Inter-Task Learning with Spatio-Temporal Abstractions

A Project Report

submitted by

ARUN TEJASVI CHAGANTY

in partial fulfilment of the requirements
for the award of the degrees of

MASTER OF TECHNOLOGY
&
BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

April 2012

THESIS CERTIFICATE

This is to certify that the thesis titled **Inter-Task Learning with Spatio-Temporal Abstractions** submitted by **Arun Tejasvi Chaganty**, to the Indian Institute of Technology, Madras, for the award of the degrees of **Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Balaraman Ravindran
Research Guide
Assistant Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

My search for understanding abstractions' role in learning began atleast a year ago, and it has been a long journey. Without a doubt, it would not have been possible without the support and guidance of Dr. Ravindran Balaraman, a professor who I feel proud to have as a mentor and friend. At the same time, many of the ideas I've had would not have been conceivable without the excellent introductions to concepts like Lie groups, conservation principles, etc. from Dr. Suresh Govindarajan. His ceaseless enthusiasm for science have enabled me to appreciate more the poetry of reality. I doubt I would have been able to keep up were it not for to-be-Dr. Naveen Sharma. Our lucid discussions on dimensionality, topolgy, homology, algebra, black holes and what not else have enriched my understanding of the aforementioned stanzas.

Five years is a long time. I am ever grateful to my friends, Kirtika Ruchandani, Tanmai "Coco" Gopal, Sriram "Tehelka", Smruthi Balakannan, Prateek "Kama" Gaur, Suryakiran "Virus" Oruganti, Siva, Subhashini and many others for keeping me sane through the years. My seniors, Pranesh Srinivasan, NG Srinivas, Vimal JK, SK Anirudh, Vikram SV, have played a dual role of friends and mentors in all matters, and I must thank them doubly for that.

Last but not least, I must thank the Computer Science Department as whole. I am immensely happy to say that I have taken something from almost every course I sat in. For this, the credit must go to remarkable teachers like Dr. Shankar Balachandran, Dr. Ravindran Balaraman, Dr. R Kalyanakrishnan, Dr. Gaurav Raina, Dr. C Chandrashekar and Dr. Jayalal Sharma.

ABSTRACT

In order for artificially intelligent agents to reliably interact and solve problems in the world, they must be able to learn from decisions they have made in the past. Reinforcement learning provides an elegant framework to model agent learning through interaction with an environment. Generalising this experience to different tasks within the same environment as well as across environments remains a key challenge for the community. Despite a generally sound understanding of how previous knowledge could be incorporated in the agent, viz. through options and homomorphisms, there are few well-founded approaches to actually selecting experiences at an appropriate granularity, or to adapting them to the current context.

This project proposes two approaches to the problem based on spatial and temporal abstractions respectively. Spatial abstractions, such as MDP homomorphisms, allow the agent to optimally transfer decision policies from one environment to another. We extend the current homomorphism framework to the tricky domain of continuous state and action spaces. Building on this theoretical foundation, we describe an algorithm to search for homomorphisms through gradient descent in the space of affine continuous homomorphisms. We show that the homomorphisms the method finds are intuitive, and can significantly reduce learning times in new domains.

Temporal abstractions, on the other hand, allow an agent to decompose a task into subtasks. By extending a well known result on small world networks in social networks, we show how to generate a set of subtasks of subtasks that efficiently span the space of all tasks. In experiments on standard domains, our agent outperforms competing subtask-selection algorithms. We also show that the subtasks we choose can be learnt more efficiently than existing methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Reinforcement learning (RL) is a widely studied learning framework for autonomous agents, particularly because of it's extreme generality; it addresses the problem of learning optimal agent behaviour in an unknown stochastic environment. In this setting, an agent explores a state space, receiving rewards for actions it takes; the objective of the agent is to maximise it's rewards accumulated over time. In the conventional setting, the agent approaches each new problem from scratch. We would like to study how the agent can exploit solutions it has already learnt in order to solve a new task.

1.1 Exploiting Structure

To reuse behavioural policies, we need to either identify some 'structure' in the environment or to impose such 'structure' ourselves to transfer the policy. This structure is roughly identified through spatial and temporal abstractions. One approach for the former is to form a hierarchical representation of the environment, creating a factored MDP (Guestrin *et al.*, 2003). Yet another, one that we adopt in this work, is to identify correspondences, or homomorphisms, between states either in the same environment (i.e. a "symmetry"), or between environments. There exist several criteria for these correspondences; Li *et al.* (2006*a*) present a survey of various approaches taken in the community, and relate them to each other.

With temporal abstractions, high-level actions are introduced which capture sequences of primitive actions. In this light, temporal abstractions capture the notion of a "subtask". The most common approach for temporal abstractions is the options framework proposed by Sutton *et al.* (1999); we adopt this framework as well. Ravindran and Barto (2003) show how temporal abstractions can be combined with spatial abstractions using relativised options. Both spatial and temporal abstractions play an important role in transfer learning, where we wish to extend optimal behaviour learnt in one task to another task. These approaches fall under the broader field of transfer learning; a survey of many other transfer learning techniques can be found in Taylor and Stone (2009).

