seems highly intuitive that the agent should be able to make use of the library tasks both to jump-start learning when some reward signal is received, and to actively improve exploration in the domain. This intuition stems in part from the fact that the library policies as recovered by Eqn.(3.9) are a structured representation of the task space, rather

[*]This work has been published as: Andrew M Saxe, Adam C Earle, and Benjamin Rosman. Hierarchy Through Composition with Multitask LMDPs. *Proceedings of the International Conference on Machine Learning (ICML)*, 70:3017–3026, 2017

than some random assortment of policies. Furthermore, it was noted in Chapter.(3) that these policies form an approximate cover for the task space. Where the transition dynamics are uniform, and the boundary rewards are equal, this corresponds to a form of state abstraction in which different library tasks drive the agent to different mutually exclusive regions of the state space. It follows that exploration over the library tasks may be more efficient than exploration over the base states.

In this chapter we develop a seamlessly integrated learning problem which incorporates the library tasks. The agent should learn to exploit the library tasks to improve exploration while it is unsure of the domain, and gradually learn to trust its base policy as it solidifies its experience on the target task. Importantly, in the limit of infinite samples, we would like the optimal policy for the integrated learning problem to converge to the true optimal policy absent the library tasks.

A similar form of this problem framing has been previously considered; most notably in the well-known options framework [30] (but see, Dayan and Hinton [71], Parr and Russell [54], Dietterich [70], Bacon and Precup [34] for additional examples). In this setting the agent is similarly equipped with a set of library tasks, and must learn to exploit these to improve exploration, and ultimately to improve learning on the target task. In order to construct an integrated learning problem, the agent's action space is augmented to include the library tasks as additional actions. At each point in time the agent may then elect to execute one of the primitive actions, or one of the available library tasks. Executing a library task corresponds to a decision by the agent to follow the action-selection policy of that task until it is stochastically terminated, after which the agent again follows its base policy over the augmented action space. Explicitly, each library task is mapped into a 3-tuple called an *option* $\langle \pi_i, \mathcal{I} \subseteq S, \mathcal{T} \subseteq S \rangle$; see Section(2.2.3) for a detailed description.

Indeed, there are a number of encouraging theoretical results that show that a suitable set of options can improve exploration, and ultimately learning on the target task[95,96]. On the other hand, there are numerous results showing that ill-chosen options can in fact hurt exploration and learning[97,46]. It turns out that it is important to select not just the right kinds of options, but also the right number of options. Due to the fact that the integrated learning problem is constructed by appending the options to the agent's action space, having too many options can be crippling, since the agent must learn a mapping from each state into the now inflated action space. Similarly, ill-chosen options may slow learning by, for example, dragging the agent to distant, uninteresting, regions of the state space. In general, it is not enough to have a useful library policy in hand, a suitable initialization and termination set must also be defined in order to bring about a useful option. A failing in any of these elements can result in a poor option.

Intuitively, in so much as an option can be thought of as a temporally extended action, it is fruitful to think about options as providing short-cuts between interesting regions of the state space. Indeed one line of work attempts to uncover options by identifying states through which the agent passes frequently[98]. Once these bottle-neck states are identified, options are constructed to reach them[99,100,101]. In another line of work, the target MDPs are first converted into a graph structure, after which graph clustering techniques are employed in order to identify connected regions of the state space. Subtasks may then be constructed to reach states at the boundaries between these regions[102]. Alternately, subtask states can be identified by their betweenness, counting the number of shortest paths that pass through each specific node[103]. In a rooms domain, such as is depicted in Fig.(3.2), this style of analysis would yield the doorways as subtask goal states.

In this chapter, we will follow a similar conceptual progression in the development of an integrated hierarchical learning problem, which incorporates the library tasks. This development extends the capability of the multitasking LMDP agent to situations in which the reward structure is not provided up front. In Section.(4.2) we construct the integrated learning problem by augmenting the state-space with a set of subtask states corresponding to the library tasks. Transitioning into a subtask state corresponds to a decision by the agent to utilize some blend of the 'higher layer' library tasks. This approach is in stark contrast with the standard options implementation outlined above, in which the library policies are first mapped to options before being added to the agent's action-space. By carefully exploiting the compositionality of the LMDP we are able to equip our agent with an arbitrary number of additional policies while avoiding the corresponding explosion in the number of parameters that must be learnt.

However, the integrated learning problem has a somewhat more complex execution model. This is a result of the agent periodically probing the higher layer for guidance, rather than receiving the one-shot initialized policy as in Chapter.(3). The new execution model is described in detail in Section.(4.3), along with pseudo-code for the algorithm. Finally in Section.(4.4), we demonstrate that viewed through a numerical lens, the integrated learning problem corresponds to a special transformation of the base problem which is well suited to expedient solution.

In Chapter.(3), the learning paradigm was described as containing two key phases. Firstly the agent would leverage its previous experience to jump-start learning on a new task by initializing its policy via Eqn.(3.3). Thereafter the agent would make minor tweaks to its policy; fine-tuning performance for the specifics of the task at hand. The integrated learning paradigm has evolved to

contain an additional phase of learning. In this initial phase of learning the agent utilizes its previous experience, absent a reward signal, to actively aid in the exploration of a domain. Moreover, the hierarchical control architecture endows our agent with great flexibility, allowing it to rapidly hone in on reward signals as they are uncovered, even in dynamic environments.

4.2 Constructing an Integrated Learning Problem

In this section we construct an integrated learning problem which incorporates the additional policies from the library tasks which are available to the agent. In this new problem construction the agent learns to utilize the subtask policies when it is unsure of the domain, and slowly transition to trusting its base policy as its experience solidifies. Moreover, the resulting hierarchical control architecture allows the agent to respond rapidly and flexibly to reward signals as they are encountered.

In line with our usual notation, the state space $\mathcal{S}$ is separated into two disjoint subsets denoting the N_i *interior* states $\mathcal{S}_\mathcal{I}$, and the N_b absorbing *boundary* states $\mathcal{S}_\mathcal{B}$. We begin with the base LMDP $L^\circ = \langle \mathcal{S}, P_{\bar{\pi}}, \mathbf{q} = [\mathbf{q}_\mathcal{I}, \mathbf{q}_\mathcal{B}] \rangle$, which is defined over state space $\mathcal{S}$, with passive transition probability $P_{\bar{\pi}}$ - and is specified uniquely by the exponentiated boundary reward structure $\mathbf{q}_\mathcal{B} = \exp(\mathbf{r}_\mathcal{B})$ for the specific task at hand. The agent is then further equipped with an additional N_t subtask policies, each of which corresponds to an LMDP $L^t = \langle \mathcal{S}, P, \mathbf{q} = \left[\mathbf{q}_\mathcal{I}, \mathbf{q}^t_\mathcal{B}\right] \rangle$ for $t = 1, \ldots, N_t$. Each of these subtasks operate in the same state-space, and under the same passive dynamics as the base LMDP but differ in their terminal boundary reward structures $\mathbf{q}^t_\mathcal{B} = \exp(\mathbf{r}^t_\mathcal{B})$.

The transition dynamics for an LMDP with transitions $P \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ defined row-wise, can be

decomposed into interior and boundary transitions as follows:

$$P = \left[\begin{array}{c|c} P_{\mathcal{II}} & P_{\mathcal{IB}} \\ \hline \mathrm{o} & I \end{array}\right], \tag{4.1}$$

where the notional boundary-to-boundary transitions $P_{\mathcal{BB}} = I$, and similarly the boundary-to-interior transitions $P_{\mathcal{BI}} = \mathrm{o}$ due to the absorbing nature of the boundary states.

In order to construct a control hierarchy, we augment the base LMDP with an additional set of N_t terminal boundary states, one state corresponding to each additional subtask in the library. The agent now operates in the augmented LMDP $\hat{L} = \langle \hat{\mathcal{S}} = \mathcal{S}_{\mathcal{I}} \cup \mathcal{S}_{\mathcal{B}} \cup \mathcal{S}_{\mathcal{T}}, \hat{P} = [P_{\mathcal{II}}, P_{\mathcal{IB}}, P_{\mathcal{IT}}], \mathbf{q} = [\mathbf{q}_{\mathcal{I}}, \mathbf{q}_{\mathcal{B}}, \mathbf{q}_{\mathcal{T}}] \rangle$, over $\hat{N} = N_i + N_b + N_t$ states. Transitions into these new 'subtask states' are governed by a new $N_i \times N_t$ transition matrix which we denote $P_{\mathcal{IT}}$. The subtask transition structure may be hand crafted by a designer, although a method to uncover a suitable transition structure is developed in Section.(5.4). The transition structure for the augmented LMDP is then:

$$\hat{P} = \left[\begin{array}{c|c|c} P_{\mathcal{II}} & P_{\mathcal{IB}} & P_{\mathcal{IT}} \\ \hline \mathrm{o} & I & \mathrm{o} \\ \hline \mathrm{o} & \mathrm{o} & I \end{array}\right], \tag{4.2}$$

and is depicted graphically in Fig.(4.1). Note that the transition dynamics for the new augmented LMDP, $\hat{P}$, must also be normalized row-wise.

The learning procedure in the augmented problem is then carried out as follows. As usual, the agent transitions stochastically over the now augmented state space; following the controlled dy-

namics. When the agent transitions into an interior or boundary state, the stochastic estimate for the controller is updated, and the agent proceeds as before. However, when the agent transitions into a subtask state, a new procedure is executed by the higher layer controller. This procedure results in a reward structure being passed down from the higher layer which defines the value of the terminal boundary rewards over the subtask states. This procedure effectively specifies an interim subgoal for the agent to achieve. By specifying a useful sequence of these subgoals, the higher layer effectively transforms away the fine-grained details of the base MDP, and instead allows the agent to learn over the abstracted problem. Formally, once the new reward structure for the subtask states has been specified, the agent may jump-start learning on the interim task by attempting to meet the new boundary reward structure via Eqn.(3.3) using some linear combination of the augmented library policies. This execution model is described in greater detail, and presented in pseudo-code, in Section.(4.3).

Semantically speaking, a transition into one of the subtask states corresponds to a decision by the agent to get guidance from the higher layer by leveraging some blend of library task policies. Procedurally, no real time passes during such a transition, and the agent remains in (or is returned to) the same base state, but with a different control policy - determined by the interim task set by the higher layer. Of course, the question remains as to how the higher layer is able to specify sensible subgoals for the base layer. The key insight is that a new learning problem can be constructed over just the subtask states, and that by solving this new, simplified learning problem, the higher layer is able to efficiently determine useful subgoals. Explicitly, we construct a new LMDP over the subtask states that suitably integrates the subtask states with the base LMDP through the corresponding

subtask policies.

In order to construct a new LMDP over the subtask states, we are required to specify the corresponding transition structure $P_{\mathcal{TT}}$,$P_{\mathcal{TB}}$, and reward structure $\mathbf{r}_{\mathcal{T}}$. Intuitively, we require that the new LMDP over the subtask states be a faithful representation of the base LMDP. It follows that the transitions *between* subtask states themselves may be suitably defined as the likelihood of being in one subtask state, and entering another by diffusion through the interior states via $P_{\mathcal{II}}$, and into the next subtask state via $P_{\mathcal{IT}}$. In this way the subtask transition dynamics may be consistently defined as:

$$P^{\mathrm{I}}_{\mathcal{TT}} = P^{\circ^T}_{\mathcal{IT}} \left(I - P^{\circ}_{\mathcal{II}}\right)^{-1} P^{\circ}_{\mathcal{IT}}, \tag{4.3}$$

$$P^{\mathrm{I}}_{\mathcal{TB}} = P^{\circ^T}_{\mathcal{IB}} \left(I - P^{\circ}_{\mathcal{II}}\right)^{-1} P^{\circ}_{\mathcal{IT}}. \tag{4.4}$$

Notice that the same term appears in each of the transition equations above, namely $\left(I - P_{\mathcal{II}}\right)^{-1}$. This term corresponds to the future state occupancy under the passive transition dynamics: the steady state distribution of the corresponding Markov chain. It is interesting to note that the same quantity is considered as a representation basis in the successor representation[71] (discussed in Section.(3.2)). Conceptually this is a useful decoupling of the expectation state occupancy from the expected reward value. Furthermore, this term is related to a transformation of the eigenvalue estimation problem into its inverse formulation - this is described in greater detail in Section(4.4).

The transition equations defined above ensure that the transition dynamics between subtask

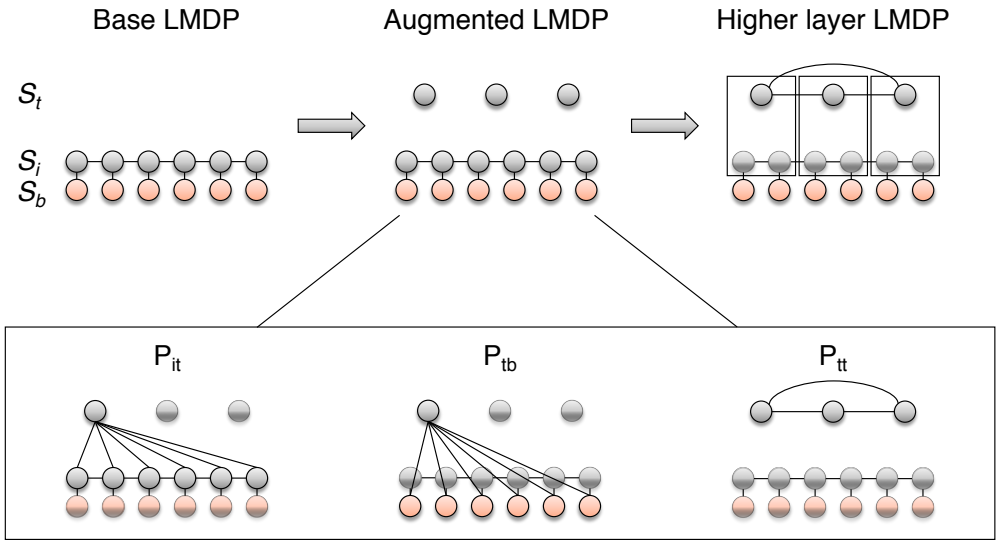


Figure 4.1: The integrated learning problem; a single layer of hierarchy. The base LMDP is first augmented with an additional set of N_t subtask states. The new transition dynamics incorporating the subtask states are specified in the second row. Transitions from the N_i interior states into the subtask states are governed by a new matrix $P_{\mathcal{IT}} \in \mathbb{R}^{N_i \times N_t}$ - these transitions typically being hand crafted. The corresponding transitions from the subtask states into the boundary states, and from subtask states into other subtask states are governed by the transition matrices $P_{\mathcal{TB}}, P_{\mathcal{TT}}$ respectively. These dynamics are constructed via Eqns.(4.3, 4.4) as the expected transition probabilities under random walk with the specified subtask transitions $P_{\mathcal{IT}}$. Finally, once the subtask state costs $\mathbf{q}_{\mathcal{T}}$ have been specified, the resulting construction is a new LMDP in which the agent learns over both the base problem, and the new subtask states. Noting that $P_{\mathcal{IT}}$ is a map from base states into subtask states, the decoupled sub-problem specified via $\hat{L} = \langle \hat{S} = S_{\mathcal{T}} \cup S_{\mathcal{B}}, \hat{P} = [P_{\mathcal{TT}}, P_{\mathcal{TB}}], \hat{\mathbf{q}} = [\mathbf{q}_{\mathcal{T}}, \mathbf{q}_{\mathcal{B}}] \rangle$, corresponds to an abstract representation of the base LMDP which may be solved in isolation.

states are consistent with the base problem with respect to the subtask transition matrix $P_{\mathcal{IT}}$. If the diffusion length between subtask states via the base transition structures $P_{\mathcal{II}}$ and $P_{\mathcal{IT}}$ is large, then the interior transition probability specified in $P_{\mathcal{TT}}$ will be small. Similarly, if the diffusion length is short, then the corresponding transition probability will be large.

The interior rewards for each of the subtask states may be computed in a similar fashion as the expected cumulative reward accrued under random walk at the base layer:

$$\mathbf{q}_{\mathcal{T}}^{\mathrm{I}} = \mathbf{q}_{\mathcal{I}}^{\circ^T} \left(I - P_{\mathcal{II}}^{\circ}\right)^{-1} P_{\mathcal{IT}}^{\circ}, \tag{4.5}$$

although in practice the interior rewards can be simply set to some small constant value, $\mathbf{q}_{\mathcal{T}} = 0.1$.

With the constituent components defined, the LMDP over the subtask states may be written as $L^{\mathrm{I}} = \langle \mathcal{S}_{\mathcal{T}}, P = [P_{\mathcal{TT}}, P_{\mathcal{TB}}], \mathbf{q} = [\mathbf{q}_{\mathcal{T}}, \mathbf{q}_{\mathcal{B}}] \rangle$, where the superscript I indicates that this is the 'higher layer' LMDP. Of course, the construction above is such that the higher layer LMDP is well defined for any number of subtask states. However, when the number of subtask states is fewer than the number of interior states, the higher layer problem constitutes an abstraction and simplification of the base problem. Critically, this abstracted problem is, by construction, consistent with the base problem with respect to the map $P_{\mathcal{IT}}$. The entire augmented learning problem $\hat{L}$ is said to have a hierarchical architecture, and to be a hierarchical control problem.

The mapping from the base LMDP into the higher layer LMDP is defined by the transition matrix $P_{\mathcal{IT}}$. The choice for $P_{\mathcal{IT}}$ is therefore critical. The subtask transition matrix is typically taken to be surjective, with each subtask state reachable via some base state transition. Moreover, $P_{\mathcal{IT}}$ is typi-

cally a compression, mapping the N_i base interior states into $N_t < N_i$ subtask states. In this case the higher layer LMDP may again be interpreted as the resulting problem from a state aggregation procedure. While the surjective nature of $P_{\mathcal{IT}}$ aligns with our intuition of the abstracted LMDP being a simplification of the base LMDP, in principle there is no reason that $P_{\mathcal{IT}}$ shouldn't correspond to a dilation map in which $N_t > N_i$ and region of the base LMDP may be over represented in the abstracted problem.

Since the higher layer is itself a bona fide LMDP, it can be solved using standard techniques. Solving the abstracted problem yields an optimal desirability function over the subtask states, $\mathbf{z}^* \in \mathbb{R}^{N_t+N_b}$. The controlled transition dynamics shift probability mass, via Eqn.(2.13), making transitions to some states more or less likely than they were under the passive dynamics. Transitions that are more likely under the controlled dynamics correspond to states that are preferred for the current task, while those that are less likely under the controlled dynamics correspond to states that are worse for the task at hand. When a higher layer state is preferred under the controlled dynamics, this may be interpreted from the perspective of the lower layer, as the corresponding subtask policy having a high weight and being useful for the current base task.

Following this intuition, the terminal boundary rewards for the subtask states passed down from the higher layer are set to be proportional to the difference between the controlled dynamics, $\mathbf{u}^{\mathrm{I}}(s)$ determined by Eqn.(2.13), and the passive dynamics, $\mathbf{p}^{\mathrm{I}}(s)$ at the current state:

$$\mathbf{q}^{\circ}_{\mathcal{T}} = \mu \left(\mathbf{u}^{\mathrm{I}}(\cdot) - \mathbf{p}^{\mathrm{I}}(\cdot) \right), \tag{4.6}$$

where $\mu \in \mathbb{R}$ is a free parameter controlling the ratio of the terminal rewards between boundary states and the subtask states. Heuristically, μ may be set so as to balance the terminal rewards such that $||\mathbf{q}_{\mathcal{B}}|| = ||\mathbf{q}_{\mathcal{T}}||$. Supposing that the task corresponds to a singleton goal state, we set $\mu = 1/||\mathbf{u}^1 - \mathbf{p}^1||$. Since $\mathbf{q}^\circ_{\mathcal{T}}$ represents the exponentiated boundary reward for the subtask states at the base layer, and must be strictly positive, we take $\mathbf{q}^\circ_{\mathcal{T}} < 0 \Rightarrow 0$: explicitly setting the negative components of $\mathbf{q}^\circ_{\mathcal{T}}$ to zero. This corresponds to a null contribution from subtask states that are disfavoured by the higher layer.

In addition, each library task is equipped with an extended boundary reward structure defined over the augmented state space. The extended boundary rewards are constructed by appending a one-hot vector, defined over the new subtask states, to the initial boundary reward structure, $\hat{\mathbf{q}}_i^1 = [\mathbf{q}_{i_{\mathcal{B}}}, \mathbf{e}_i]$ for $i \in [1, \ldots, N_t]$, where $\mathbf{e}_i \in \mathbb{R}^{N_t}$ is the i^{th} coordinate vector. Intuitively, this corresponds to fact that each subtask may be trivially represented within the task library with a coordinate weight vector.

Then, in order to jump start learning on the interim task, the agent may attempt to meet the target boundary reward structure for the augment task as before:

$$min_{\mathbf{w}} ||\mathcal{Q}^\circ_{\mathcal{B}\cup\mathcal{T}}\mathbf{w} - \mathbf{q}^\circ_{\mathcal{B}\cup\mathcal{T}}||. \tag{4.7}$$

Critically, while the higher layer periodically specifies the interim task that that agent must perform, it does not provided any specification for how it should be achieved. Abstractly, the idea that a hierarchical controller should specify tasks, but not directly affect the policies, for the lower layer

controllers is not new. This idea was first expressed in a framework called Feudal RL[71], and has been revived in some recent work[104].

It is clear that the properties of the subtask transition matrix will have an important effect on learning in the augmented LMDP. In Chapter.(5) we suggest an analytic alternative for the construction of the subtask transition matrix which further aligns the abstraction mechanism with the library subtasks. Intuitively, we would like for $P_{\mathcal{IT}}$ to satisfy some restricted isometry property, such that states that are close in the base problem, remain close in the abstracted problem. Such a property would bring the procedure described herein in line with more common ideas in the state abstraction literature.

4.3 The Execution Model

The execution model for the offline multitask LMDP (MLMDP) contains two distinct phases. In the first phase, the agent initializes an estimate for the desirability function by approximating the reward structure for the new task by a linear combination of reward structures of the library tasks. In the second phase, the agent executes the stochastic z-learning procedure to fine-tune the estimate of the optimal desirability function. The resulting execution paradigm differs from the initial procedure described in Todorov [45] only in the first phase, in which the agent uncovers an improved initial estimate. In contrast, the execution model for the online MLMDP is more complex, involving a dynamic interplay between the agent and the task library. In this section we described the online execution model in detail, and provide heuristics for the additional free parameters introduced.

In Section.(4.2) we described the construction of an augmented LMDP which we referred to as the integrated learning problem. In the online execution model, the agent operates in this augmented LMDP, iteratively drawing next states, and stochastically updating estimates of key variables as before. The primary difference between the online and offline execution models is the way in which the operative desirability function is constructed. In the offline model the agent maintains a single estimate for $\mathbf{z}$. This estimate being initialized via the task library as a function of the boundary reward structure $\mathbf{r}_b$. In the online model, this initialization procedure is not possible because the agent does not have access to the boundary reward structure up front. Instead, the agent will periodically consult the task library for guidance as it experiences more of the domain, and improves its estimate of the boundary reward $\hat{\mathbf{r}}_{\mathcal{B}}$. As such the agent maintains two different values for the desirability function. The first is a persistent estimate derived exclusively from the agent's direct experience in the domain, $\mathbf{z}^{base}$. This estimate is stochastically updated on each transition in the standard fashion, and changes slowly in the sense that $\mathbf{z}^{base}$ will differ between successive estimates by at most a single component. The second is a variable estimate derived from the task library $\mathbf{z}^{hl}$ which is updated only when the agent enters a subtask state. This estimate is liable to change rapidly as a single updated element in the boundary reward estimate $\mathbf{r}_{\mathcal{B}}$ may result in a significant re-weighing of the library tasks. Finally, the agent transitions under the *effective* controller $\mathbf{z}^{eff}$ which is derived as a linear combination of the base estimate $\mathbf{z}^{base}$ and the higher layer estimate $\mathbf{z}^{hl}$:

$$\mathbf{z}^{eff} = \gamma \mathbf{z}^{base} + (1 - \gamma)\mathbf{z}^{hl}, \tag{4.8}$$

where $\gamma \in \mathbb{R}$ is an interpolation parameter. We defer the discussion of how this parameter ought to be established to later in this section. First, notice that this simple blend of estimates for the desirability function makes critical use of the compositionality of the LMDP formulation. At each point in time, the agent acts under the influence of both the base policy and the higher layer policy in a graded fashion. When the directed behaviours encoded by the policies align the agent is wholly committed to that course of action. However, when the behaviours are opposed, this uncertainty is expressed in the agent's effective behaviour. Furthermore, the fact the the constituent policies are updated at different frequencies naturally captures the multiscale nature of the resultant behaviours.

As noted above, the updates to $\mathbf{z}_{base}$ occur at every transition via:

$$z_s^{base} = \alpha z_s^{base} + (1 - \alpha) q_s z_{s'}^{base}, \tag{4.9}$$

similarly the estimate $\hat{\mathbf{r}}_{\mathcal{B}}$ is updated for each boundary transition $s' \in \mathcal{S}_{\mathcal{B}}$ via the direct assignment:

$$\hat{r}_{\mathcal{B}_{s'}} = r_{\mathcal{B}_{s'}}. \tag{4.10}$$

The updates to $\mathbf{z}^{hl}$ occurs only on subtask transitions, $s' \in \mathcal{S}_{\mathcal{T}}$. As described in Section.(4.2) above, the action of the higher layer LMDP is to implicitly set an interim subtask for the lower layer by defining a particular reward structure over the subtask boundary states $\mathbf{q}_{\mathcal{T}}$ via Eqn.(4.6). The higher layer components $\mathbf{z}^{hl}$ are determined as the agent attempts to meet this new boundary reward structure via Eqn.(4.7). Importantly, this requires an estimate for the full reward structure

$\hat{\mathbf{r}}_{\mathcal{B}}$. The initialization of this estimate is crucial since it directly modulates the effect of the task library. The pessimistic initialization $\hat{\mathbf{r}}_{\mathcal{B}} = -\infty$ results, initially, in a null contribution from the higher layer. It is only as higher value components are revealed from experience that the task library it utilized. This corresponds to a mode of operation in which the higher layer policies are used to direct the agent to desirable states only once those states have been uncovered via the unguided exploration of the base policy. Alternatively, the optimistic initialization $\hat{\mathbf{r}}_{\mathcal{B}} = 0$ results in a uniform initial contribution from the higher layer. And it is only as states are revealed to have low value, that the higher layer discards their contributions. This corresponds to a mode of operation in which the higher layer guides early exploration in the domain, driving the agent to unexplored regions of the domain until local rewards are determined to be sufficiently valuable to offset this behaviour. In practice we find the optimistic initialization to perform better than the pessimistic initialization, as the agent is able to leverage the structural information encoded in the higher layer sooner. This is nevertheless another free-parameter.

We now return to the discussion of the interpolation parameter γ. With a renewed intuition for the early exploratory role that the course higher layer estimate $\mathbf{z}^{hl}$ plays, we propose a simple schedule for valuing γ. In practice it is sufficient to set γ to be equal to the learning rate α (however this is determined). As the learning rate is annealed from $1 \rightarrow 0$, so the agent transitions from depending solely on the higher layer for guidance to fully trusting its base policy. It is worth noting that in the first few epochs, when $\alpha \approx 1$, the interplay between $\mathbf{z}^{hl}$ and $\mathbf{z}^{base}$ exhibits a teacher-student like duality. The base policy is continually assessing the value of states along a guided trajectory, and in time, using that evaluation to shape future trajectories. This also has the pleasing property that the

asymptotic solution is equal to the flat implementation.

Algorithm 2 Online Multitask LMDP

Require: $Z, \mathcal{Q}_{\mathcal{B}}, \mathcal{S}_{\mathcal{B}}$

procedure ONLINE MLMDP($\mathcal{N}$) # Given maximum step count $\mathcal{N}$
  # *Initialize values*
  $\mathbf{q}_{\mathcal{T}} = 0$
  $\mathbf{z}^{HL} = 0$
  $\mathbf{z} = 1$
  $\hat{\mathbf{q}}_{\mathcal{B}} = 1$ # Optimistic initialization
  $\mathbf{z}^{eff} = \gamma \mathbf{z} + (1 - \gamma)\mathbf{z}^{hl}$
  # *Z-iteration loop*
  for $k \leftarrow 1$ to $\mathcal{N}$ do
    $\mathbf{u} \leftarrow \mathcal{N}(\mathbf{p} \odot \mathbf{z}^{eff})$ # Obtain normalized controlled dynamics
    $(q_s, s') \sim \mathbf{u}$ # Draw next state
    $z_s = \alpha z_s + (1 - \alpha) q_s z_{s'}$ # Update base $\mathbf{z}$
    # *Update boundary reward estimate*
    if $s' \in \mathcal{S}_{\mathcal{B}}$ then
      $\hat{\mathbf{q}}_{\mathcal{B}}(s') = \mathbf{q}_{\mathcal{B}}(s')$
      Break # Terminate at boundary states
    else if $s' \in \mathcal{S}_{\mathcal{T}}$ then
      $\mathbf{q}_{\mathcal{T}} \propto \mathbf{u}^{\mathrm{I}} - \mathbf{p}^{\mathrm{I}}$ # In-paint subtask rewards
    end if
    # *Update effective* $\mathbf{z}$
    $\mathbf{w} \leftarrow min_{\mathbf{w}} ||\mathcal{Q}_{\mathcal{B}\cup\mathcal{T}}\mathbf{w} - \hat{\mathbf{q}}_{\mathcal{B}\cup\mathcal{T}}||$
    $\mathbf{z}^{hl} = \mathcal{Z}\mathbf{w}$ # Set $\mathbf{z}^{hl}$ as blend of basis functions
    $\mathbf{z}^{eff} = \gamma \mathbf{z} + (1 - \gamma)\mathbf{z}^{hl}$
  end for
  return $\mathbf{z}$ # The optimal desirability function
end procedure

The performance of the online MLMDP is considered in Fig.(4.2). The agent is again tasked with reaching the top-left, bottom-right, bottom-left, and top-right corners of the domain. The tasks are presented in the same predefined order, with a new task being presented every 150 epochs. Each epoch consists of a run of up to 100 steps with early-exit behaviour on boundary state-transitions.

Details of the experimental setup can be found in Appendix.(A.5).

As expected, the agent significantly out performs the flat implementation in all cases. This is an important observation, as it implies that the reliance on the higher layer policies in early phases of learning has not detrimentally affected exploration. Interestingly however, in a stark reversal of behaviour from the offline MLMDP, the agent actually performs better as the number of subtasks is reduced. The increased performance is due to the fact that individual observations which the agent makes in the domain, are attributable to more general abstractions when the number of subtasks is reduced. By way of example, when the agent is equipped with $N_t = 52$ subtasks, each subtask corresponds roughly to two rewarded states at the base layer. When the agent then observes that one of those states is not a goal state, this signal is propagated to the subtask state, and by down weighting its contribution to $\mathbf{z}^{hl}$, the agent effectively avoids both of the base layer states corresponding to the single subtask state. Alternatively, when the agent is equipped with $N_t = 4$ subtasks, each subtask corresponds roughly to one whole room. Now when the agent observes that a particular state is not a goal state, this signal is again propagated to the subtask state, and the agent effectively avoids the entire room.

This attribution from specific observations to general abstractions is justified for real-world tasks in which rewards are often clustered at higher layer states. For example, a person may be tasked with 'getting to the post office' without being told exact where to stand once they've arrived. Said differently, there is a broad region of the state space which is rewarded for the task, rather than a singleton state. Nevertheless, spurious tasks may always be constructed which break these good-in-principle behaviours.

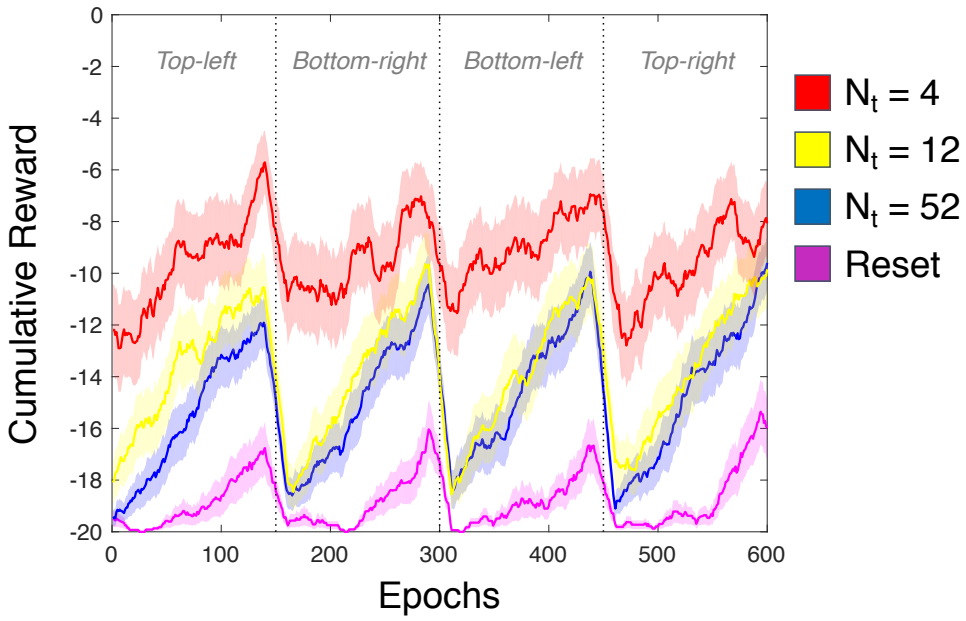


Figure 4.2: Online multitasking with the MLMDP. The agent is again tasked with reaching each of the 4 corners of the domain. The tasks are presented in the same predefined order, with a new task being presented every 150 epochs. When a new task is presented, the agent resets both its estimate of the boundary reward function and its base controller to the default initialization values. In a reversal of behaviour from the offline setting, the agent here exhibits improved performance as the number of subtasks decreases. This is due to the fact that the agent is able to better attribute individual observations to more general abstractions when the number of subtasks is fewer. Nevertheless, the agent's performance exceeds the flat implementation in all cases by a significant margin. This suggests there there is an optimal number of subtasks for the distribution of target tasks in the domain.

4.4 Learning Dynamics in the Hierarchy

In Section(4.2) we interpreted the higher layer transitions dynamics $P^{\pi}_{\mathcal{I}\mathcal{I}}$, $P^{\pi}_{\mathcal{I}\mathcal{B}}$, obtained via Eqns.(4.3, 4.4), as corresponding to the probability of transitioning between higher layer states by diffusing at the lower layer. We argued there, that these equations forced a consistency requirement on the abstracted LMDP, such that solving the higher layer LMDP would provide a good approximate solution to the base layer LMDP. Here we explore a dual interpretation in which the higher layer LMDP is seen to be a special transformation of the base problem, which is particularly well suited to expedient numerical solution.

In the LMDP formalism, the optimal controller may be obtained through the solution of a linear eigenvalue problem, as described in Section.(2.12). When the transition dynamics and reward structure are known, one may solve for the dominant eigenvector via power iteration. The power method is known to converge geometrically with ratio $\mathcal{O}\left(\lambda_2/\lambda_1\right)$, where the fraction λ_2/λ_1 is the ratio of the secondary eigenvalue to the dominant eigenvalue. When the value of the target eigenvalue is known *a priori*, the corresponding eigenvector may sometimes be obtained more expediently by instead solving the the inverse problem[105,106].

In the interests of generality, the following analysis is presented for a general eigenvalue problem rather than for the specific closed form solution of the optimal control in the LMDP formulation. We switch notation here to reinforce this change.

The fact that the inverse transformation allows for more expedient numerical solution follows immediately from the fact that eigenvectors of the full rank square matrix $\mathcal{A} \in \mathbb{R}^{N\times N}$ and $(\delta I - \mathcal{A})^{-1}$,

for some scalar $\delta \in \mathbb{R}$ are the same. Specifically:

$$\lambda \mathbf{x} = \mathcal{A}\mathbf{x} \implies \gamma \mathbf{x} = (\delta I - \mathcal{A})^{-1}\mathbf{x}, \tag{4.11}$$

where $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathcal{A}$, and $\gamma \in \mathbb{R}$ is an eigenvalue of the inverse system $(\delta I - \mathcal{A})^{-1}$. Moreover, the eigenvalues λ and γ are related via the so-called shift parameter δ:

$$\gamma = \frac{1}{\delta - \lambda}. \tag{4.12}$$

In particular, power iteration on the inverse problem converges to the eigenvector associated with the eigenvalue closest to δ. When the eigenvalue λ of the target eigenvector is known, the convergence ratio $\mathcal{O}\left(\gamma_2/\gamma_1\right)$ can be made arbitrarily small by noting that $\gamma_2/\gamma_1 = {}^{\delta-\lambda_1}/_{\delta-\lambda_2} \to 0$ as $\delta \to \lambda_1$. This result is intuitive in our particular setting in light of the fact that when $\mathcal{A}$ is a stochastic matrix, then $(I - \mathcal{A})^{-1}$ corresponds precisely to the steady state distribution. Notice that from the block structure of the transition dynamics in Eqn.(4.2), the components of the desirability function associated with the boundary states remain unchanged through the inverse transformation.