1.1.1 A Motivating Example: Controlling an Octopus Arm

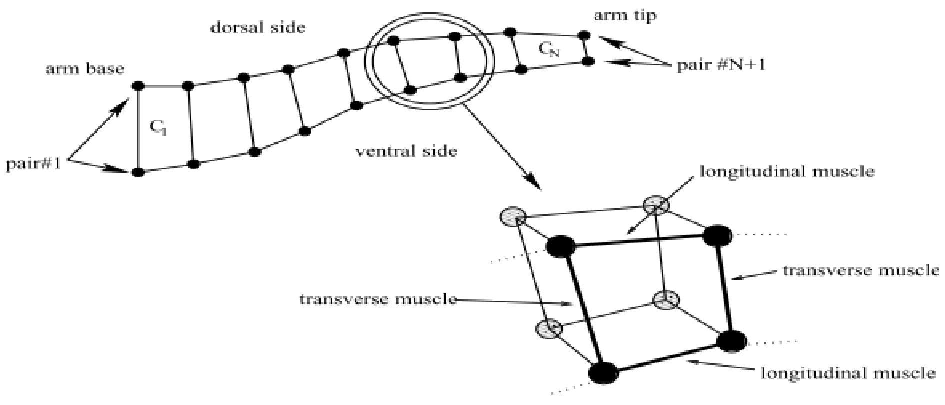


Figure 1.1: Dynamics of an Octopus Arm

As a motivating example for abstraction, consider the control of an octopus arm, as shown in Figure 1.1. This arm can be modelled as a composition of 10 compartments, each of which can be controlled by contracting longitudinal and transverse muscles (Engel *et al.*, 2006). In total, there are 88 continuous state variables associated with the arm, and it is controlled with a 32 dimensional, continuous valued signal. A typical task in such a domain is to move the arm to grab at a particular location, somewhere in space. The space may include obstacles that should not be touched.

In this domain, spatial abstractions would study correspondences between arm configurations. While exactly specifying equivalent states is difficult, one could visualise several redundancies in the state space description. For example, consider the final segment in the arm; imagine an axis drawn from the previous component of the arm – positions on one side of the arm are equivalent to positions on the other side. One could imagine two segments forming a bend to also be equivalent. The complexity of analytically specifying these symmetries is all the more a reason to have the agent to find these approximate symmetries itself, rather than have them specified upfront.

On the other hand, remembering the set of actions to restore the arm to a particular configuration would correspond to a temporal abstraction; we have abstracted the sequence of steps required to achieve the configuration with a single one.

This example was introduced by Engel *et al.* (2006); its scale is somewhat unprecedented in reinforcement learning, but it has sparked some interest, appearing later as a challenge for the RL competition in 2009. Even the best entry in the competition had a very poor absolute score on the domain. If this is the challenge posed by the control of a single arm, it begs the question; how can we scale to control the remaining seven arms?

The main question we seek to address in this thesis is how an agent can use spatial and temporal abstractions to learn how a task more quickly. Moreover, we focus on techniques that do not require a complete and exact specification of the target domain.

1.2 Learning Spatial Abstractions

MDP homomorphisms (Ravindran, 2004) provide an important theoretical framework for transfer by describing when a policy can be transferred from one MDP to another. While Soni and Singh (2006) describe the use of homomorphisms to transfer option policies in continuous domains, they only consider variable remappings, and do not study the general properties of continuous MDP homomorphisms. In Chapter **??**, we define continuous MDP homomorphisms, and show that lifted policies share the same properties as in the discrete case.

We are primarily interested in finding homomorphisms, and in general, spatial abstractions, in continuous spaces. Most techniques present in the transfer learning community require hand-coded task mappings, or search a very limited set of transformations. Both Soni and Singh (2006) and Taylor *et al.* (2008) search across various one-to-one variable mappings from the source to target domains. Taylor *et al.* (2007) learns inter-task mappings by training a classifier to predict the action taken given (s, s', r). When the source and target state spaces are not equivalent, they train a classifier for each variable subset, and combine the outputs of these classifiers.

1.2.1 Our Contribution: Homomorphic Filters

In contrast, the technique we describe, homomorphic filtering, is applicable to any differentiable set of homomorphisms. It operates by performing a stochastic gradient descent in this set of homomorphisms. In particular, we study the set of continuous affine homomorphisms, namely homomorphisms involving rotation and translation of the state and action space. Variable remapping is a trivial subset of the affine family. We evaluate our algorithm on the Cart Pole domain, considering performance of the lifted policy on distorted versions of the state space. We also study how these homomorphisms can bootstrap learning in the distorted state space.

1.3 Learning Temporal Abstractions

As mentioned earlier, we use the options framework to describe temporal abstractions. An "option" is a special action that the agent can choose to take; once the agent has selected an option, it follows a pre-specified behavioural policy till some termination conditions for the option have been satisfied. The framework decribes how learning algorithms should be suitably modified in order to incorporate options.

Unfortunately, the framework does not describe how the options themselves should be constructed, and neither is there a consensus in the community on the same. The prevalent view is that subtasks should represent skills, i.e. partially defined action policies that constitute a part of many reinforcement learning problems (Thrun and Schwartz, 1995). For this reason, much of the existing work centres around identifying 'bottlenecks', regions that the agent tends to visit frequently (McGovern and Barto, 2001), either empirically as in (McGovern and Barto, 2001), or, more recently, using graph theoretic methods like betweenness centrality (Şimşek and Barto, 2008) or graph partitions (Menache *et al.*, 2002). The intuition is that options that navigate an agent to such states helps the agent move between strongly connected components, thus leading to efficient exploration.