As a sanity check, the convergence results for direct and inverse power iteration are investigated numerically in Fig.(4.3). Here we generate 1000 random left stochastic matrices, and run 20 steps of power iterations, and inverse power iteration on each; recording at each iteration the normed error $||\mathbf{x}^* - \mathbf{x}^k||$, where $\mathbf{x}^k$ corresponds to the approximate eigenvector at the k^{th} iterate. Since the matrices are stochastic, we know that the dominant eigenvalue has value 1[107,45] (this well known fact

follows directly as a result of Gershgorin's circle theorem, although there are many other ways to see this), and compute the inverse power iterate via Eqn.(4.11) with $\delta = 1 - 10^{-6}$. The asymptotic log-error is bounded below by machine precision, and should be taken as functionally zero.

Furthermore, suppose that an oracle had provided us the matrix B, whose columns are the eigenvectors of the inverse problem $(\delta I - \mathcal{A})^{-1}$. Note that the eigenvectors are unique since $\mathcal{A}$ is assumed to be full rank. In this basis, the eigenvectors are one-hot and the eigenvalue problem may be written as:

$$\gamma(B\mathbf{e}_i) = (\delta I - \mathcal{A})^{-1}(B\mathbf{e}_i), \tag{4.13}$$

where $\mathbf{e}_i$ is the i^{th} coordinate vector. The fact that the eigenvectors we seek are one-hot has implications for the online estimation problem. Typically, when performing online sample estimation, as in Eqn.(2.14) for example, a non-zero learning rate is annealed down to zero in order to guarantee convergence. If, however, we know up front that the target values are explicitly 1 or 0, then the learning rate can be set to 0 and a sort of 'one-shot' learning is possible.

The form of the equation above is similar to the form of the higher layer interior transition dynamics specified in Eqn.(4.3). Indeed the higher layer interior dynamics are recovered by left multiplication by the basis transpose, $\gamma B^T B\mathbf{e}_i = B^T(\delta I - \mathcal{A})^{-1}(B\mathbf{e}_i) = P^1_{\mathcal{I}\mathcal{I}}$, when $\mathcal{A} = P^0_{\mathcal{I}\mathcal{I}}$ and $\delta = 1$. The choice $\delta = 1$ ensures that, in the inverse problem, we continue to estimate the dominant eigenvector associated with $\lambda = 1$. Specifically, when the vectors in the eigenbasis are orthonormal, then $B^T B = I$ and we recover the higher layer dynamics exactly.

More generally, we have that when $B^T \in \mathbf{R}^{N\times k}$ is a basis for a maximally variance capturing

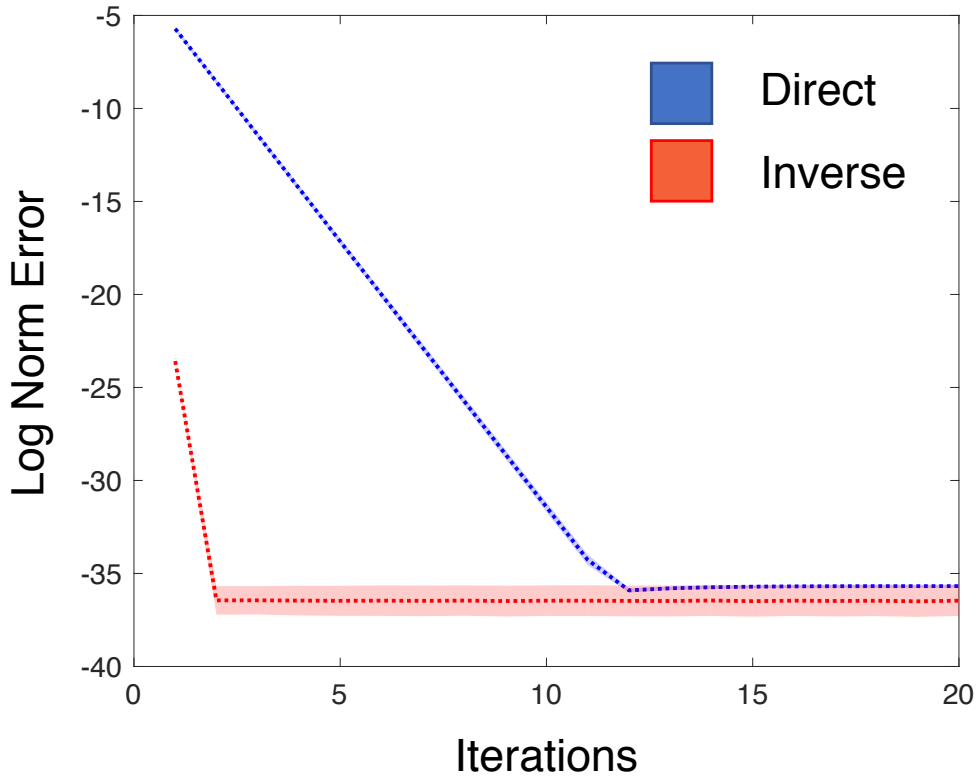


Figure 4.3: Inverse power iteration converges much more rapidly than direct power iteration, when the value of the target eigenvalue is known. In fact, the inverse method is able to converge in a single iteration. The non-zero asymptotic error in the figure is a vestige of machine precision and should be taken to be functionally zero. Furthermore, the known geometric convergence of the power method is clearly evidenced here in the linear log-error of the direct method.

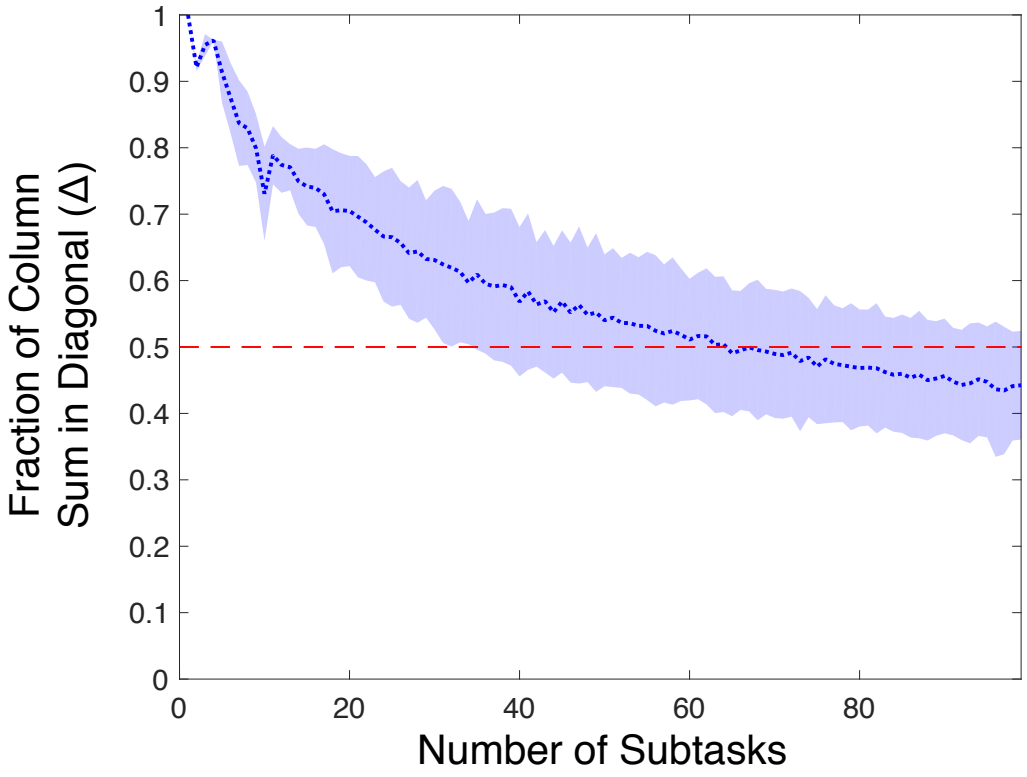


Figure 4.4: The low-rank approximations to the task basis uncovered via NNMF are approximately orthogonal for the canonical rooms domain. We observe that for the low-rank approximation W, the product $W^T W$ is approximately diagonally dominant. The degree of orthogonality reduces as the rank of the approximation increases. This is expected since the underlying target matrix is not orthogonal. For higher-rank approximations, the product $W^T W$ falls below the threshold for formal diagonal dominance.

subspace, then $B^T B$ is typically diagonally dominant, and the eigenvector is maximally sparse in the chosen basis. The inverse formulation is therefore a new eigenvalue problem in which the solution is maximally sparse, and converges rapidly. We investigate the orthogonality of the of subtask transition matrices uncovered via NNMF for different values of $k = N_t$ in Fig.(4.4). For each $N_t \in [1, \ldots, N]$ we compute the low-rank approximation to the task basis $W \in \mathbf{R}^{N \times N_t}$ and compute a scalar measure of the orthogonality of the resulting matrix. Explicitly we measure $\Delta = \bar{d}$ where $d = {}^{w_{ii}}/_{\sum_j w_{ji}}$ as the average ratio of the diagonal element to the column sum; reporting both $\bar{d}$ and $\langle d \rangle$ in Fig.(4.4).

The interpretation of the higher layer transition equations as an inverse transformation of the base eigenvalue problem yields some important insights. By way of example: up until this point we have assumed that the subtask transition matrix $P_{\mathcal{T}}$ would typically be handcrafted by a designer as a problem specific parameter. However, in order to qualify as a suitable candidate for an approximate similarity transformation of the base problem, the subtask transition matrix must possess some specific additional properties. This suggests a path to automating the selection of $P_{\mathcal{T}}$, and in so doing remove the final manual step in the higher layer transformation procedure.

Also, the downward communication between the higher layer and the base layer has thus far been interpreted as the higher layer setting a sequence of interim tasks for the base layer to achieve, by dynamically specifying the terminal boundary rewards at the subtask states. However the subtask policies may also be directly interpreted in terms of the resulting behaviours they command at the base layer. We explore both the autonomous construction of the subtask transition matrix and the direction interpretation of the subtask policies in the following chapter.

The key to artificial intelligence has always been the representation

-Jeff Hawkins

5

Autonomous Skills Discovery

5.1 Introduction

Decomposing a single complex task into a set of simpler subtasks is a well known approach to simplify the learning problem. The resulting subtask structure provides a number of important benefits. Firstly it presents a way to short cut the sparse reward signal of the initial task by instead providing the agent with a pseudo-reward signal for completing the subtasks; each of which typically have a

much shorter planning horizon. In addition, the subtask structure itself is a potentially powerful source of transfer. Where subtasks are shared between multiple target tasks, an agent attempting to learn a number of those tasks need only learn the shared subtasks once. However, the problem of uncovering a useful set of subtasks in the first place is notorious difficult. In fact, for general MDPs, the problem is known to be NP-hard[108].

Various authors have proposed a number of differing approaches to skills discovery. Some better known approaches seek to uncover subtasks by finding so-called bottleneck states[109,110,111]. These correspond to states in the transition graph with a high betweenness measure. In another approach, subtasks are constructed which direct the agent to states which occur frequently in historically successful trajectories[112,113]. In yet another line of work, subtask policies are uncovered through analysis of the latent space representation of the policies of constituent tasks - either through a distillation[114] or embedding process[115,116]. An important conceptual benefit of subtasks uncovered in this fashion is that new policies are often available through interpolation or regularization in the latent space, and the agent needn't explicitly operate in an action space inflated by the subtask policies.

Importantly, while each of these approaches captures some valuable intuition, they are typically not based on the optimization of an explicit objective function. In addition, there are also a number of equally important corollary problems. For example, given a set of predefined subtasks, how might the agent determine the best k subtasks to keep such that planning time for the integrated problem is minimized? Or indeed, when the number of allowed subtasks is not restricted in some way, what is the optimal number of subtasks to have in the first place?

In this chapter we position the task basis decomposition procedure described in Eqn.(3.9) as a

novel subtask discovery mechanism. In Section.(5.2), we describe the way in which the unusual mathematical construction of the LMDP allows us to address a number of the important questions for subtask discovery outlined above [*]. Critically, and in stark contrast with alternative methods, the proposed subtask discovery mechanism is formulated in terms of an explicit and tractable objective function. In order to provide some intuition regarding the key differences in our approach, in Section.(5.3) we make a qualitative comparison of the subtasks uncovered by our methods with those uncovered by popular alternatives.

Finally, in Section.(5.4), we recall the insights gleaned in Section.(4.4), and propose an approach for the autonomous construction of the subtask transition matrix $P_{\mathcal{IT}}$, which utilizes the new subtask discovery mechanism. By automating this critical step in the construction of the integrated learning problem, we set the stage for the fully autonomous construction of deep hierarchical control architectures. This is further explored in Chapter.(6).

5.2 Subtask Discovery with NNMF

In this section we formalize a running intuition which we have thus far left unspecified, by explicitly interpreting the column vectors of the non-negative matrix factorization of the task basis as a set of subtasks. This positions the decomposition scheme as a novel subtask discovery method. We consider a number of standard questions in subtask discovery, by contrasting subtasks uncovered by the new method with the well known options framework.

[*]This work has been published as: Adam C Earle, Andrew M Saxe, and Benjamin Rosman. Hierarchical Subtask Discovery with Non-Negative Matrix Factorization. *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–11, 2018

Firstly, we consider the question of how a good set of subtasks may be uncovered. Importantly, one can only assess how good or bad a set of subtasks is in the context of a particular distribution from which target tasks are to be drawn. A set of subtasks that will aid learning on one distribution of target tasks, may harm learning on another. It stands to reason therefore, that subtask discovery methods which are agnostic to the specific set of tasks under consideration will forever be vulnerable to negative transfer, even with perfect knowledge of the system. Interestingly, many popular methods are based on the betweenness centrality of the underlying transition graph [111,100] and are therefore exposed to this vulnerability. On the other hand, subtask discovery methods based on finding states which are frequently visited in real trajectories are implicitly task dependent, and are therefore guaranteed to find subtasks which are useful for the given task ensemble. However, these task dependent methods require a large set of trajectories from which to extract subtasks. In order to simulate these trajectories the agent would first need a good approximate model of the transition dynamics, at which point a model-based planner might as well be used. In practice, these methods attempt to strike a delicate balance by establishing subtasks based on state frequency count over a small set of trajectories, and attempt to dynamically adjust these as the agent accrues more experience. In our proposed method, the quality of a set of subtasks can be assessed simply by considering its ability to approximate the weighted task basis $\hat{Z}$.

Secondly, we consider the related question of subtask equivalence. Given two sets of different subtasks, it is in general difficult to assess which set is better. This question typically depends on both the number and type of subtasks in the set, as well as on the task ensemble the agent will face. In our setting, this question is again simply answered by considering which set of subtasks is better

able to approximate the weighted task basis $\hat{Z}$. This result follows as a direct consequence of the convexity of the objective function, and the fact that we can couple the convergence rate with the quality of the initial guess. It is not obvious how this relation might be established for schemes formulated as general MDPs.

Thirdly, in a direct comparison to the options framework, we note that the subtasks uncovered by our proposed method are defined entirely by their optimal policies and the shared subtask transition structure. In contrast, each option must be constructed by first defining an initialization and termination set before it can be integrated into the online learning task.

Another important property of the MLMDP, absent in other formulations, is the ability to execute multiple policies concurrently. This critical feature is important for subtask discovery, since it ensures that the agent is not hamstrung by an ever inflated action space as additional subtasks are made available. At each point in time, the agent may elect to operate under an arbitrary linear combination of subtask policies, rather than needing to choose one-of-N as is typical in other methods. Conceptually the hierarchical LMDP is also distinguished in that the subtask policies correspond to an abstraction of both the state space, and the action space simultaneously. This follows from the fact that actions in the LMDP formalism correspond to distributions over next states, rather than a sequence of serially executed primitives. Furthermore, bidirectional communication between layers in the MLMDP is reminiscent of a feudal-like architecture in which the higher layer specifies goals for the lower layer controller to achieve, without specifying any restrictions as to how; these details being left to the lower layer to figure out.

To demonstrate that our method recovers intuitive decompositions in a variety of domains, we

consider in Fig.(5.1), the subtasks uncovered by our method for a few hand-picked values of the decomposition factors. Here, we compute the data matrix $D \in \mathbb{R}^{N \times k}$ for $k = \{4, 9, 16\}$ via Eqn.(3.9) for two navigation domains: the Nested Rooms domain, and the Hairpin domain. A cursory analysis of the results suggests that the uncovered subtasks share a number of interesting properties. Notably, individual subtasks do not correspond to singleton states (like bottle-neck states), but rather to complex distributions of preferred states. Consider for example the subtasks uncovered in the nested rooms domain for the decomposition factor $k = 16$. Semantically speaking, each individual subtask corresponds to a task of the form 'go-to-room-X', where any state in the target room is suitable as a terminal state.

Also, notice that taken together, the distributed patterns of the family of subtasks form an approximate cover for the full task space. This is a corollary result which follows from the fact that the NNMF procedure uncovers a maximally variance capturing subspace, and the data matrix D is derived from the full basis matrix Z. This property of the families of uncovered subtasks is present regardless of the specific value of the decomposition factor k. Another interesting property of our method is that the decompositions uncovered via NNMF, and therefore the subtasks uncovered by our scheme, are incrementally refactored as the value of the decomposition factor k is increased. Importantly this means that the subtasks uncovered for $k = 4$ say, are not a subset of those uncovered for $k = 5$. Rather than simply adding a single subtask to the set as the decomposition factor is incremented, the additional representational capacity is instead distributed evenly amongst the subtasks resulting in a global refactor. It follows that there is no preferred ordering amongst subtasks with some subtasks being uncovered first, or being more valuable amongst the set. Instead, the subtasks

uncovered via our method are optimized with respect to their ability to represent possible tasks in a distributed way. The ability to execute multiple policies concurrently in a graded fashion is therefore of central importance to the discovery methodology.