These option generation schemes suffer from two serious drawbacks; (i) they either require complete knowledge of the MDP or follow a sample-heavy approach of constructing a local model from trajectories, and (ii) there are, in general, several options to bottlenecks that can be initiated by the agent. This leads leading to a blowup in the decision space, often causing the agent to take more time to learn the task as it filters through the unnecessary options.

If one considered these options as additional edges to the bottleneck states, in the sense that a single decision is sufficient to transit the agent from a state, to the bottleneck, the resultant state-interaction graph would now be "more" connected. To highlight the importance of the connectivity of the state-interaction graph, consider the Markov chain induced by a policy for an Markov decision process. It is well known that the convergence rate of a Markov chain (mixing time), is directly related to its conductance (Jerrum and Sinclair, 1988), and thus its algebraic connectivity.

1.3.1 Our Contribution: Small World Options

Recognising the importance of connectivity, we apply concepts from Kleinberg's work on small world networks, to the context of problem solving with autonomous agents. These graphs have been shown to have exceptionally high algebraic connectivity, and thus fast Markov chain mixing times (Tahbaz-Salehi and Jadbabaie, 2007). In a small-world network, each node has one non-neighbouring edge, which connected to another node with a probability inversely proportional to the distance between them. With this simple construction, Kleinberg (2000) showed that an agent can discover a short path to any destination using only local information like the coordinates of it's immediate neighbours. In contrast, other graph models with a small diameter only state the existence of a short path, but do not guarantee that an agent would be able to find such a path.

In our context, we construct subtasks distributed according to the small world distribution as follows; create an option that will take the agent from a state s to another state s' with a probability inversely proportional to the distance between s and s'. We prove that this set of subtasks enables the agent to easily solve any task by using only a logarithmic number of options to reach a state of maximal value (Section 5.2). As this scheme adds at most one additional option per state, we do not explode the decision space for the agent.

Furthermore, in Section 5.3, we devise an algorithm that learns small world options from the optimal policies learnt over a few tasks in the domain. Thus not only are small world options effective to use, they are also simple to learn, and do not require any global analysis of the MDP. Experiments on several standard domains show that small-world options outperform bottleneck-based methods, and that small world options require significantly fewer learning epochs to be effective.

1.4 Organisation of Thesis

We present an overview to topics in reinforcement learning, homomorphisms, and small world network in Chapter 2. The field of related work is reviewed in Chapter 3. In Chapter 4, we describe the peculiarites of symmetries in continuous domains and extend the definition of MDP homomorphisms to capture the unique behaviour of continuous symmetries. Building on this theoretical foundation, we describe our online algorithm to find continuous homomorphisms, as well as the results we obtained. At this point, we shift focus to temporal abstractions, and describe small world options in Chapter 5. Finally, we conclude, discussing how spatio-temporal abstractions could provide a direction to solving complex problems such as the octopus domain, as well as future directions in Chapter 6. We also describe some alternate approaches we attempted to find continuous homomorphisms in Appendix A.

CHAPTER 2

Background

2.1 Reinforcement Learning

In reinforcement learning, the standard representation of an environment and task instance is a Markov decision process (MDP). An MDP can be represented as the tuple, $\langle S, A, P, R, \gamma \rangle$, where S and A are the state and action domains which are known to the agent. $P : S \times A \rightarrow \mathcal{X}(S)$, where $\mathcal{X}(S)$ is the set of all probability measures over S, describes the dynamics of the world through state-action transition probabilities. $R : S \times A \rightarrow \mathbb{R}$ describes the task at hand by ascribing rewards for state transitions. Both P and R are initially unknown to the agent. Finally, $\gamma \in [0, 1)$ is a parameter, called the 'discount factor', that weighs the value of future rewards.

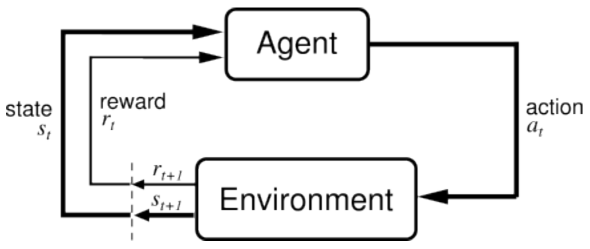


Figure 2.1: Agent-Environment Interface

In this setting, an agent in a state $s \in S$ chooses an action $a \in A$, and moves to a state s' with probability $P(s'|s, a)$, receiving a reward $R(s, a)$ (Figure 2.1). In the fully observable setting, the agent is aware of which state it is in, and the objective of the agent is to find a policy $\pi : S \rightarrow \mathcal{X}(A)$, i.e. a decision procedure for selecting actions, that maximises the reward it accumulates in the long run, $R = \sum_i \gamma^i r_i$. R is also called the return.

In the remainder of this section, we will describe the basic approach for finding an optimal policy π for a discrete MDP, and its continuous variant.