The subtasks we have considered thus far, both in the Nested Rooms and Hairpin domains in Fig.(5.1), and tacitly in Figs.(3.5, 4.2), have been exclusively in spatial domains. At first glance this may suggest that our discovery scheme is interpretable as form of state aggregation. However, our subtask discovery method is certainly applicable to a much broader class of problems than the sort of simple navigation tasks considered thus far. To demonstrate this fact we consider, in the final experiment in Fig.(5.1), the application of our method to the canonical TAXI domain[55].

The TAXI domain consists of a 5×5 grid in which a taxi driver operates. Some subset of the states are demarcated as special locations from which a passenger might be picked up or dropped off. At each time step the taxi driver acts by electing to moving in one of the coordinate directions, staying stationary, or executing a pick-up or drop-off action. A task in this domain is initialized with the taxi driver at one of the 25 states in the grid, and the passenger at one of the specially demarcated pick-up/drop-off locations. The task is solved when the passenger has been successfully relocated to their final destination at one of the other demarcated locations. To successfully complete such a task, our agent (the taxi driver) must first navigate to the base state where the passenger is currently located, execute a pick-up action, navigate to the base state corresponding to the passengers final destination, and execute a drop-off action. For the specific instantiation of the domain depicted in Fig.(5.1), we have 4 pick-up/drop-off locations, and a single passenger. The agent here operates in the product space of the base domain ($5 \times 5 = 25$) and the possible passenger locations ($\binom{5}{1} = 5$) for

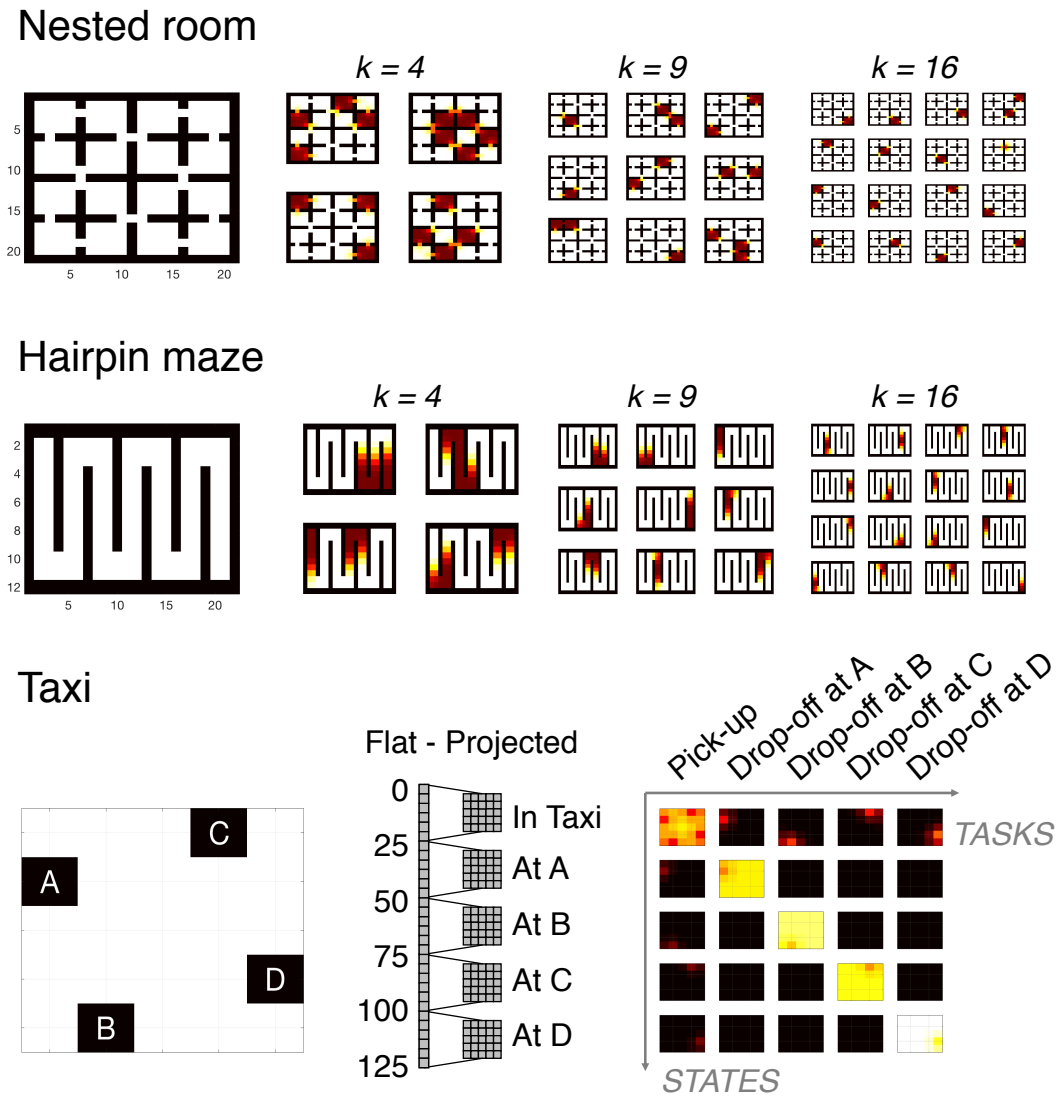


Figure 5.1: Our method uncovers intuitive subtasks in various domains (reproduced from Earle et al.[47]). Nested Rooms and Hairpin: our method uncovers intuitive subtasks in these structured domains. By considering differing domains and decomposition factors, we uncover a number of invariant properties of the family of subtasks. In both domains, and for both decomposition factors, the union of subtasks form an approximate cover for the task space. As the number of subtasks is increased the decomposition is refactored, and each constituent subtask expresses more localized behaviour. Taxi: the subtasks uncovered by our method correspond to behaviours with intuitive semantics, even in non-spacial domains. The middle graphic shows the procedure by which the subtask vector is reshaped for visual clarity. Each column of the right more figure represents a single reshaped subtask, tagged with the corresponding subtask semantic.

a complete state-space of 125 states.

We consider, in Fig.(5.1 - Taxi), a decomposition of the task basis with factor $k = 5$. Each of the resulting subtasks is represented as a vector in $\mathbb{R}^{125}$. For visual clarity, each of the subtask vectors is divided into five equally sized segments. Each segment is a vector in $\mathbb{R}^{25}$ and corresponds to the subset of states associated with the passenger in one of the five possible locations (the four pick-up/drop-off locations and in the taxi itself). Each segment in then reshaped into a 5×5 grid corresponding to a single copy of the base domain. This segmentation and reshaping procedure is depicted in the middle column of Fig.(5.1 - Taxi).

In order to better understand the semantic interpretation of the uncovered subtasks, we focus our analysis on the first column of the reshaped subtasks in Fig.(5.1 - Taxi). It is clear that this subtask places almost all of its mass on the first segment. Within this segment however, all states are rewarded. The segment in which this subtask has placed rewards corresponds to the subset of states in which the passenger is in the taxi. Semantically speaking, this corresponds to a general pick-up action, since it results in the passenger being in the taxi agnostic to the particular location of the taxi itself within the grid. A similar analysis of columns $2 - 5$ reveal that these subtasks correspond to drop-off actions at each of the four demarcated locations. Again, rewards are evenly distributed amongst all states for which the passenger is at the demarcated location. The corresponding subtask is therefore considered to have been solved when the passenger reaches that location, regardless of the position of the taxi within the grid world. Collectively, the subtask library contains policies for picking up the passenger, and for dropping the passenger off at each location.

5.3 A Note on Subtask Equivalence

While there are many important distinguishing features of our discovery mechanism, it also bears thinking about the ways in which the new mechanism is similar to previous approaches. To see the connection between our new approach to subtask discovery and popular alternatives, we must first bring to light an important conceptual difference. Where many other schemes uncover subtasks corresponding to shortcuts between specific states, the subtasks discovered through our approach correspond to complex distributions over broad regions of the task space. By analyzing the boundaries between these regions, we find that the competing approaches are in some sense dual: where our approach finds something akin to strongly connected regions of the task space, previous approaches uncover subtasks through betweenness measures.

To probe this intuition more rigorously, we assign a weight vector to each state[47]:

$$\mathbf{w}_s = \min_{\mathbf{w}} ||Z\mathbf{w} - \mathbf{z}_s||_2^2, \tag{5.1}$$

where $\mathbf{z}_s \in \mathbb{R}^N$ is the desirability function corresponding to a task with a singleton goal state at s, and $Z \in \mathbb{R}^{N \times N_t}$ is the matrix whose columns are the desirability functions for each of the subtasks in the library. The weight vector $\mathbf{w}_s$ can be thought of as the representation of the task desirability function $\mathbf{z}_s$ in the basis Z.

We may then assign to each state a real-valued measure of *instability*, by considering how this representation in the task library changes under state transition. Explicitly we consider the instability

function:

$$g_s = \sum_i p_{is} ||\mathbf{w}_i - \mathbf{w}_s||_2^2, \tag{5.2}$$

which computes a weighted blend of the absolute deviation between the weight vector at the target state, $\mathbf{w}_s$, and those of its neighbours. States for which g_s takes on a high value are considered to be unstable in the sense that their representation in Z is subject to drastic change on state transition. Similarly, states for which g_s takes on a small value are considered to be stable, with their representation in the task library changing only slightly.

In Fig.(5.2) we compare the subtasks uncovered by our method with those uncovered using a popular alternative method based on the *betweenness centrality* of the underlying state transition graph. Specifically, we follow the approach outlined in Şimşek and Barto [111], and assign a score to each state s by counting the number of shortest paths between all state pairs that pass through s. We then take our subtask states to be those with the highest such scores.

We see that the instability function provides a direct way to compare the subtasks uncovered by these competing ideologies. States for which g_s takes on a high value correspond directly to the subtasks uncovered by methods which utilize the betweenness centrality of the underlying transition graph to extract subtasks. Notice that this duality is only approximate. This is due fundamentally to the fact that our new scheme is task dependent while the alternate schemes are task agnostic. And so, if the underlying reward structure were to shift, or the task distribution were to become biased in some fashion, our subtask discovery mechanism would respond to capture this change in task distribution while the alternate methods would not.

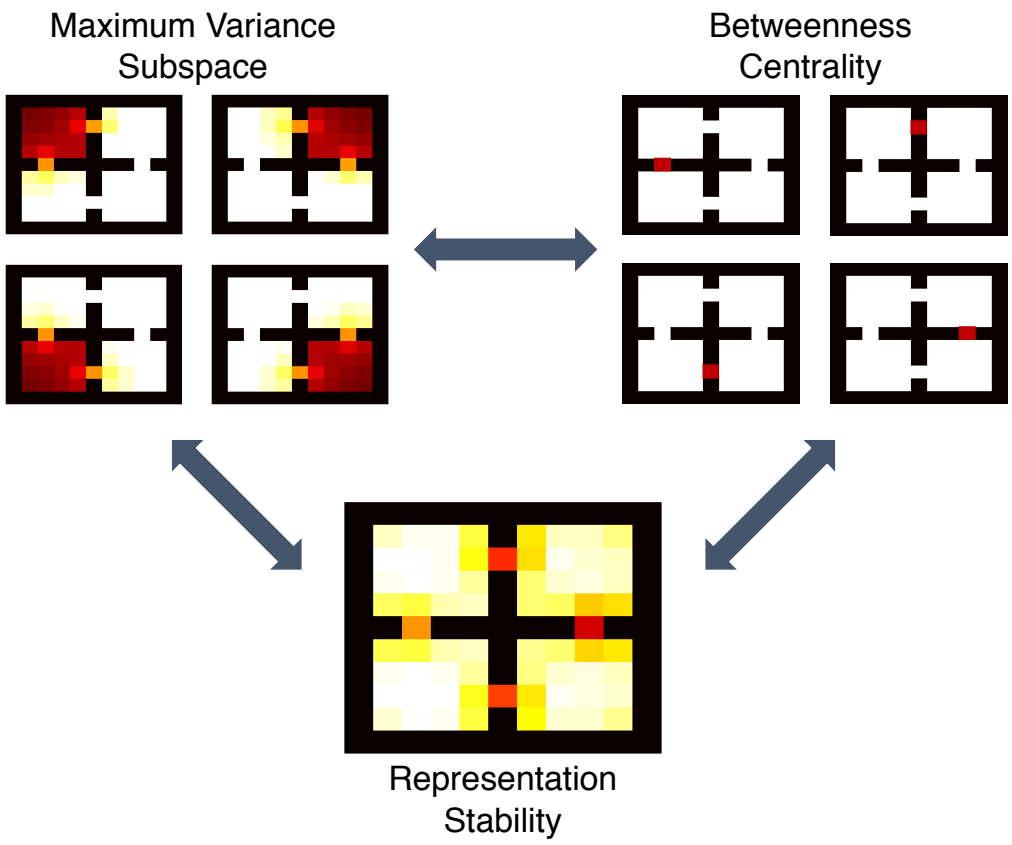


Figure 5.2: A natural duality exists between the subtasks uncovered by our scheme, and those typically uncovered by other methods. Maximum Variance Subspace: the subtasks uncovered by a rank 4 decomposition of the task basis Z with NNMF. By inspection, we have that each of the subtasks corresponds to one of the four rooms in the domain. Importantly a single subtasks is seen to place boundary rewards at multiple states in the domain. Note that this is distinct from a subtask with a singleton goal state at the center of the room. Betweenness Centrality: subtasks discovered by considering the betweenness centrality of the underlying transition graph correspond to doorway states. These are the subtasks typically uncovered by task agnostic methods. Representation Stability: by computing the instability function g_s for each state in the domain, it is immediately clear that unstable states, those for which the representation in Z changes starkly, correspond precisely to doorways. The instability function is computed with respect to the task library uncovered by the rank 4 decomposition of the task basis Z.

The ability to formally evaluate the contribution of a subtask in the context of a library of tasks, and the corresponding notion of subtask equivalence is especially important. Firstly, this metric provides a mechanism for incrementally pruning the set of available subtasks; ensuring that the agent is not hampered by deleterious subtasks. Secondly, as touched on in Section.(3.3), this provides a measure for the optimal number of subtasks.

The fact that the subtasks uncovered through this procedure are task dependent is a particularly important property of the new discovery mechanism which appears to be aligned with the way in which humans and animals develop their internal representations. It would seem to be a matter of course that a person directed to perform two different tasks, with the first being attempted much more frequently than the second, would develop a more fine grained representation of the first task. This fact should be evidenced in their improved ability to adapt to minor perturbations in the first task over the second, and their improved ability to transfer learning to a third task that is more similar to the first, than to the second. If any bias exists in the task distribution, it follows that the subtask representation should be similarly adjusted; freeing up representation capacity from less likely tasks to provide higher resolution and better estimates for more likely tasks.

5.4 Autonomous Construction of the Subtask Transition Structure

With the subtask discovery process automated by the factorization mechanism above, the final manual piece in the construction of the higher layer LMDP lies in the specification of the subtask transition structure $P_{\mathcal{IT}}$. In general, there is no restriction on the choice of subtask transition structure,

although some choices will result in abstractions which are more suitable for rapid learning. Here we propose a method for automating this step by forcefully aligning the subtask transition structure with the subtask policies themselves.

In Section.(4.4) we established that the pseudo-similarity transform which arises by taking $B \approx Z$ is a numerically efficient choice, since the optimal controller is likely sparse in that basis. These observations suggest, both from a computational and intuitive perspective that the low-rank approximation to the task basis D is a natural choice for the subtask transition structure. Explicitly we take $P_{\mathcal{IT}} = \alpha D$ for some $\alpha \in \mathbb{R}^+$. Note that D must first be normalized row-wise in order to constituent a valid transition structure. Note that we have introduced a new scaling parameter to the integrated learning problem whose dynamics are now:

$$P = \left[\begin{array}{c:c:c} P_{\mathcal{II}} & P_{\mathcal{IB}} & \alpha D \\ \hdashline \mathbf{0} & I & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{0} & I \end{array}\right]. \tag{5.3}$$

This alignment procedure positions the hierarchical learning problem as a multiscale learning task in which the agent learns both a base layer controller, and a higher layer controller, but over different time scales. These time-scales are determined by the subtask transition structure $P_{\mathcal{IT}} \in \mathbb{R}^{N \times N_t}$. By taking $N_t << N$, the abstracted LMDP has many fewer states than the base task, but the agent also transitions into the higher layer less frequently. When the multiscale nature of the control hierarchy aligns with the multiscale nature of the environment, the agent is well positioned

to expedite learning.

One that would have the fruit must climb the tree

-Thomas Fuller

6

Discovering and Using Deep Control Hierarchies

6.1 Introduction

A key insight to be gleaned from the explosive recent success of deep learning systems in perceptual domains, is the value of constructing representations through the recursive composition of simple

basis elements. Indeed, adopting the view point that compositionality is a fundamental component of any scalable learning systems was a primary motivation for our selection of LMDPs as our framework of choice. In Chapter.(4) we leveraged this compositionality to construction a fully online hierarchical control architecture. Functionally, the compositionality of the MLMDP allows the agent to execute a multiscale and concurrent action selection paradigm; periodically choosing high-level actions which bias primitive action selection in pursuit of the high-level desire. The control hierarchies explored thus far have been only one layer deep. Here, we bring together all of the ideas we have explored in the development of a recursive procedure through which the agent may autonomously construct arbitrarily deep control hierarchies. To the best of our knowledge, this ability is unique amongst existing deep RL methods.

The higher layer transition dynamics defined in Eqns.(4.3, 4.4), constitute a transformation between a 'base' LMDP and its higher layer abstraction, $L^n \rightarrow L^{n+1}$. This transformation depends crucially on the subtask transition matrix $P_{\mathcal{IT}}$, which would typically be manually chosen by a designer. The construction of the abstracted LMDP is otherwise fully autonomous. However, we noted in Section.(4.4), that the equations defining the higher layer transition dynamics could also be interpreted as a transformation to the inverse eigenvalue problem with a suitable shift. Furthermore, when the subtask transition matrix is taken to be an approximation to the eigenbasis, the higher layer dynamics are seen to be a sort of similarity transformation of the inverse problem in which the eigenvector we seek is sparse. This observation suggests that the subtask transition matrix might be constructed directly from the eigenbasis itself. This insight was realized in Section.(5.4) where the subtask transition structure was defined as the low-rank approximation to the eigenbasis recovered

via NNMF. In so doing we remove the final manual step in the abstraction process. We therefore have a fully autonomous procedure for mapping between a base LMDP and its higher layer abstraction.

With this tooling in hand, it is instructive to view the LMDP as a black-box control module which may be stacked by recursively associating the states of one module with a set of controllers over another. This association is defined via the subtask transition structure. Considered as a whole, a stack of LMDP modules constitutes a deep hierarchical controller, in which higher layers correspond to abstract representations of lower layers. The conceptual benefits of this sort of control architecture are clear.