2.1.1 Discrete Markov Decision Processes

In the discrete case, S and A are predictably finite sets of states and actions. We define the value function $V^{\pi} : S \to \mathbb{R} = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s\right]$ to be the expected return from s, and $Q^{\pi} : S \times A \to \mathbb{R} = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s, A_0 = a\right]$ to be the expected return from s, after taking the action a. These can be written in a recursive form,

$$\begin{aligned} V^{\pi}(s) &= \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi}(s') \\ Q^{\pi}(s,a) &= R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) Q^{\pi}(s',a'). \end{aligned}$$

Our objective can then be stated as finding a policy π with an optimal value function, i.e. $V^*(s) = \sup_{\pi} V^{\pi}(s)$ at all $s \in S$. The optimal value functions must satisfy the Bellman optimality conditions,

$$\begin{aligned} V^*(s) &= \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^*(s') \\ Q^*(s,a) &= R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a'} Q^*(s',a'). \end{aligned}$$

Given an optimal Q, it is possible to construct a greedy policy that is optimal; $\pi(s,a^*) = 1$ when $a^* = \operatorname{argmax}_a Q(s,a)$, and 0 otherwise. In principle, if the agent knew the MDP, it could construct the optimal value function, and from it an optimal policy. However, in the typical setting, the agent is only aware of the state-action space, S and A, and must learn Q through exploration. The Q-learning algorithm learns Q with a simple update for every step the agent takes,

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right],$$

where $\alpha \in [0,1]$ is a parameter that controls the learning rate. It has been shown that the Q-learning algorithm converges to the optimal value function in the limit with fairly permissive assumptions.

2.1.2 Continuous Markov Decision Processes

In continuous domains, defining S and A must be done with some care. To begin with, S must be a measurable Euclidean space [1]; let us denote the Lebesgue measure [2] by λ. Consider the following (mild) regularity assumption proposed in Andra *et al.* (2008),

Property 1. *(MDP Regularity) S is a compact subset of the d_S-dimensional Euclidean space, A is a compact subset of $[-A_\infty, A_\infty]^{d_A}$. The random immediate rewards are bounded by $\hat{R}_{\max}$ and R is uniformly bounded by $R_{\max} : \|r\|_\infty \leq R_{\max}$.*

We define the evaluation operator; $T^\pi : B\left(S \times A\right) \to B(S \times A)$,

$$T^\pi Q\left(s,a\right) = R\left(s,a\right) + \gamma \int_{S,A} \mathrm{d}s' \,\mathrm{d}a' \; P\left(\,\mathrm{d}s'|s,a\right)\pi\left(a'|s'\right) Q\left(s',a'\right).$$

We now define the analogue of the Q-value function recurrence relation, $Q^\pi = T^\pi Q^\pi$. The fixed point of the Bellman operator $T : B(S \times A) \to B(S \times A)$,

$$TQ\left(s,a\right) = R\left(s,a\right) + \gamma \int_S \mathrm{d}s' \sup_{a' \in A} P\left(\,\mathrm{d}s'|s,a\right) Q\left(s',a'\right),$$

$Q^* = TQ^*$ is then the optimal value function. By the regularity conditions imposed in 1, V^π and Q^π are both bounded by $\frac{R_{\max}}{1-\gamma}$.

In order to approach learning an optimal policy, we must estimate the current value function, Q. One method is to use a function approximator. Another popular scheme is the Fitted Q-iteration approach, wherein the Q function is estimated by regressing on a finite trajectory collected using a stationary policy π_b. Let $[S_t, A_t, R_t]_{1 \leq t \leq N}$, be the dataset, and then Q_{k+1} can be got by,

$$Q_{k+1} = \text{Regress}\left(\left\{\left[\left(S_t, A_t\right), R_t + \gamma \max_{a' \in A} Q_k\left(X_{t+1}, a'\right)\right]_{1 \leq t \leq N-1}\right\}\right).$$

The regression itself can be solved using a variety of methods, including neural networks and SVMs. A further discussion on suitable regression meth-

[1]Roughly, a space X is said to be measurable if a monotonic function $\mu : 2^X \to \mathbb{R}$ that is additive over finite union can be defined.

[2]The Lebesgue measure is a generalisation of length/area. It is possible to construct a Lebesgue measure for any Euclidean space.

ods, and a proof of convergence subject to niceness assumptions can be found in Andra *et al.* (2008).

2.2 Spatial Abstractions: Homomorphisms

A homomorphism is defined in general to be a structure-preserving map. In the context of MDPs, we want a correspondence between the two systems' dynamics (P) and tasks (R).

Definition 1. *(MDP Homomorphism) An MDP homomorphisms h from an MDP $M = \langle S, A, P, R, \gamma \rangle$ to and MDP $M' = \langle S', A', P', R', \gamma \rangle$ is a surjection for the state-action spaces $S \times A \to S' \times A'$ such that,*

$$P'\left(h\left(s,a\right), \underline{s'}\right) = \sum_{s' \in f^{-1}\left(\underline{s'}\right)} P\left(s,a,s'\right) \tag{2.1}$$

$$R'\left(h\left(s,a\right)\right) = R\left(s,a\right), \tag{2.2}$$

where f is h restricted to S, i.e. $f\left(s\right) = h\left(s,a\right)|_S$, and $\underline{s'}$ is the image of s' in M', i.e. $\underline{s'} = f\left(s'\right)$.

With minor abuse of notation, we assume $P\left(h\left(s,a\right), \underline{s'}\right) \triangleq P\left(\underline{s}, \underline{a}, \underline{s'}\right)$, where $h\left(s,a\right) = \left(\underline{s}, \underline{a}\right)$. Similarly, $Q\left(h\left(s,a\right)\right) = Q\left(\underline{s}, \underline{a}\right)$. For brevity, we will write $M \xrightarrow{h} M'$ if M' is the homomorphic image of M under h. We will also use the notation $(s,a) \in M$ to mean $(s,a) \in S \times A$. Finally, two state-action pairs (s,a) and (s',a') are equivalent if $h\left(s,a\right) = h\left(s,a'\right) = \left(\underline{s}, \underline{a}\right)$. We write this as $(s,a) \sim (s',a')$.

There can be other, less strict definitions of homomorphisms based on preserving the value function, policy behaviour, etc. A survey of these definitions can be found in Li *et al.* (2006*b*).

Given this definition, there are perhaps four serious questions that we wish to answer when talking about homomorphisms,

1. (Transfer) Given a homomorphisms g between MDPs M and M', can one use a policy learnt in M in M', and vice versa?
2. (Discovery) Given MDPs M and M', are they homomorphic?

3. (Minimisation) Given an MDP M, what is the smallest M' isomorphic to M?
4. (Approximate Minimisation) Given an MDP M, and a class of homomorphisms H, what is the closest approximation $M' \in H(M)$ of M?

All of these questions have been convincingly answered in the case of discrete MDPs.

2.2.1 Transferring Policies in Discrete MDPs

The primary application of MDP homomorphisms is in the transfer of learning in problem to another.

Definition 2. *(Lifted Policy) Given $M \xrightarrow{h} M'$, and a policy π' in M' we can define a* lifted policy $\pi = h^{-1}(\pi')$ *in M as follows,*

$$\pi(s,a) \triangleq \frac{\pi'(h(s,a))}{|\{a' : h(s,a') = h(s,a)\}|}. \tag{2.3}$$

$$= \frac{\pi'(h(s,a))}{|\{a' : (s,a') \sim (s,a)\}|}. \tag{2.4}$$

Note that the denominator equally distributes the selection probability between equivalent actions for a state.

A powerful result that we will later review shows that the lifted policy of an optimal policy is itself optimal. This is a special case of the result that the value of the lifted policy is equal through out M to the value of the original policy in M', i.e. for all s, $V^{\pi}(s) = V^{\pi'}(\underline{s})$.

Note 1. *(Partial Homomorphisms and Relativised Options)*

The ability to 'lift' policies from one MDP to another can be exploited to solve sub-problems in a reinforcement learning domain. Consider the grid-world shown in Figure 2.2(a) taken from Ravindran and Barto (2003); in this domain, there are several rotated copies of the room shown in Figure 2.2(b). For each room in (a), let us define a homomorphism mapping each cell within the room to those of the template room (b). All states outside the room are mapped to the 'sink' state, denoted by the black square in

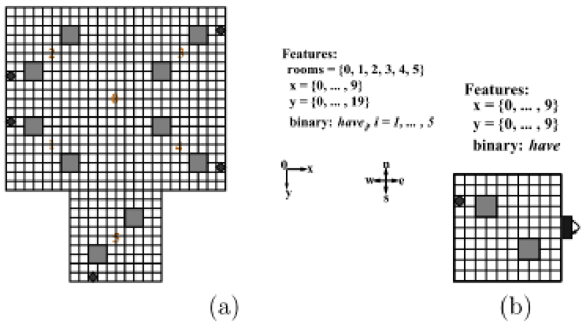


Figure 2.2: Relativised Options

(b). We can now define options *Sutton* et al. *(1999) for each room, using the optimal policy for (b), π, and the above homomorphisms; $\mathcal{O} = \{{h_r}^{-1}\pi\left(a\right) \mid \text{for each room } r\}$.*

2.3 Temporal Abstractions: The Options Framework

The options framework provides a temporal abstraction through subtasks. An option $\langle\mathcal{I}, \pi, \beta\rangle$ is described by an initiation set $\mathcal{I} \subset S$, a policy π, and a terminating condition β. An agent can exercise an option in any state $s \in \mathcal{I}$, following which, it will follow the policy π described by the option, until the terminating condition $\beta\left(s\right)$ is satisfied. The terminating condition β can be stochastic.

Several learning algorithms have been proposed for agents using options Sutton *et al.* (1999); Barto and Mahadevan (2003). One simple such method that we will use is MacroQ, a generalisation of the Q-learning algorithm described above. The MacroQ algorithm updates the value function only after completion of the option. If the option o was initiated in the state s, and continues for k steps before terminating in s', the corresponding Q function update will be,

$$Q\left(s, o\right) \;=\; Q\left(s, o\right) + \alpha\left[r + \gamma^k \max_{o' \in A \cup \mathcal{O}} Q\left(s', o'\right) - Q\left(s, o\right)\right].$$

Different tasks in the same domain can be described by different R. Let R be sampled from the family $\mathcal{R}$. Our objective then is to find a set of options O that reduces the expected learning time over $\mathcal{R}$.

Example 1. *To make the discussion more tangible, let us look at an example, the Taxi domain, shown in Figure 2.3. The agent is a taxi navigating in this road-map. It must*

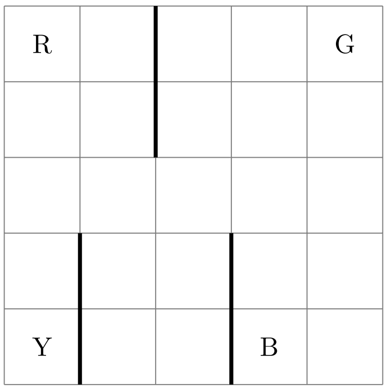


Figure 2.3: The Taxi Domain

pick up a passenger at one of the 4 pads, R, G, B or Y. Subsequently, it must carry the passenger to a destination, which is also one of the above four pads. The states of the taxi would then be a tuple containing the location of the passenger (in one of the four pads, or within the taxi), the destination of the passenger, and location of the taxi in the map. The actions the taxi can perform are moving up, down, left or right in the map, as well as pick up or drop a passenger at a pad. Typical options for such a domain would be an option that can be started anywhere, and has a policy that takes the taxi to the one of the pads in the shortest possible manner. Such an option is generic, and does not depend on where the passenger or destination are. The RL agent must then learn to choose the right option when picking up the passenger.