It bears mention that the presence of depth is ubiquitous in the hierarchical organization of cortical structures in the human brain[117,118,119]. Moreover, deep hierarchical control architectures appear to provide a natural representation for many real-world tasks of interest. As a motivating example, we consider a navigation task in which a person based in Johannesburg, South Africa, seeks to meet up with a friend in a coffee shop in Paris, France. A plan to achieve this sort of task would typically be stated at varying levels of abstraction, rather than being specified in terms of motor control primitives - first walk to the corner, then catch a cab to the airport, then take a flight to Paris, etc. Of course, in order for an agent to execute planning in this way, its internal control architecture must permit a representation of these abstractions.

In this chapter we describe a recursive strategy for constructing arbitrarily deep control hierarchies by stacking MLMDP modules. Furthermore, by leveraging the ideas developed in preceding chapters, the recursive procedure we develop can be carried out in a fully autonomous fashion by

the agent. Learning solely from direct experience in the domain, the agent is able to construct a deep compositional representation of its actions, and periodically leverage this representation to improve subsequent learning. In addition, we show that the recursive construction of the control hierarchy may be interpreted as a form of deep skills discovery, and demonstrate that the skills uncovered naturally correspond to the multiscale description of the domain. Finally, we demonstrate the quantitative improvement in learning with deeper control structures.

6.2 The Benefits of Depth

Far from being simply a nice-to-have feature, a deep hierarchical representation of the agent's action space would appear to be essential to efficiently capture the multiscale nature of real-world control tasks. Indeed many real-world tasks unfold over a wide range of spacial and temporal scales. In order to efficiently and flexibly plan against these sorts of tasks, abstracted concepts must be realizable within the agent's control representation.

While a strong case has been made for the necessity of a hierarchical control architecture, we have said little of the structural properties that such an architecture should exhibit. If each subsequent layer of the hierarchy constitutes an abstraction of the preceding layer, then a natural trade-off presents itself in the form of the balance between abstraction and resolution. Recall the single layer hierarchy considered in Section.(4.3): there, and more generally in our construction, the number of subtasks we choose is a good proxy for the degree of abstraction the higher layer controller is able to achieve. As we allow for more subtasks, the higher layer LMDP is able to revolve finer details in

the lower layer tasks it is trying to represent. However, each of the higher layer states corresponds to a more narrow representation of the task space. Alternatively, as we restrict the number of subtasks, the higher layer LMDP is more course-grained and loses the ability to represent the details of the lower layer tasks. But in so doing, each of the higher layer states corresponds to a broader representation of the task space capturing the shared features of a set of tasks, and therein constituting a more abstract representation. In addition, the learning problem presented by the higher layer is itself typically more difficult as the number of states is increased.

The trade-off between abstraction and resolution was empirically demonstrated in Fig.(4.2) in which the agent's performance in the online MLMDP improved as the number of subtasks decreased. Crucially, this trade-off is affected by the presence of depth, and cannot be simply determined layer-wise. To see this, it is instructive to view the effect of the higher layer as reducing the effective planning horizon at the base layer. This has the effect of improving abstraction, since signals can propagate more efficiently. Now suppose that we have a two layer control architecture for which we have found the optimal number of subtasks. By introducing a third layer to the control architecture we improve abstraction at the second layer. Alternatively we may increase the number of states at the second layer, while achieving the same degree of abstraction. This in turn increases the effective resolution at the base layer. By induction, we conclude that adding subsequent layers to our control architecture allows the agent to achieve a globally optimal trade-off between abstraction and resolution which is strictly better than the layer-wise solution.

Another important observation is that real-world tasks modeled as MDPs will typically have terminal rewards at many different states, rather than the singleton 'goal state' setup commonly con-

sidered in the RL literature. This follows from the fact that real-world tasks are simply agnostic to the primitive state and action space representation, instead corresponding to some abstract state. To solidify this intuition, consider a reaching task in which an agent with a multi-joint robotic arm is tasked with placing the end effector at some target location. Here one can imagine that there are many sets of joint-angles that would correspond to the same position for the end-effector. Said differently, the task is such that the same terminal reward should be received for many primitive states in the base MDP. The terminal rewards distributed over a set of primitive states are often clustered together in such a way that a suitable transformation is able to extract them as corresponding to a small number of states in the new representation. A good hierarchical structure should permit a representation of this clustering of terminal rewards as corresponding to just a few higher layer states.

It is also instructive to notice that these deep hierarchical representations allow for tasks to be solved in a more numerically efficient manner. Viewed in this light, the hierarchical construction is a pure algebraic transformation to a representation better suited for the learning problem. This view allows us to assess the structure of the induced hierarchy simply in terms of how efficiently the resulting learning problem may be solved. However, this view may be limiting given that few details of the task ensemble are typically known a priori. Nevertheless, one may consider bounds on the optimal construction of the hierarchy in an idealized setting, and infer some useful heuristics for establishing the number of subtasks to use at each layer to obtain a reasonable balance between abstraction and resolution.

Absent any domain specific information, in practice we find that a useful heuristic for determining the depth and breadth of the control hierarchy can be succinctly stated in terms of the *factor*

ratio. We define the factor ratio as $k = {}^{N_i}/_{N_t}$, where $N_i, N_t \in \mathbb{Z}^+$ are the number of interior states and the number of subtask states respectively. Intuitively the factor ratio gives a proxy measure of the abstraction / resolution achieved at a given layer. By way of example a factor ratio of say $k = 3$ implies that there is one subtask state for every three interior states, and that the learning problem at the higher layer is therefore one third of the size.

Heuristically then, the factor ratio for a given problem is determined from the relation $k^k = N_i^\circ$, where N_i° is the number of interior states at the base layer. Intuitively, this sets the factor ratio such that the abstraction / resolution is the same between any two adjacent layers of the hierarchy. Suppose we have $N_i^\circ = 256$ state: then, given our heuristic, the factor ratio is calculated to be $k = 4$, since $4^4 = 256$. This means that we would construct a 4-layer hierarchy with the number of interior states specified layer-wise as $(256, 64, 16, 4)$. Notice that the depth of the hierarchy is equal to the factor ratio, and that the number of interior states at a given layer l is equal to ${}^{N_i^\circ}/_{k^l}$.

6.3 Recursive Construction of the Control Hierarchy

As noted above, by autonomously constructing the subtask transition matrix layer-wise as a low-rank approximation to the task basis, we have developed a fully autonomous mapping between an LMDP and its higher layer abstraction. This abstraction protocol is made possible by the coming together of the disparate elements developed in preceding chapters.

The recursive procedure begins by first obtaining a task basis Z at the base layer LMDP. Next a low-rank approximation to the task basis is uncovered via NNMF, $Z \approx DW$. The rank of this

approximation is determined by analyzing the spectral properties of the task basis via one of the methods described in Section.(3.3), or via the heuristic described above, $N_t = {}^{N}/_{k}$, where $k^k = N$. We then construct an augmented LMDP which incorporates the new subtask states via Eqn.(4.2). Next the subtask transition matrix is set to be proportional to the low-rank approximation $P_{\mathcal{IT}} = \alpha D$. With the subtask transition structure in hand, we define an abstracted LMDP over the subtask states via the consistency Eqns.(4.3, 4.4). The abstracted higher layer is itself a bona fide LMDP, and the process may therefore be applied recursively.

Note that the procedure outlined above describes an idealized version of the recursion in which the agent has access to an explicit representation of the task basis at each layer. This corresponds to a learning paradigm in which the hierarchy is constructed through some offline process, and the agent need only execute online learning on a new set of tasks, while the hierarchical architecture itself remains static during the course of learning. One might imagine an alternate version of the algorithm in which the hierarchy itself is incrementally updated online as the agent accrues experience in the domain. In this setting the agent will not have access to an explicit task basis and will instead need to estimate the low-rank approximation based on a sequence of samples as in Zhao et al. [120]. The incremental growing of the hierarchy is a prospect left for future work.

6.4 Deep Skills Discovery

In Section.(5.2) we were able to interpret the column vectors of the low-rank approximation to the task basis at the base layer as corresponding to the desirability functions for a set of subtasks. This

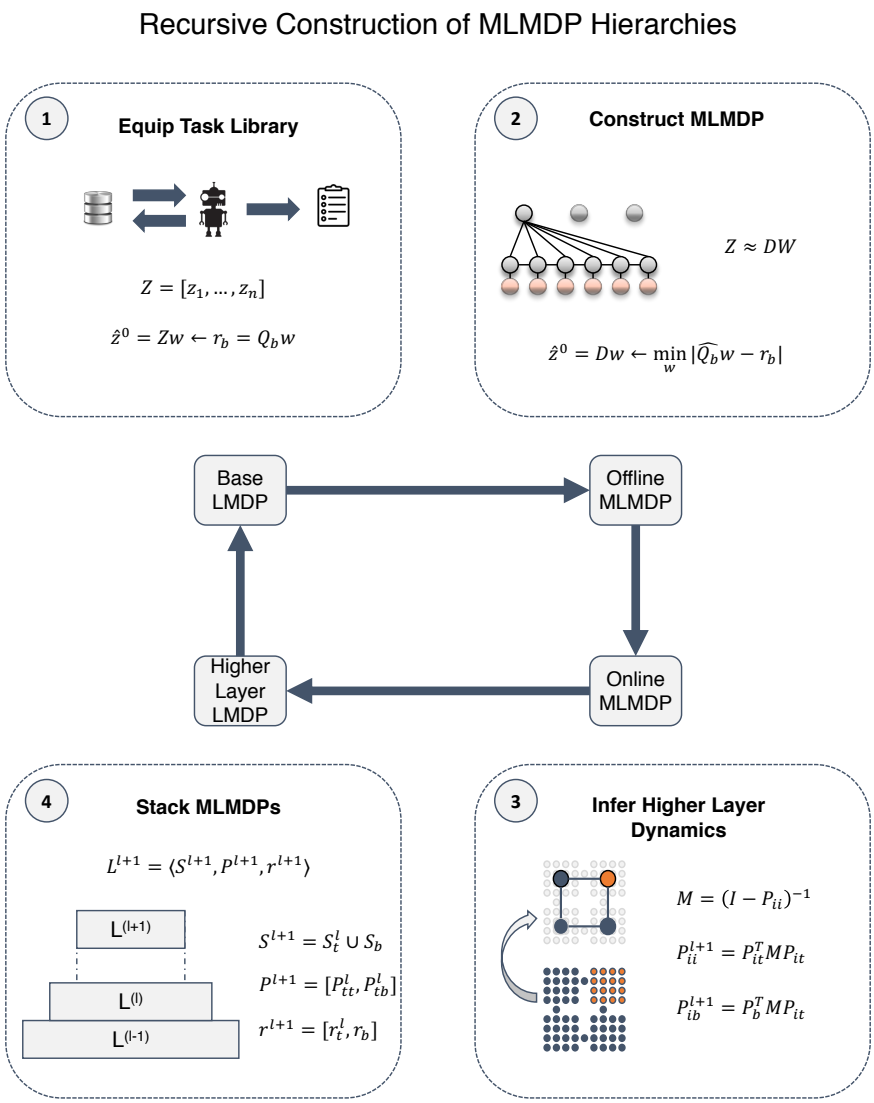


Figure 6.1: Recursive construction of deep control hierarchies. 1. Equipping the Task Library Firstly, the agent is equipped with the task basis for the current layer. 2. Constructing the MLMDP A low-rank approximation to the task basis is uncovered via NNMF. The rank of the approximation is determined by analyzing the spectral properties of the task basis. The base LMDP is then augmented with a set of additional terminal boundary states corresponding to the available subtasks. Transitions into these new subtask states are governed by the subtask transition matrix, which is set to be proportional to the low-rank approximation. The agent operates within the augmented LMDP, periodically receiving guidance from the higher layer by transition into subtask states. 3. Inferring Higher Layer Dynamics A new learning problem is constructed over the subtask states by defining the transition and reward structures in such a way that they are consistent with the base LMDP with respect to the subtask transition structure. 4. Stacking MLMDPs The abstracted LMDP simply constitutes another layer in the stack of hierarchical controllers. This sets the stage for the recursive construction of further layers of abstraction.

mode of analysis proved fruitful in providing some important intuitions about the nature of the higher layer abstractions. Continuing in this spirit, we considered the subtasks uncovered layer-wise through the recursive application of the abstraction procedure. At each layer, a subtask is a column vector of the approximation to the task basis in $\mathbb{R}^{N^l}$, were N^l is the number of states at that layer. In order to visualize higher layer subtasks intuitively, we simply project these vectors back into the base domain. This is functionally equivalent to recursively determining the set of lower layer states preferred from the perspective of the higher layer.

Our intuition is that the hierarchical control architecture is particularly well suited to capture the multiscale structure of tasks when it is present. To directly probe this intuition we consider the subtasks uncovered in the construction of the control hierarchy in a domain which is known to exhibit multiscale structure by construction. In Fig.(6.2) we consider a domain inspired by the emergent multiscale structure of a city. Our city is comprised of 3 separate communities, each of which is itself comprised of 5 separate houses, each of which is composed of 4 rooms, which are ultimately realized as a 4×4 grid of base states.

In Fig.(6.3) we construct a 4 layer deep hierarchy using the recursive procedure described in Section.(6.3) above. The number of subtasks at each layer is hand picked to represent the specific multitask structure of the city domain. Explicitly, we take $[N_t^0 = 960, N_t^1 = 60, N_t^2 = 15, N_t^3 = 3]$. In the left column of Fig.(6.3) we plot individual subtasks for a shallow hierarchy constructed with a different number of subtasks. This corresponds to a single layer of abstraction with differing tradeoffs between abstraction and resolution. In the right column of Fig.(6.3) we plot individual subtasks for a single deep hierarchy constructed with the subtask signature specified above. At each layer we

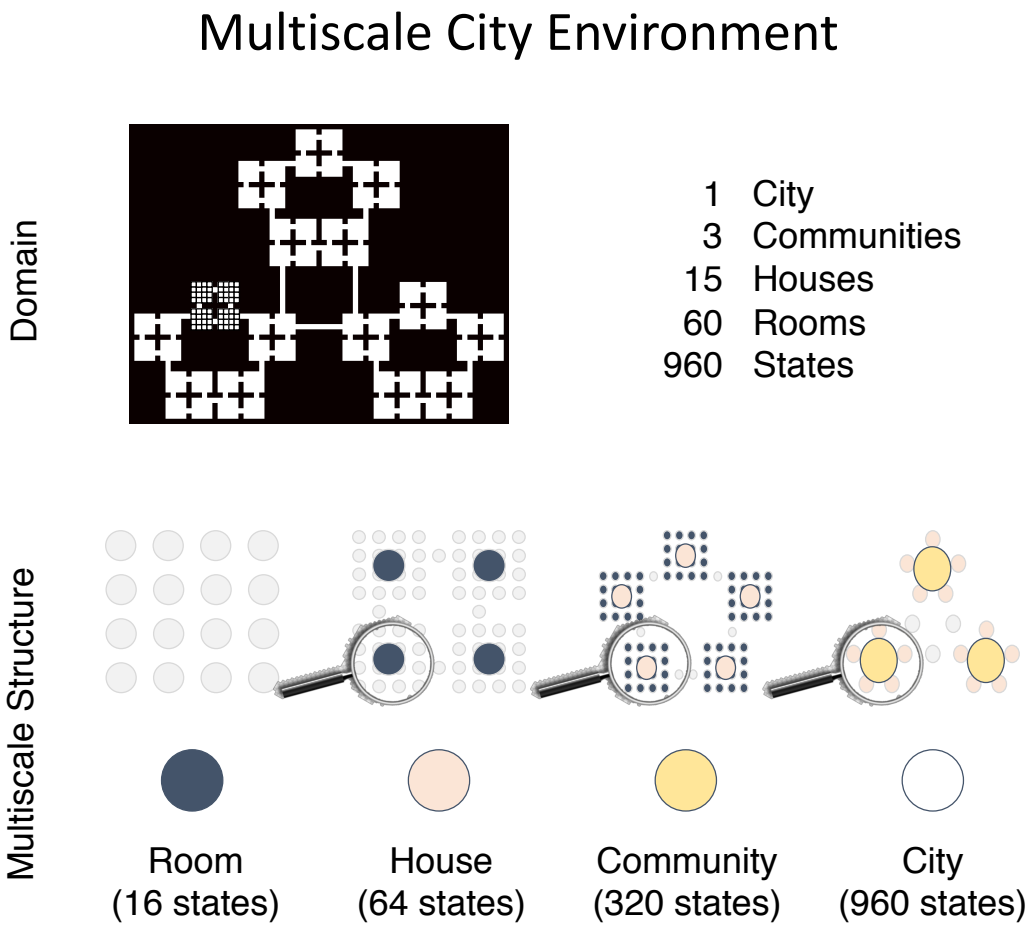


Figure 6.2: The city environment. We consider a multiscale grid-world environment in which the full domain consists of 3 identical communities, each of which consists of 5 houses, each of which consists of 4 rooms which are ultimately realized as a 4×4 grid of base states. This domain provides a useful test bed for subtask discovery methods, since it has a compositional multiscale structure which captures a number of important challenges for any subtask discovery methodology.

project a single subtask back into the base domain.

The deep control hierarchies we construct typically uncover interpretable and intuitive subtasks at all layers of the hierarchy. A cursory analysis of the subtasks uncovered in Fig.(6.3) provides evidence for the fact that higher layer subtasks represent progressively more abstract representations. As we step upwards through the hierarchy we see that subtasks correspond to individual states, then rooms, houses, and whole communities respectively. Note the side by side comparison of the shallow subtasks with the higher layer subtasks. Although both sets of subtasks are extracted from approximations of the same rank, those uncovered with increased depth in the hierarchy are qualitatively different from those uncovered with a single layer of abstraction. A qualitative assessment of the subtasks uncovered in the deep hierarchy show them to be highly intuitive.

Of course, the linearity of the objective function in the LMDP formalism coupled with the particular construction of the integrated learning problem means that every hierarchical policy is ultimately realized as some linear combination of the library policies at the base layer. If these linear maps ultimately collapse, it raises the question of what, if anything, is to be gained from a deep representation. As alluded to above, the primary value in a deep representation is that it shifts the abstraction resolution trade-off such that the agent is capable of more fine-grained control at the lower layers without sacrificing abstraction at the higher layer. In particular this imbues the agent with the ability to perform flexible and rapid replanning.