2.4 Small World Networks

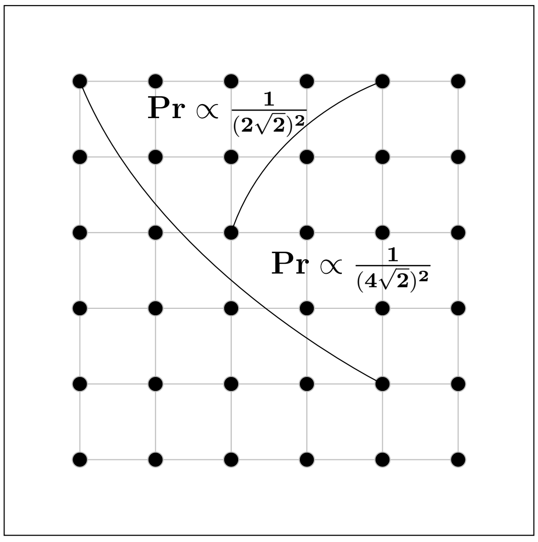

$\mathbf{Pr} \propto \frac{1}{(2\sqrt{2})^2}$
$\mathbf{Pr} \propto \frac{1}{(4\sqrt{2})^2}$

Figure 2.4: Kleinberg's Small World Network

In Kleinberg's small-world network model (Figure 2.4), each node u is given one 'long-range' edge to a node v, which was chosen with a probability $P_r(u,v) \propto \|u-v\|^{-r}$, where $\|u-v\|$ denotes the least distance between nodes u and v in the graph. On an r-dimensional lattice, $\mathcal{K}_r$, the distance from any node u to a target node t is bounded by $\|u-t\|$, a quantity which is locally computable. When given long-range edges distributed according to P_r, Kleinberg (2000) showed that a greedy distributed algorithm $\mathcal{GA}$ that chooses a neighbour v closest to t will reach t with an expected time $O\left(\log\left(|V|\right)^2\right)$. This result is significant because the expected time for the algorithm on a graph with edges distributed uniformly, i.e. without dependence on distance, is $O\left(|V|^2\right)$.

Theorem 3. *Let $f : V \to \mathbb{R}$ be a vertex function, and M_f be the global maxima of f. Consider an algorithm, $\mathcal{GA}$, that greedily chooses the neighbour with the highest value of f. Suppose the algorithm is currently at u, it will choose $N(u) = \operatorname{argmin}_v \|f(v) - f(M_f)\|$.*

If $\mathcal{G}(V,E)$ is r-dimensional lattice, and contains a long distance edge distributed according to $P_r : p(u,v) \propto \|u-v\|^{-r}$, then $\mathcal{GA}$ takes $O\left((\log|V|)^2\right)$ steps to reach M_f.

Proof. We defer the proof of this theorem to Section 5.2, where we prove a generalisation of this theorem (Theorem 9). □

CHAPTER 3

Related Work

In this chapter we will review existing methods for option-discovery and transfer learning.

3.1 MDP Minimisation and Homomorphism Discovery in Discrete MDPs

In order to answer the minimisation question, Narayanamurthy and Ravindran reduced the MDP homomorphism query to a graph automorphism problem in Narayanamurthy and Ravindran (2008) by constructing an equivalent weighted digraph from an MDP $M\langle S, A, P, R\rangle$.

We briefly describe the reduction: Construct a graph G_M, with S as nodes. For each node s, add an edge to the node s' if there is some action a that takes the agent from s to s'. Each edge is weighted with the vectors, $\langle p_{a_1}, \cdots, p_{a_{|A|}}\rangle$ and $\langle r_{a_1}, \cdots, r_{a_{|A|}}\rangle$, where $p_{a_i} = P(s, a_i, s')$, and $r_{a_i} = R(s, a_i, s')$. We could also view this as a construction of two graphs, one for P, and the other for R; the graph isomorphisms we are looking for belong to the common subset.

There are cases for which using the symmetry group may not suffice to capture the symmetries present; Section 4.1.1 of Ravindran (2004) presents one such example. This method can not be extended to continuous domains easily, as the number of states is no longer countable, and hence a reduction to a graph isomorphism problem is not possible.

3.2 Approximate Homomorphisms for Discrete MDPs

In practice, exact symmetries are rarely found. In such situations, approximate homomorphisms may be more suitable. The most common approach is to aggregate states within a certain error bound ϵ of the transition and reward functions; this is the approach followed in Ravindran and Barto (2004) and Taylor *et al.* (2009). Both papers also present a bound on the approximation loss, which is linearly dependent on the maximum error in expected reward (R) and transition probability.