In addition, although there is no doubt some objective function for which the decomposition at the base layer will recover the decomposition obtain through the hierarchy, it is not clear what this objective function is. Fundamentally, this is due to the fact that the decomposition ought to take

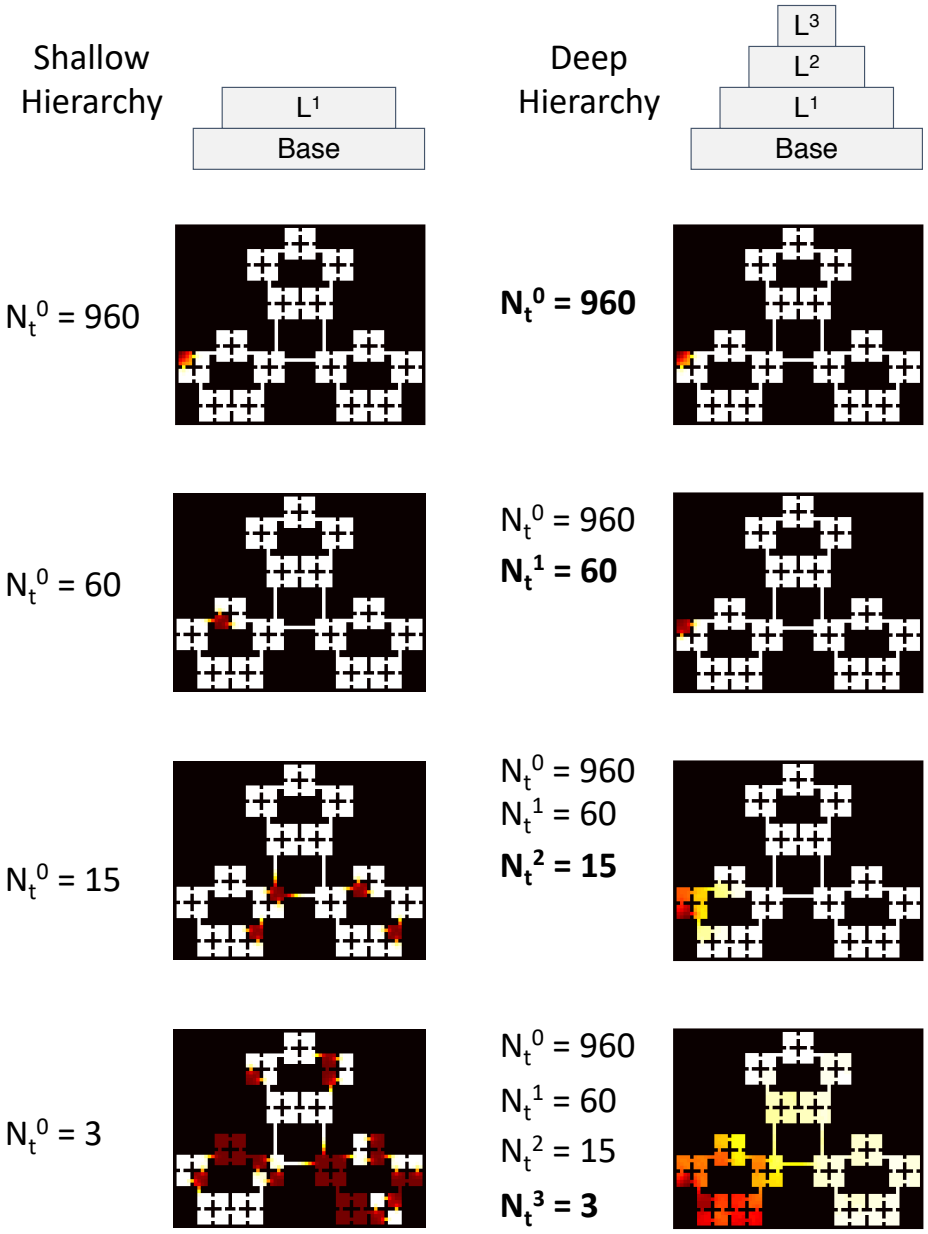


Figure 6.3: Deep subtask discovery uncovers qualitatively different subtasks from shallow subtask discovery. Each of the plots above represent a projection of the desirability function for a single subtask back into the base domain. The 'breadth' of the subtasks increases in both the shallow and deep hierarchies as the number of subtasks is decreased, with a single subtask rewarding many more states. However, the distribution of those state in the domain differs significantly. Subtasks uncovered in the deep hierarchy capture the natural multiscale structure of the domain, with subtasks at progressively deeper layers representing more abstract behaviours. Subtasks at the lowest layer represent individual states, whereas higher layer subtasks represent rooms, houses, and communities.

into account both the transition dynamics and reward structures, for both the interior and boundary states. To see why the hierarchical decompositions produces intuitive results, it is instructive to view successive layers as undergoing two simplifying transformations via Eqns.(4.3, 4.4). The first is a relaxation step corresponding to the multiplication by the diffuse steady-state dynamics, the second being a compression step corresponding to the projection into the subtask states via $P_{\mathcal{IT}}$. The hierarchical decomposition can therefore be viewed as a process of successive decomposition / simplification steps which take into account both the reward and transition structure of the task.

The fact that higher layer subtasks correspond to progressively more abstract representations seems to fall-out naturally from our recursive construction. It bears mentioning that this is typically not the case. By way of example, it is not at all clear how one might begin to define an option over a set of options for example, or indeed how progressively more course-grained options might be uncovered directly. The recursive construction therefore corresponds, independently, to a deep skills discovery mechanism capable of extracting the multiscale nature of real-world tasks.

6.5 Learning with Deep Hierarchies

In this section we consider a final expository demonstration of the quantitative benefits of depth in the control hierarchy. In order to isolate the property of depth we consider, in Fig.(6.4), a simple 1-dimensional ring domain. The fact that the domain is perfectly symmetrical shields our analysis from any structure artefacts in the environment. In particular we can be sure that any results we observe are not simply a peculiarity of the geometry of the domain. Furthermore, the domain is fully

extensible, and as such the results are scale invariant. And so, by construction, our results should hold if the domain is uniformly scaled up. The domain we consider is a variant of the canonical 'n-chain' domain used in Osband et al. [121]. Indeed the 'n-chain' domain is often chosen for these sorts of analyses for precisely the reasons stated above.

Since the domain contains no additional structural cues from which a preferred decomposition ratio might be extracted, we simply use the decomposition heuristic described above; namely $k^k = N$. Note that this heuristic implies a rather dense decomposition. Even with 10 layers of hierarchy, the compression between layers is only of order $1 : 10$. Nevertheless the value of k^k grows rapidly. This suggests that very deep control hierarchies may be unnecessary in practice. As an aside, it is an interesting curiosity that 10^{10} is approximately the number of neurons present in the human brain, and 10 is approximately the number of layers of the deepest cortical structures in the human brain. To the extent that biological example has been a guiding light, it is instructive to note that the depth of the hierarchical structures present in the human brain are comparable to those that we would uncover heuristically.

In Fig.(6.4) the agent acts in an instantiation of the ring domain with 200 interior states. The agent is tasked with solving 4 tasks presented sequentially. The 4 tasks correspond to reaching singleton goal states, each of which is positioned one quarter the way around the ring. The tasks are presented in the predefined order $(t_{q_1}, t_{q_3}, t_{q_2}, t_{q_4})$, where the subscript q_i refers to the task that is i quarters of the way along the ring. The agent is permitted to perform 150 epochs on each task after which it resets its estimate of the boundary reward structure, and its base controller. We consider 3 separate realizations of the control hierarchy in addition to the flat agent. In the first case, the agent

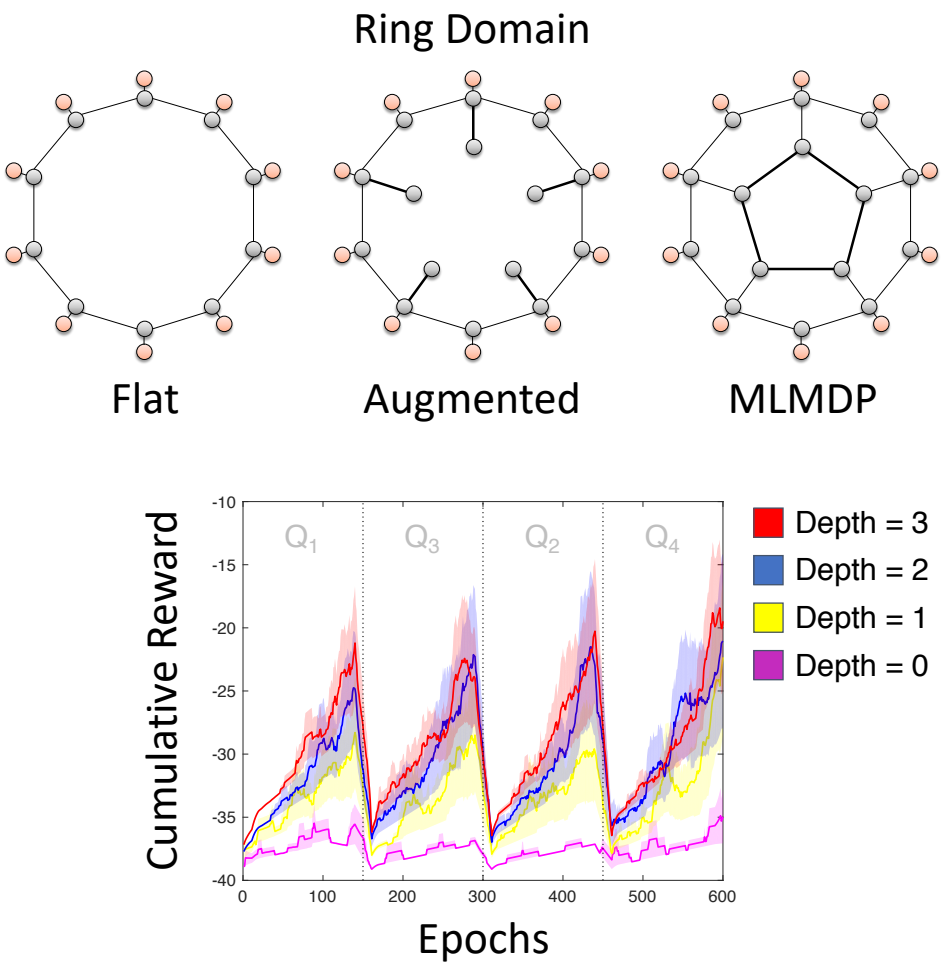


Figure 6.4: Deeper control hierarchies yield quantitative performance improvements. The Ring Domain: the domain consists of a symmetrical 1-dimensional ring of states. The domain is a symmetric modification of the 'n-chain' domain used in Osband et al. [121]. The symmetric and extensible nature of the domain makes it a good expository environment since it is void of extraneous features. Deep Online Multitasking: deeper control hierarchies can impart quantitative improvements in performance. The improvement as the agent moves from a single layer control hierarchy to a two layer control hierarchy is significant. Although a performance improvement is still observed as the agent transitions from a two layer hierarchy to a three layer hierarchy, the absolute improvement is less pronounced.

is equipped with a single layer hierarchical control architecture with a factor ratio of $k = 3$. This corresponds to $N_t = 66$ library subtasks. In the second realization, the agent is equipped with a 2 layer deep control hierarchy, again with a factor ratio of $k^0 = k^1 = 3$. This corresponds to $N_t = 66$ library tasks at the first layer, and a further $N_t = 22$ library tasks at the second layer. Continuing in this fashion, in the third and final realization, the agent is equipped with a 3-layer control hierarchy corresponding to a subtask signature of $N_t = (66, 22, 7)$. Details of the experimental implementation can be found in Appendix.(A.6).

In a consistent display of previous results, all instantiations of the hierarchy perform significantly better than the flat policy. In addition, we observe the hypothesized performance improvement with increased depth of the hierarchy. The performance delta is significant as the agent transitions from a single layer hierarchy to a two layer hierarchy. The subsequent performance improvement in moving from a two layer hierarchy to a three layer hierarchy is less pronounced. This is a specific artefact of the problem size. However, given the scale invariant nature of the domain, it stands to reason that a similar performance multiplier would be observed as the size of the domain is increased proportional to the factor ratio.

In addition to the quantitative benefits of deep control architectures, these models also hold promise as providing a descriptive framework for complex emergent behavioural phenomena. Already we have seen that where the lower layers of these architectures exhibit behaviours representative of simple motor primitives and localized reflexes, higher layers reflect spatially and temporally abstract actions. It may be argued that these higher level abstractions corresponded to common planning symbols. Consider for example the higher layer abstractions abstracted in Fig.(6.3). A human

navigating in a similar domain would likely construct a symbolic plan involving navigation to the correct city, house, and finally, room. One may hope that by further extending the depth of the control hierarchy these models may yield descriptions of yet more abstract elements of, for example, social behaviours.

7

Conclusion

There is little question that the rampant success of deep learning methods in the last few years has revolutionized the field of machine learning. Underpinning the success of these methods is the special manner in which deep neural networks transform their inputs, such that the deep representations are learnable and generalize. Typical neural architectures rely on the recursive, combinatorial composition of simple representations to autonomously extract deep hierarchical representations.

In this work we have sought to apply these insights to problems in reinforcement learning, in the pursuit of an action selection protocol that exhibits a similar recursive compositional structure. The inherently unstructured nature of general MDPs prevents the sort of composition required, and so we sought a framework in which a basic sort of compositionality was permitted *ab initio*. This thesis presents a method for the autonomous construction of deep hierarchical control architectures, within which an agent selects actions in precisely the recursive compositional manner we sought to achieve.

7.1 Summary of Contributions

The primary contribution of this thesis is the development of an algorithm for the recursive extraction of deep compositional control hierarchies which may be implemented in a fully autonomous fashion, and the associated execution protocol allowing the agent to perform online learning within this control structure. The method was presented in a modular fashion.

An offline multitasking method: we first considered an offline multitask learning paradigm in which the agent was incrementally provided with new tasks. Each time the agent was presented with a new task, it was provided the reward structure for that task up front. In this setting we developed a new multitask learning method capable of zero-shot optimal performance. This was achieved by leveraging the boundary state compositionality permitted in the LMDP formulation to formally define a bona fide basis for the task space. Critically, the optimal policy for *any* task in the domain has an exact distributed representation within the basis. We then leveraged this fact by equipping the

agent with an external memory bank containing a representation of the task basis, and allowing the agent to query this external memory when faced with a new task. The method has a bi-partite execution protocol in which the agent first determines a preferred initial policy by querying the external memory, and thereafter performs standard z-learning. The guaranteed optimal compositionality of the LMDP implies that the initial policy is also the optimal policy. We highlighted the fact that the space of tasks within a domain is susceptible to a combinatorial explosion, so that even in a toy rooms domain (of $\approx$ 100 states) the agent might face around 10^{30} prototypical tasks. Critically, our agent is able to perform zero-shot learning on all 10^{30} tasks while equipped with an external memory of just $\approx$ 100 basis tasks.

Results in source task selection and transfer quality: in addition, the conceptual shift to thinking about a basis for action selection allowed us to recast a number of difficult questions in transfer learning as questions in linear algebra, for which solution methods are well established. In particular we explored the question of subtask selection, and the question of the quality of transfer that could be expected with the new method. In the parlance of linear algebra, the problem of subtask selection can be framed as the problem of finding a low-rank approximation to the full task basis. The presence of the additional non-negativity constraint on the task basis made non-negative matrix factorization a natural choice for the decomposition procedure. Another important component of subtask selection is determining the correct number of subtasks to include. We presented two quantitative approaches for choosing the number of subtasks; the first based on the inflection in the approximation-error curve, and the second based on the spectral analysis of the task basis. Finally, we invoked the convexity of the transformed objective function to assert that a weakly monotonic

relationship existed between the initialization value, and the learning speed. From this assertion we conclude a number of important results. Firstly, the quality of transfer is necessarily monotonic in the number of available subtasks. Secondly, a unique optimal initialization exists within the task library, and that it may be readily uncovered. And finally non-negative transfer is guaranteed when the initialization in the task library is better than random.

A shallow online multitasking method: armed with the insights from the analysis of the offline method, we turned our attention to the development on an online version of the new multitask method. In this setting the agent is not provided with the reward structure for the target task up front, and the task basis must be incorporated into the learning procedure in a more dynamic way. Rather than query the external memory just once to initialize the new policy, the agent is instead permitted to query the external memory multiple times through its trajectory, at each stage receiving guidance from the subtask library in accordance with the agent's running estimate of the boundary reward structure. This dynamic execution protocol is formalized in the development of an augmented LMDP in which the new set of terminal boundary states are added to the base LMDP; each corresponding to one of the library subtasks. A transition into one of the new boundary states corresponds to a decision by the agent to receive updated guidance from the subtask library. We further leveraged the compositionality of the LMDP by allowing the library subtasks to act concurrently in a weighted blend. This blend is determined by first constructing an abstracted higher layer LMDP task over the new subtask states, and then solving this abstracted LMDP to obtain an optimal blend of higher layer states. The mapping between the base LMDP and the higher layer LMDP is controlled by a new subtask transition matrix which is chosen by a designer. Finally we showed that

with the right choice of the subtask transition matrix the abstracted higher layer LMDP could be viewed as a special transformation of the base LMDP in which the resulting eigenvalue problem can be solved in a numerically efficient way.

A deep online multitasking method: stepping back, in Chapter.(6), we argued that the augmented LMDP could be viewed as a black box control module, and that the subtask transition matrix could be seen as simply defining a mapping between two such modules. These abstract control modules may then be stacked such that the output from one module specifies a set of preferred states for the module below it and so on. In this way we were able to construct deep compositional control hierarchies in which the agent could follow an identical execution protocol to the shallow method. Finally, by automating the construction of the subtask transition matrix at each layer, the recursive algorithm may be implemented in a fully autonomous fashion allowing an agent to uncover a powerful multitasking control architecture that is both flexible and extensible.

7.2 The Applicability of Linear Models

A valid criticism of the proposed method is its reliance on the drastically simplified Linearized MDP framework. Indeed, LMDPs constitute a very specific subclass of MDPs which may not be suitable for modeling many phenomena of interest. Nevertheless, we argue that the method has value on two distinct fronts: firstly, as a descriptive and analytical tool, and secondly as a directly applicable method for the set of problems which are well approximated by an LMDP embedding.