Another approach to approximate homomorphisms, proposed by Sorg and Singh (2009), is to probabilistically map states onto the homomorphic image through "soft homomorphisms"; i.e. $h : S \to \mathcal{X}(S')$, such that,

$$\begin{aligned}\sum_{\underline{s} \in S'} P'\left(\underline{s}, a, \underline{s'}\right) h\left(\underline{s}|s\right) &= \sum_{s' \in S} P\left(s, a, s'\right) h\left(\underline{s'}|s'\right) \\ \sum_{\underline{s} \in S'} R'\left(\underline{s}, a\right) h\left(\underline{s}|s\right) &= R\left(s, a\right).\end{aligned}$$

Finding these soft homomorphisms can be shown to be a quadratic program,

$$\begin{aligned}\forall_a H P'^a &= P^a H \\ \forall_a H R'^a &= R^a \\ H 1_{|S'|} &= 1_{|S|} \\ H_{ij} &\geq 0.\end{aligned}$$

where,

$$\begin{aligned}H : |S| \times |S'| &= h\left(\underline{s'}|s\right) \\ P^a : |S| \times |S| &= P\left(s'|s, a\right) \\ R^a : |S| \times 1 &= R\left(s, a\right) \\ P'^a : |S'| \times |S'| &= P\left(\underline{s'}|\underline{s}, a\right) \\ R'^a : |S'| \times 1 &= R\left(\underline{s}, a\right).\end{aligned}$$

Sorg and Singh (2009) further show how soft homomorphisms can be used to map learning to a scaled up version of the state space, as well as map a continuous state space to a discrete one. One of the drawbacks of this approach however, is the lack of state dependent action recoding.

Of the two approaches, using soft homomorphisms seems more easily applicable in the continuous domain. In fact, we attempted to approach the problem in this fashion, however we found the optimisation problem to be intractable (Section A.2).

3.3 Option Discovery

One of the first option discovery schemes was by McGovern and Barto (2001), who identify states that are visited frequently through an empirical count. The problem is cast as a multiple-instance learning problem, where an option or "concept" that explains the most number of successful trajectories is chosen. Our approach of constructing options from multiple trajectories is similar to McGovern and Barto (2001)'s method of using several trajectories to find the minimal explanation.

Menache *et al.* (2002) approaches the option discovery also through the angle of finding bottlenecks. They make the observations that bottleneck states join well connected regions of the state-space graph, and use a graph cut algorithm to find bottlenecks, which are nodes in the cut set.

The state-of-the-art option discovery technique uses the betweenness scores of states as a measure of importance (Şimşek and Barto, 2008). This is an intuitive measure, since graph betweenness measures the proportion of paths that go through a particular node. In a domain like Taxi, the pickup and drop actions are states with high betweenness values. In navigational domains likes Rooms, doorways are high betweenness centers. Şimşek and Barto (2008) also show how a locally constructed model of the MDP can be used to compute betweenness scores.

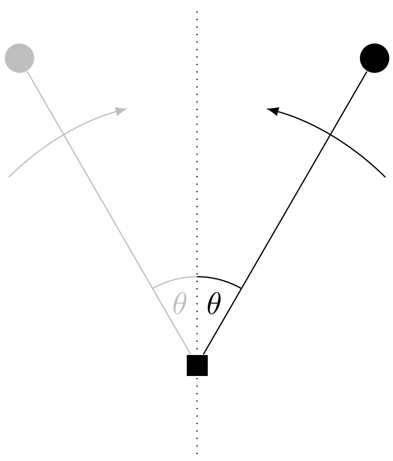
CHAPTER 4

Searching for Homomorphisms

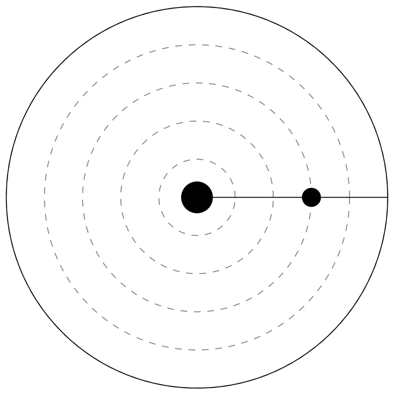
In this chapter, we explore spatial homomorphisms in continuous domains. We begin by describing the challenges of such a definition (Section 4.1), and describe a extension to existing MDP homomorphism definitions that tackles them (Section 4.2). As an aside, we detail an interesting family of continuous homomorphisms, namely continuous affine homomorphisms in Section 4.2.1. We develop an online algorithm to find homomorphisms from this family in Section 4.3, and evaluate this algorithm in Section 4.4.

4.1 Continuous Homomorphisms

An important property of symmetries in continuous domains is that they may reduce the dimensionality of the domain; the type of symmetries present in discrete domains can only "fold" the state space in finitely many ways. To motivate this distinction, consider the following examples,



(a) Discrete Symmetry



(b) Continuous Symmetry

4.1.1 Inverted Pendulum

In this example, the agent is aware of a single continuous variable, the angular displacement of the pole from the vertical, and has either two discrete actions to move to the left or to the right with a certain acceleration/impulse, or a continuous range of acceleration/impulse (Figure 4.1a). There is an obvious symmetry of reflection about the vertical. This is an example of a finite symmetry group. The dimensionality of the homomorphic space remains unchanged.

4.1.2 Reach the Center

Imagine a circular disc-world wherein an agent must navigate to the center of the disc. The only relevant variable here is the radial distance, and all points along a circle with a particular radius are homomorphically equivalent (Figure 4.1b). This is an example of an infinite symmetry group. The dimensionality of the homomorphic space has reduced to 1 from 2.

In general, the dimensionality of the homomorphic image reduces by the "height" of the symmetry group; this is a standard result from algebra.

4.2 Well-Definedness of Continuous MDPs

In order to capture continuous symmetries in our framework, we need a definition of continuous MDP homomorphisms that captures both finite and infinite pre-images. This problem is resolved if, instead of maps between points, we considered maps between *closed topological sets*. The circle described in the previous example, while composed of an infinite number of points is still a closed set, and so is its image, a single point. Topologically speaking, a map that takes closed sets to closed sets is continuous. The coincidence that homomorphisms in continuous domains must themselves be continuous allows us to unambiguously use the term "continuous homomorphisms" in the sequel.