In the development of the new method we made theoretical progress on a number of important

questions in transfer learning - questions which have otherwise been considered almost exclusively through empirical means due to the intractability of the problem formulation. Providing a theoretical basis for transfer learning in this approximate setting is an important step made possible by our use of a simple linear model. While it remains to be seen how much of the insight and intuitions gleaned from this analysis ports to the general setting, understanding the nature of transfer in this simplified setting has intrinsic value. It follows that the simplified model is, by construction, free of any extraneous degrees of freedom, the properties exhibited by this method are therefore independent of the specific features of more general models. In this way the analysis of these simplified frameworks helps to tease out key intuitions, and unpack the underlying concepts which drive behaviours in more complex models.

While there are no doubt many real world problems which cannot be well approximated by an LMDP embedding, there are many real world tasks that can be. The convex relaxation over the action space which allows for the linearization can be interpreted as an assumption that the optimal behaviour satisfies a minimum energy solution. This presupposes that there is a stationary, action-free, behaviour, and that all actions which drive the system away from this stationary behaviour pay a proportional cost. Encouragingly, this assumption is naturally satisfied in a number of real world systems. In practice the new method may form the base of a more complex algorithm in which the approximate initial behaviours are determined with our compositional linear system before being fine-tuned with some nonlinear controller.

In addition to the general analytic tractability of the linear formulation, the LMDP permits a specific form of guaranteed optimal composition. We argued strongly that composition was a critical

property of any scalable RL algorithm, and indeed it appears as a fundamental component of the new method.

The use of deep non-linear neural function approximators has brought phenomenal success in the last few years in a wide variety of perceptual domains. It is only natural then that these same function approximators be applied to RL problems, and indeed this has been the case with some notable successes. Nevertheless, it is not always clear that this complex machinery is actually required to achieve this level of performance. Indeed, simple linear function approximation is often sufficient to achieve near state-of-the-art results on a number of challenging domains[122]. These sorts of observations are promising, in that they provides further evidence that the virtues of linear models may yet be sufficient to offset their restrictions on an even broader set of tasks.

7.3 Possible Extensions

In this thesis we have remained within the confines of an explicit LMDP representation. Much of our analysis relied heavily on our ability to leverage both the special properties of the formulation and the explicit representation of key quantities. By adhering to these restrictions, we were able to recast a host of otherwise intractable questions in non-linear optimization as problems in linear algebra. In this way we were able to make progress on a number of important questions in transfer learning, skills discovery, and hierarchical control. Of course, our reliance on simple linear models and explicit representations limits the applicability of both the new method, and the theoretical results. A number of promising avenues for future work exist at this frontier; namely, by relaxing

some of the formal constraints under which the present analysis was conducted.

Function Approximation: for very large domains, the explicit representations employed in this thesis are not practical. In the Atari domain for example, the sheer size of the state space renders it infeasible to represent a state of the underlying MDP as a one-hot vector. Instead, one would need to employ function approximation schemes in order to scale the procedures described herein. Importantly, as the underlying representations change so too must the operators. Returning to the Atari domain, deep RL practitioners typically replace the one-hot state representation with a distributed representation over pixel values. Supposing that one were to employ this sort of approximation scheme in the implementation of our new method, one would also need to replace each of the operators with an approximate analog. The explicit NNMF scheme, for example, might be replaced with an auto-encoder. Given the conceptual gravity of a recursive compositional action selection paradigm, an extension of the current method to massive state spaces would seem to be an invaluable contribution.

Incremental evolution: in both the online and offline setting, we have assumed that the hierarchical control architecture was constructed before hand, and remained unaltered during learning. While this is a sensible assumption for the exploration and analysis of the new method, it seems likely that the agent would benefit from more frequent integration of new experience into the hierarchy. In the limiting case, the agent would incrementally grow the hierarchical representation in real-time as it accrues more experience. This presents a natural trade off between the additional computational overhead of integrating the new experience into the hierarchy, and the immediacy of the improved planning capabilities of the agent with the new experience in hand. In this paradigm, the

agent would be able to immediately leverage incidence of critical experience to adjust its behaviour. One might imagine an agent engaged in a multi-room navigation task in which the higher layer control policies corresponds to each of the rooms that the agent has seen. As soon as the agent encounters a new room, the agent's autonomous skills discovery mechanism would run and the higher layer representation would shift to incorporate the new room as a novel subtask state. This has the effect of immediately altering the agent's behaviour as it is guided by a new higher layer picture of the world.

Non-stationary transition dynamics: the multitasking capabilities of our method have been considered in the restricted case in which the state and action space for the agent remained fixed, and it was exclusively the boundary reward structure that was altered between tasks. In the LMDP formulation, this restriction corresponded to the fact that the passive transition dynamics remained fixed across all of the tasks. From a mathematical perspective, a perturbation in the passive transition dynamics invalidates the linearity of the eignenvalue objective function, and policies no longer compose optimally. However, a well developed theory exists describing and bounding the effect of perturbations to eigenvalue problems[123]. It seems promising that the LMDP formulation may allow us to specify formal bounds on transfer between tasks which differ in their transition dynamics, by recasting the question as an eigenvalue perturbation problem.

Response dynamics: the optimal compositionality of the LMDP makes an explicit zero-shot prediction about the agent's ability to transfer skills between tasks when a complete basis is available. We extended this analysis to show that significant jump-start performance should be expected even when only a drastically depleted, low-rank approximation to the basis is available. Importantly, we

are able to make explicit predictions about the learning dynamics of an LMDP agent in response to a change in reward structure in the new method. These predictions might be tested in humans as evidence of a form of compositional action selection. Moreover, the deep hierarchical control architectures make predictions about response dynamics in the presence of depth. Verifying that these predictions are present in human action selection is an exciting prospect.

Concurrent exploration drivers: we have made extensive use of the concurrent execution of policies permitted in the LMDP framework. Another natural application of this capability is in the concurrent execution of multiple exploration drivers. Already we have argued that the concurrent execution of a base policy with a blended higher layer policy corresponds to a multiscale action selection paradigm in which the higher layer policy determines the long-term goal while the base policy specifies the short-term behaviour in pursuit of that goal. Similarly, one might introduce explicit sources of randomness at multiple spacial and temporal scales to aid in exploration. By way of example, recent work has employed simple count-based exploration over higher level abstractions[124], with promising results. Conceptually, the concurrent execution of multiple exploration drivers aligns well with human experience. We often make primitive decisions as a function of their service towards achieving short-term goals, which we pursue as a function of their service towards achieving mid-term goals, and so on.

A formal theory of transfer: the linearity of the LMDP framework permits us to reframe many difficult questions in terms of simple linear algebraic operations. We have exploited this fact in the development of a recursive procedure for discovering latent structure in the task space, and formulating subsequent learning problems over this abstracted representation. Following from this analysis,

we were able to derive a host of interesting results pertaining to generally important questions in transfer learning. However, we stopped some way short of a complete theory of transfer for LMDPs, and more generally still, a complete theory for approximate transfer in MDPs. In particular, proving formal bounds on the acquisition of useful subtasks, the optimal number of subtasks, and the performance improvement would go some way to set the field of transfer learning on more sure footing.

7.4 A Final Word

Human intelligence is ultimately realized as the subtle interplay of a cacophony of many different elements. While it is likely that much of its specific character is the vestige of millions of years of unstructured evolutionary pressure, it is equally likely that it displays features required of any generally intelligent agent. One such feature would seem to be the ability to recall previous experience, and to flexibly repurpose that experience to tackle new challenges - absent which the agent would be functionally amnestic. Another would seem to be the ability to derive complex new behaviours from a composition of many simple behaviours - absent which the agent would be crippled by the explosive dimensionality of the real world. That these capabilities are fundamental requirements of any form of scalable intelligence is so intuitive as to be held as self-evident. Nevertheless, current state of the art methods do not exhibit these fundamental properties - and for good reason. The inherently unstructured nature of general MDPs positions them as diametrically opposed to the simple compositionality we require. While current state of the art algorithms at times demonstrate impressive

results, this may be a simulacrum. In this work we have sought the outline of a general approach for developing control structures which places a premium on these core capabilities. Our hope is that the demonstration of the general capability of a scheme that bakes these conceptual insights into the heart of the method might encourage further work in this principled direction.

A

A.1 The LMDP Setup for Experimental Domains

A.1.1 Canonical Rooms Domain

The canonical rooms domain is considered in a number of experiments throughout the thesis (∼ Figs.(3.3, 3.4, 3.7, 4.2)). Recall that the LMDP is defined as a 3-tuple $\langle \mathcal{S}, P, \mathbf{r} \rangle$. We now define the setup of each of these constituents in detail. The rooms domain contains four rooms, each of which is realized as a 5×5 grid of states. The rooms themselves are laid out in a 2×2 grid and are joined top to bottom, and left to right, via a single doorway states such that the domain contains a total

of $N_i = 104$ interior states, as depicted in Fig.(3.2). For each interior state we construct a 'shadow' boundary state resulting in a total of $N = N_i + N_b = 208$ states.

The passive dynamics may be written in block form as $P = [[P_{\mathcal{II}}, P_{\mathcal{IB}}] ; [0, I]] \in \mathbb{N} \times \mathbb{N}$ (see Section.(2.3.1) for a detailed discussion), where the interior passive transition dynamics are determined as the collapsed Markov Chain under some reference policies. Throughout this thesis we have assumed the reference policy to be the uniformly random policy. Where $\hat{P} \in \mathbb{R}^{(N_i \times N_i) \times |\mathcal{A}|}$ is a tensor of state transition probabilities, such that $p_{ijk} = p(s_i|s_j, a_k)$ is the probabilities of transitioning to state s_i after having taken action a_k from state s_j, the passive interior dynamics are then defined as:

$$P_{\mathcal{II}} \quad \rightarrow \quad p_{ij} = \frac{1}{|\mathcal{A}|} \sum_k p_{ijk}. \tag{A.1}$$

In all cases we take $|\mathcal{A}| = 5$ corresponding to motion in the 4 cardinal direction, and a no-opt stay action. The boundary transition dynamics are simply defined as:

$$P_{\mathcal{IB}} \quad \rightarrow \quad \alpha I, \tag{A.2}$$

where $\alpha \in \mathbb{R}$ is taken to be $1/10$, and transitions into boundary states are only possibly from their corresponding interior state.

The reward structure is taken to be $\mathbf{r} = [\mathbf{r}_{\mathcal{I}}, \mathbf{r}_{\mathcal{B}}]$, where $\mathbf{r}_{\mathcal{I}} = -0.1$ and $\mathbf{r}_{\mathcal{B}} = 0$ at goal states, and $-\infty$ at non-goal states.

A.1.2 Ring Domain

The ring domain, considered in Fig.(6.4), is a variation of the so-called N-chain domain utilized in Osband et al.[121]. In this domain the agent operates in a 1-dimensional ring of 200 interior states, and may at each time step, transition to the left, to the right, or remain stationary. Once again, for each interior state, we construct a corresponding shadow boundary state resulting in a total of $N = N_i + N_b = 400$ states in the domain. The passive transition dynamics are again taken to be the collapsed Markov chain under the uniformly random policy. Explicitly then, the passive dynamics are:

$$P_{\mathcal{I}\mathcal{I}} \quad \rightarrow \quad p_{ij} = \begin{cases} 1/3, & \text{if } j \in (i-1, i, i+1) \\ 0, & \text{otherwise} \end{cases}, \tag{A.3}$$

and,

$$P_{\mathcal{I}\mathcal{B}} \quad \rightarrow \quad \alpha I, \tag{A.4}$$

where again $\alpha \in \mathbb{R}$ is taken to be $1/10$. Similarly, the reward structure for the ring domain is again taken to be $\mathbf{r} = [\mathbf{r}_{\mathcal{I}}, \mathbf{r}_{\mathcal{B}}]$, where $\mathbf{r}_{\mathcal{I}} = -0.1$ and $\mathbf{r}_{\mathcal{B}} = 0$ at goal states, and $-\infty$ at non-goal states.

A.2 Initializations, Learning Rates Schedules, and Moving Averages

Initializations: online learning is carried out through the online stochastic estimation of the desirability function $\mathbf{z}$ via Eqn.(2.14). The desirability function is, in all cases, initialized as $\mathbf{z} = [\mathbf{z}_{\mathcal{I}}, \mathbf{z}_{\mathcal{B}}] = \mathbf{1}$, with the learning rate α annealed in a standard fashion (described below). When the agent completes an episode, it is initialized for the new episode by choosing between the interior states uni-

formly at random.

Learning Rate: in all experiments that involve an online learning component, the learning rate is annealed as a function of the epochs in accordance with the time based schedule:

$$\alpha = \alpha \times \frac{1}{1 + decay * epoch}. \tag{A.5}$$

The decay value is taken to be 0.05 and was determined by course grid-search. While this value is almost certainly not strictly optimal for most of the tasks presented, it represents a suitable middle-of-the-road value for the set of experiments we consider. Furthermore, this choice of a fixed value for the decay constant liberates us from the meta-problem of additional task specific hyper parameter optimization.

Moving Averages: in a number of experiments the presented results have been smoothed by averaging results over a fixed sized moving window. In all cases, results are smoothed over a centralized window (typically of size $k = 10$). Explicitly, the averaged results in a sequence of elements $[x_0, \ldots, x_n] \rightarrow [\hat{x}_0, \ldots, \hat{x}_n]$ are computed as:

$$\hat{x}_i = \frac{1}{min(i + k/2, n) - max(i - k/2, 1)} \sum_{max(i-k/2,1)}^{min(i+k/2,n)} x_i. \tag{A.6}$$

Notice that the centralized window has the effect of computing average values over smaller sample sizes at the beginning and at the end of the sequence. This effect was noted in the discussion of Fig.(3.4) from Section.(3.3).

A.3 Exact Multitasking in the Offline Setting

The results of the exact multitasking experiment in the offline setting are presented in Fig.(3.3). The base LMDP is constructed as described in Appendix.A.1.1. In this experiment the agent is required to solve four separate navigation tasks. The tasks are presented incrementally with each new task arriving after the agent has experienced 500 epochs. Each epoch consists of an episode with a maximum of 100 steps. Note that the absorbing state formulation is such that the agent may complete an episode by making a single transition into an absorbing boundary state, after which no further transitions occur and the agent remains fixed for the remainder of the episode.

The task library available to the agent is uncovered via the NNMF procedure formalized in Eqn.(3.9). For the actual NNMF algorithm we utilize the off-the-shelf coordinate descent implementation from SciKit Learn[125]. In the interests of reproducibility the state is always initialize via the signature ($\mathit{init} = \mathit{'nndsvdar'}, \mathit{random\text{-}state} = 0, \mathit{max\text{-}iter} = 100$). Note that the cost function for the NNMF procedure is taken to be the generalized Kullback-Liebler divergence.

When the new task is presented, the agent receives the corresponding boundary reward structure $\mathbf{r}_{\mathcal{B}}$ and is reset. Resetting the agent involves two operations: first the initial estimate for the desirability function is set according to Eqn.(3.3), and second, the learning rate is reset to 1, after which it is again annealed according to the schedule outlined in Section.(A.2) above.

A.4 Linear Convergence

In Fig.(3.4) we demonstrated the linear relationship between initialization error and convergence rate for the LMDP. As noted in Section.(3.3), this result follows from the convexity of the objective function. In this experiment the agent again operates in the 4-rooms domain whose setup is described in Appendix.A.1.1.

In this experiment we collect data from 10, 000 independent runs. Each run constitutes a full course of learning in which the agent executes the flat z-learning algorithm over 50 epochs, with each epoch consisting of an episode with a maximum of 1000 steps. For each run we collect two scalar values: the initialization error $||\mathbf{z}^* - \mathbf{z}^\circ||$, and the average number of steps taken over the run. Note that should the agent enter a boundary state which is not a goal state, the step count is taken to be the maximum step count since the agent can only make the null transition back to its current state. The initial value for the desirability function is determined according to $\mathbf{z}^\circ = \lambda\mathbf{z}^* + (1 - \lambda)\mathbf{z}$ where λ is drawn at even internals in $[0, 1]$, $\mathbf{z}^*$ is the optimal desirability function computed via power iteration on the explicit eigenvalue problem, and $\mathbf{z}$ is a vector whose elements are chosen uniformly at random and normalized.

A.5 Multitasking in the Online Setting

The online implementation of our method was considered in an experiment whose results are depicted in Fig.(4.2). In this experiment the agent operates in the canonical rooms domain, with the base LMDP being constructed as described in Appendix.A.1.1. Once again, the agent is tasked with

solving four separate navigation tasks. The tasks are presented incrementally in the predefined order: top-left, bottom-right, bottom-left, top-right. Each new task is presented to the agent after 150 epochs of learning. The absorbing state formulation is such that the agent may enter an absorbing state early in its trajectory, and remain stationary for the remainder of the episode.

In this experiment we compare the agent's performance when equipped with differently sized task libraries. In each case the agent's task library is uncovered via the NNMF procedure described in Eqn.(3.9), for different values of the decomposition factor. In Fig.(4.2) we consider results for decomposition factors $k = \{4, 12, 52\}$. The online implementation of our methods requires the specification of the subtask transition structure. In line with Eqn.(5.3), we take $P_{\mathcal{IT}} = \alpha D$, where D is the normalized low-rank approximation to the full task basis Z, and $\alpha = 1$. Notice that $P_{\mathcal{IT}}$ remains constant during the course of learning and is not dynamically adjusted.

At any point in time, the agent operates under the effective desirability function $\mathbf{z}^{eff} = \gamma \mathbf{z}^{base} + (1 - \gamma)\mathbf{z}^{hl}$, where $\mathbf{z}^{base}$ is stochastically estimated via the z-iteration procedure, and $\mathbf{z}^{hl}$ is obtained via Eqn.(4.7). The interpolation parameter is taken to be equal to the learning rate, $\gamma = \alpha$, which is annealed as described in Section.(A.2). This has the effect of transitioning the agent from an effective policy comprised solely of the higher layer influence $\mathbf{z}^{hl}$ to one comprised solely of the agent's base layer policy $\mathbf{z}^{base}$.

When a new task is presented, the agent state is reset. This includes resetting the agent's estimate of the boundary reward structure $\hat{\mathbf{r}}_{\mathcal{B}}$, and resetting the learning rate back to 1. The estimate for the boundary reward structure is optimistically initialized as $\hat{\mathbf{r}}_{\mathcal{B}} = 1$.

A.6 Multitasking with Deep Control Hierarchies

The results of our experiment with deep control hierarchies is depicted in Fig.(6.4). Here, the agent operates in a 200 state ring domain as described in Section.(A.1.2). The agent is tasked with solving four separate navigation tasks corresponding to reaching four singleton goal states evenly distributed around the ring. The tasks are identified by their relative position in the ring Q_1, Q_2, Q_3, Q_4 corresponding to goal states at position 50, 100, 150, 200 respectively. The tasks are again presented incrementally in a predefined order Q_1, Q_3, Q_2, Q_4. Each new task is presented after the agent has experienced 150 epochs of learning.

In this experiment we compare the performance of various MLMDP agents equipped with control hierarchies of different depths. We compare four such agents. The first agent is not equipped with any additional control structure and corresponds to the flat implementation. The second agent is equipped with a single layer control structure with $N_t = 66$ higher layer states. The third agent is equipped with a two layer control structure with $N_t = 66$ states at the first layer and $N_t = 22$ states at the second layer, and the forth is equipped with a three layer hierarchy with $N_t = \{66, 22, 7\}$ for each layer of the hierarchy respectively. In each case, the hierarchy is constructed via the recursive procedure depicted in Fig.(6.1), with $P_t^l = \alpha D^l$, where D^l is the normalized low-rank approximation to the full task basis at that layer Z^l, and $\alpha = 1$.

References

[1] Ivan Petrovich Pavlov. *The work of the digestive glands: lectures*. C. Griffin, 1902.

[2] J.B. Watson. Psychology as the behaviorist views it. *Psychological Review*, 20:158–177, 1913.

[3] B. F. Skinner. Two types of conditioned reflex and a pseudo type. *The Journal of General Psychology*, 12(1):66–77, 1935.

[4] A. Bandura and R.H Walters. *Social Learning Theory*. Prentice-hall Englewood Cliffs, NJ, 1977.

[5] Robert Epstein. Skinner, creativity, and the problem of spontaneous behavior. *Psychological Science*, 2(6):362–370, 1991.

[6] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[7] Richard S Sutton, Andrew G Barto, and Ronald J Williams. Reinforcement Learning is Direct Adaptive Optimal Control. *IEEE Control Systems Magazine*, 12(2):19–22, 1992.

[8] Yann LeCun. Predictive learning: lecture, July 2016.

[9] Marco Wiering and Intelligent Systems Group. Reinforcement Learning for Robot Control. *Mobile Robots*, 4573(314):92–104, 2002.

[10] Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.

[11] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. *Advances in Neural Information Processing Systems (NIPS)*, 19:1, 2007.

[12] Jens Kober, J Andrew Bagnell, and Jan Peters. Designacion Supervisor.pdf. *The International Journal of Robotics Research*, 32(11):1238–1274., 2009.

[13] Zhiyong Tan, Chai Quek, and Philip Y.K. Cheng. Stock trading with cycles: A financial application of anfis and reinforcement learning. *Expert Systems with Applications*, 38(5):4741 – 4755, 2011.

[14] Pierpaolo G Necchi. Reinforcement Learning For Automated Trading. Technical report, Politecnico di Milano, 2016.

[15] Gerald Tesauro. Temporal Difference Learning and TD-Gammon. *Communications of the ACM*, 38(3):58–69, 1995.

[16] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *ArXiv*, pages 1–19, 2017.

[17] D Silver, J Schrittwieser, K Simonyan, I Antonoglou Nature, and Undefined 2017. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2016.

[18] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. StarCraft II. *ArXiv*, 2017.

[19] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M. Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Yuhuai Wu, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II, 2019.

[20] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[22] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building Machines That Learn and Think Like People. *Behavioral and Brain Sciences*, 40:1–101, 2016.

[23] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[24] Tianhe Yu, Pieter Abbeel, Sergey Levine, Chelsea Finn, and L G Oct. One-Shot Hierarchical Imitation Learning of Compound Visuomotor Tasks. *ArXiv*, 2018.

[25] Yan Duan, Marcin Andrychowicz, Bradly Stadie, and Jonathan Ho. One-Shot Imitation Learning. *Advances in Neural Information Processing Systems (NIPS)*, pages 1087–1098, 2017.

[26] Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Thomas L Griffiths, and Alexei A Efros. Investigating Human Priors for Playing Video Games. *ArXiv*, 2018.

[27] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[28] Robert M French. Catastrophic Forgetting in Connectionist Networks : Causes , Consequences and Solutions. *Trends in Cognitive Sciences*, 3:128–135, 1999.

[29] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, and Andrei A Rusu. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521–3526, 2017.

[30] R.S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, aug 1999.

[31] A. Bonarini, A. Lazaric, and M. Restelli. Incremental Skill Acquisition for Self-motivated Learning Animats. In S. Nolfi, G. Baldassare, R. Calabretta, J. Hallam, D. Marocco, O. Miglino, J.-A. Meyer, and D. Parisi, editors, *Proceedings of the International Conference on Simulation of Adaptive Behavior (SAB)*, volume 4095, pages 357–368, Heidelberg, 2006. Springer Berlin.

[32] Christopher M Vigorito and Andrew G Barto. Intrinsically Motivated Hierarchical Skill Learning in Structured Environments. *IEEE Transactions on Autonomous Mental Development*, 2(2):132–143, jun 2010.

[33] G. Konidaris, S. Kuindersma, R. Grupen, and A.G. Barto. Autonomous Skill Acquisition on a Mobile Manipulator. *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1468–1473, 2011.

[34] P.L. Bacon and D. Precup. The option-critic architecture. In *Advances in Neural Information Processing Systems (NIPS), Deep Reinforcement Learning Workshop*, 2015.

[35] Hilary Putnam. Brains and behavior. *Mind, Language and Reality*, pages 325–341, 2010.

[36] Michael Rescorla. The computational theory of mind. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2017 edition, 2017.

[37] Ferdinando A Mussa-Ivaldi. Modular features of motor control and learning. *Current Opinion in Neurobiology*, 9:713–717, 1999.

[38] J Randall Flanagan, Eri Nakano, Hiroshi Imamizu, Rieko Osu, Toshinori Yoshioka, and Mitsuo Kawato. Composition and Decomposition of Internal Models in Motor Learning under Altered Kinematic and Dynamic Environments. *The Journal of Neuroscience*, 19(20), 1999.

[39] Neville Hogan and Dagmar Sternad. Dynamic primitives of motor behavior. *Biological Cybernetics*, 106:727–739, 2012.

[40] Simon F Giszter. Motor primitives - new data and future questions. *Current Opinion in Neurobiology*, 33:156 – 165, 2015. Motor circuits and action.

[41] Thomas Serre, Gabriel Kreiman, Minjoon Kouh, Charles Cadieu, Ulf Knoblich, and Tomaso Poggio. A quantitative theory of immediate visual recognition. *Progress in Brain Research*, 165:33–56, 2007.

[42] Maximilian Riesenhuber and Tomaso Poggio. Models of object recognition. *Nature Neuroscience*, 3(11s):1199–1204, 2000.

[43] J Fuster. Upper processing stages of the perception-action cycle. *Trends in Cognitive Sciences*, 8(4):143–145, 2004.

[44] E Koechlin, C Ody, and F Kouneiher. The architecture of cognitive control in the human prefrontal cortex. *Science*, 302(5648):1181–1185, 2003.

[45] E. Todorov. Linearly-solvable Markov decision problems. *Advances in Neural Information Processing Systems (NIPS)*, pages 1369–1376, 2006.

[46] Andrew M Saxe, Adam C Earle, and Benjamin Rosman. Hierarchy Through Composition with Multitask LMDPs. *Proceedings of the International Conference on Machine Learning (ICML)*, 70:3017–3026, 2017.

[47] Adam C Earle, Andrew M Saxe, and Benjamin Rosman. Hierarchical Subtask Discovery with Non-Negative Matrix Factorization. *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–11, 2018.

[48] Dimitri Panteli Bertsekas. *Dynamic Programming and Optimal Control*. Athena scientific, 2001.

[49] Ronald J. Willia. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256, 1992.

[50] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems (NIPS)*, pages 1–7, 2000.

[51] M.E. Taylor and P. Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

[52] Pierre-Arnaud Coquelin and Rémi Munos. Bandit Algorithms for Tree Search. *ArXiv*, 2007.

[53] Andrew G Barto and Sridhar Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13:343–379, 2003.

[54] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1043–1049, 1998.

[55] Thomas G Dietterich. The MAXQ Method for Hierarchical Reinforcement Learning. *Proceeding of the International Conference on Machine Learning (ICML)*, pages 118–126, 1998.

[56] H.J. Kappen. Linear Theory for Control of Nonlinear Stochastic Systems. *Physical Review Letters*, 95(20):200201, nov 2005.

[57] John Schulman, Pieter Abbeel, and Xi Chen. Equivalence Between Policy Gradients and Soft Q-Learning. *ArXiv*, pages 1–15, 2017.

[58] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement Learning with Deep Energy-Based Policies. *Proceedings of the International Conference on Machine Learning*, 2017.

[59] Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized Markov decision processes. *ArXiv*, 2017.

[60] A.M. Saxe. Multitask Model-free Reinforcement Learning. In *CogSci*, 2014.

[61] Benjamin van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Will it Blend? Composing Value Functions in Reinforcement Learning. *ArXiv*, 2018.

[62] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable Deep Reinforcement Learning for Robotic Manipulation. *ArXiv*, 2018.

[63] E. Todorov. Compositionality of optimal control laws. *Advances in Neural Information Processing Systems (NIPS)*, pages 1856–1864, 2009.

[64] Anders Jonsson and Vicenç Gómez. Hierarchical Linearly-Solvable Markov Decision Problems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2016.

[65] Lisa Torrey and Jude Shavlik. *Transfer Learning*. IGI Global, 2009.

[66] Alessandro Lazaric. Transfer in Reinforcement Learning : a Framework and a Survey. *Reinforcement Learning*, pages 143–173, 2012.

[67] Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. *Proceedings of the International Conference on Machine learning (ICML)*, pages 240–247, 2008.

[68] Marco da Silva, Fredo Durand, and Jovan Popovic. Linear Bellman combination for control of character animation. *ACM Transactions on Graphics*, 28(3):82, 2009.

[69] D. Borsa, T. Graepel, and J. Shawe-Taylor. Learning Shared Representations in Multi-task Reinforcement Learning. *ArXiv*, 2016.

[70] T.G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[71] Peter Dayan and Geoffrey Hinton. Feudal Reinforcement Learning. *Advances in Neural Information Processing Systems (NIPS)*, pages 271–278, 1993.

[72] George D Konidaris and Andrew Barto. Autonomous Shaping : Knowledge Transfer in Reinforcement Learning Autonomous Shaping : Knowledge Transfer in Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 489–496, 2006.

[73] Gheorghe Comanici and Doina Precup. Optimal Policy Switching Algorithms for Reinforcement Learning. In *Proceedings of the International Confernece on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 709–714, 2010.

[74] A. Barreto, R. Munos, T. Schaul, and D. Silver. Successor Features for Transfer in Reinforcement Learning. *ArXiv*, 2016.

[75] Benjamin Rosman and Subramanian Ramamoorthy. What good are actions? Accelerating learning using learned action priors. *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, 2012.

[76] K Ferguson and S Mahadevan. Proto-transfer Learning in Markov Decision Processes Using Spectral Methods. In *Proceedings of the International Conference on Machine Learning (ICML), Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[77] Jian Zhang, Zoubin Ghahramani, and Yiming Yang. Flexible latent variable models for multi-task learning. *Machine Learning*, 73(3):221–242, 2008.

[78] P. Ruvolo and E. Eaton. ELLA: An efficient lifelong learning algorithm. *Proceedings of the International Conference on Machine Learning (ICML)*, 28(1):507–515, 2013.

[79] Yuri Nesterov. *Introductory lectures on convex optimization*. Springer, 2004. ISBN 9781461346913.

[80] James L. Carroll and Kevin Seppi. Task similarity measures for transfer in reinforcement learning task libraries. *Proceedings of the International Joint Conference on Neural Networks*, 2:803–808, 2005.

[81] D.D Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91, 1999.

[82] Chris Ding, Xiaofeng He, and Horst D Simon. On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 606–610, 2005.

[83] A Basu, B T Road, I A N R Harris, and M C Jones. Robust and Efficient Estimation by Minimising a Density Power Divergence. *Biometrika*, pages 1–26, 1998.

[84] S Eguchi and Y Kano. Robustifing Maximum Likelihood Estimation by Psi-divergence. *ISM Research Memorandam*, 2001.

[85] A. Cichocki, S. Cruces, and S. Amari. Generalized alpha-beta divergences and their application to robust nonnegative matrix factorization. *Entropy*, 13(1):134–170, 2011.

[86] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[87] C. Ding, T. Li, and W. Peng. On the equivalence between Non-negative Matrix Factorization and Probabilistic Latent Semantic Indexing. *Computational Statistics and Data Analysis*, 52 (8):3913–3927, 2008.

[88] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems (NIPS)*, pages 1141–1148, 2004.

[89] M.M. Botvinick, Y. Niv, and A.C. Barto. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, 113(3):262–80, dec 2009.

[90] Kimberly L. Stachenfeld, M M Botvinick, and Samuel J. Gershman. Design Principles of the Hippocampal Cognitive Map. *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9, 2014.

[91] Matan Gavish and David L. Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014.

[92] Satinder Pal Singh. Transfer of Learning by Composing Solutions of Elemental Sequential Tasks. *Machine Learning*, 8(3):323–339, 1992.

[93] Thomas Ringstrom and Paul Schrater. Compositional Constraint Satisfaction Control with Hierarchical Goals and Dynamics. In *Proceedings of the Conference on Cognitive Computational Neuroscience*, pages 1–2, 2017.

[94] Thomas J. Ringstrom and Paul R. Schrater. Constraint Satisfaction Propagation: Non-stationary Policy Synthesis for Temporal Logic Planning. *ArXiv*, 2019.

[95] Doina Precup, Richard S. Sutton, and Satinder Singh. Theoretical results on reinforcement learning with temporally abstract options. In *Proceedings of the 10th European Conference on Machine Learning*, pages 382–393, 1998.

[96] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. A Laplacian Framework for Option Discovery in Reinforcement Learning. *Proceedings of the International Conference on Machine Learning (ICML)*, 70:2295–2304, 2017.

[97] Ronan Fruit and Alessandro Lazaric. Exploration–Exploitation in MDPs with Options. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[98] M. Stolle and D. Precup. Learning options in reinforcement learning. *Abstraction, Reformulation, and Approximation*, 2371:212–223, 2002.

[99] Sander G. van Dijk and Daniel Polani. Grounding subgoals in information transitions. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 105–111. IEEE, apr 2011.

[100] A. Solway, C. Diuk, N. Córdova, D. Yee, A.G. Barto, Y. Niv, and M.M. Botvinick. Optimal Behavioral Hierarchy. *PLoS Computational Biology*, 10(8), aug 2014.

[101] C. Diuk, A. Schapiro, N. Córdova, J. Ribas-Fernandes, Y. Niv, and M. Botvinick. Divide and conquer: Hierarchical reinforcement learning and task decomposition in humans. *Computational and Robotic Models of the Hierarchical Organization of Behavior*, pages 271–291, 2013.

[102] S. Mannor, I. Menache, A. Hoze, and U. Klein. Dynamic abstraction in reinforcement learning via clustering. *Proceedings of the International Conference on Machine learning (ICML)*, page 71, 2004.

[103] O. Simsek. *Behavioral building blocks for autonomous agents: Description, identification, and learning*. Phd thesis, University of Massachussetts, Amherst, 2008.

[104] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal Networks for Hierarchical Reinforcement Learning. *ArXiv*, 2017.

[105] H Wielandt. Das iterationsverfahren bei nicht selbstadjungierten linearen eigenwertaufgaben. *Mathematische Zeitschrift*, 50:93–143, 1944.

[106] Ilse C. F. Ipsen. A History of Inverse Iteration. *Mathematical Works, Linear Algebra and Analysis*, II:464–72, 1996.

[107] Peter J Olver and 1951 Shakiban, Chehrzad. *Applied linear algebra*. Upper Saddle River, NJ : Prentice Hall, 2006.

[108] Yuu Jinnai, David Abel, Michael Littman, and George Konidaris. Finding Options that Minimize Planning Time. *ArXiv*, 2018.

[109] Parham Moradi, Mohammad Ebrahim Shiri, and Negin Entezari. Automatic skill acquisition in reinforcement learning agents using connection bridge centrality. *Communications in Computer and Information Science (CCIS)*, 120:51–62, 2010.

[110] Pierre-Luc Bacon. *On the Bottleneck Concept for Options Discovery*. PhD thesis, McGill University, 2013.

[111] Ö. Şimşek and A.S. Barto. Skill Characterization Based on Betweenness. *Advances in Neural Information Processing Systems (NIPS)*, pages 1497–1504, 2009.

[112] B. Bakker and J.H. Schmidhuber. Hierarchical reinforcement learning with subpolicies specializing for learned subgoals. In *Neural Networks and Computational Intelligence*, pages 125–130, 2004.

[113] Amy McGovern and Andrew G Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. *Proceedings of International Conference on Machine Learning (ICML)*, pages 361 – 368, 2001.

[114] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral : Robust Multitask Reinforcement

Learning. *Advances in Neural Information Processing Systems (NIPS)*, pages 4496–4506, 2017.

[115] Sanjay Krishnan, Roy Fox, Ion Stocia, and Ken Goldberg. DDCO : Discovery of Deep Continuous Options for Robot Learning from Demonstrations. *1st Conference on Robot Learning (CoRL)*, pages 1–21, 2017.

[116] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning An Embedding Space For Transferable Robot Skills. *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–16, 2018.

[117] Kalanit Grill-Spector and Kevin S Weiner. The functional architecture of the ventral temporal cortex and its role in categorization. *Nature Reviews Neuroscience*, 15(8):536, 2014.

[118] Daniel J. Felleman and David C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1):1–47, 1991.

[119] Q. Wang, O. Sporns, and A. Burkhalter. Network Analysis of Corticocortical Connections Reveals Ventral and Dorsal Processing Streams in Mouse Visual Cortex. *Journal of Neuroscience*, 32(13):4386–4399, 2012.

[120] Renbo Zhao, Vincent Y. F. Tan, and Huan Xu. Online Nonnegative Matrix Factorization with General Divergences. *ArXiv*, pages 1–25, 2016.

[121] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Quality on Demand via Bootstrapped DQN. *Advances in Neural Information Processing Systems (NIPS)*, pages 4026–4034, 2016.

[122] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards Generalization and Simplicity in Continuous Control. *Advances in Neural Information Processing Systems (NIPS)*, pages 6550–6561, 2017.

[123] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[124] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-Based Exploration with the Successor Representation. *ArXiv*, 2018.

[125] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher,

M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.