Definition 4. *(Continuous MDP Homomorphism) An continuous MDP homomorphisms h from a continuous MDP $M = \langle S, A, P, R, \gamma \rangle$ to and a continuous MDP $M' = \langle S', A', P', R', \gamma \rangle$ is a* continuous *surjection $S \times A \to S' \times A'$ such that,*

$$\begin{aligned} P'\left(h\left(s,a\right),\underline{s'}\right) &= \int_{f^{-1}\left(\underline{s'}\right)} \mathrm{d}s'\, P\left(s,a,s'\right) && (4.1) \\ R'\left(h\left(s,a\right)\right) &= R\left(s,a\right), && (4.2) \end{aligned}$$

where f is h restricted to S, i.e. $f(s) = h(s,a)\,|_S$, and $\underline{s'}$ is the image of s' in M', i.e. $\underline{s'} = f(s')$.

The continuity condition also ensures that $f^{-1}(\underline{s'})$ is a well defined, measurable set. Though $f^{-1}(\underline{s'})$ may be a finite set, we slightly abuse notation and use $\int_{f^{-1}(\underline{s'})}$ uniformly; the integral must be replaced by a summation in case $f^{-1}(\underline{s'})$ is finite.

For the remainder of this section, unless specified otherwise, we will assume all MDPs and homomorphisms are continuous. The lifted policy of a continuous MDP is defined as follows.

Definition 5. *(Continuous Lifted Policy) Given $M \xrightarrow{h} M'$, and a policy π' in M' we can define a* lifted policy *$\pi = h^{-1}(\pi')$ in M as follows,*

$$\pi(s,a) \triangleq \frac{\pi'(h(s,a))}{\int_{h_s^{-1}(\underline{a})} \mathrm{d}a}. \tag{4.3}$$

where $h_s^{-1}(\underline{a})$ is set of actions equivalent to a in the state s, i.e. $\{a \mid h(s,a) = (\underline{s}, \underline{a})\}$.

We will now prove the value equivalence between an continuous MDP and its homomorphic image and show that the above definitions are sufficient for this purpose.

Lemma 6. *Let $M \xrightarrow{h} M'$. Let π' be any policy in M', and π be its lifted policy in M'. Define $V^{\pi}(s) = \int_A \mathrm{d}a\, \pi(s,a)\, Q^{\pi}(s,a)$. If $Q^{\pi}(s,a) = Q^{\pi'}(h(s,a))$, then $V^{\pi}(s) = V^{\pi'}(\underline{s})$.*

Proof. This follows directly from the definition of the lifted policy π.

$$\begin{aligned}
V^{\pi}(s) &= \int_A \mathrm{d}a\, \pi(s,a)\, Q^{\pi}(s,a) \\
&= \int_{A'} \mathrm{d}\underline{a} \int_{h_s^{-1}(\underline{a})} \mathrm{d}a\, \pi(s,a)\, Q^{\pi}(s,a) \\
&= \int_{A'} \mathrm{d}\underline{a} \int_{h_s^{-1}(\underline{a})} \mathrm{d}a\, \frac{\pi'(\underline{s},\underline{a})}{\int_{h_s^{-1}(\underline{a})} \mathrm{d}a} Q^{\pi'}(\underline{s},\underline{a}) \\
&= \int_{h_s^{-1}(\underline{a})} \pi'(\underline{s},\underline{a})\, Q^{\pi'}(\underline{s},\underline{a}) \\
&= V^{\pi'}(\underline{s}).
\end{aligned}$$

□

Theorem 7. *(Continuous Value Equivalence) Let $M \xrightarrow{h} M'$. Let π' be any policy in M', and π be its lifted policy in M'. For any $(s,a) \in S \times A$, $Q^{\pi}(s,a) = Q^{\pi'}(h(s,a))$.*

Proof. Consider the recursive definition of the m-step discounted action value function,

$$Q^{\pi}_{m}(s,a) = R(s,a) + \gamma \int_{S} \mathrm{d}s'\, P(s,a,s') \int_{A} \mathrm{d}a'\, \pi(s',a')\, Q^{\pi}_{m-1}(s',a'),$$

with $Q^{\pi}_{-1}(s,a) = 0$ for any $(s,a) \in M$. We will also define an m-step discounted value function $V^{\pi}_{m}(s) = \int_{A} \mathrm{d}a\, \pi(s,a)\, Q^{\pi}_{m}(s,a)$. We can rewrite the previous equation as,

$$Q^{\pi}_{m}(s,a) = R(s,a) + \gamma \int_{S} \mathrm{d}s'\, P(s,a,s')\, V^{\pi}_{m-1}(s').$$

For the base case, consider the case when $m = 0$. Then, $Q^{\pi}_{0}(s,a) = R(s,a) = R'(h(s,a)) = Q^{\pi'}_{0}(h(s,a))$. Assuming $Q^{\pi}_{j}(s,a) = Q^{\pi'}_{j}(h(s,a))$ for all $(s,a) \in M$ and all $j < m$, we must show that $Q^{\pi}_{m}(s,a) = Q^{\pi'}_{m}(h(s,a))$.

$$\begin{aligned} Q^{\pi}_{m}(s,a) &= R(s,a) + \gamma \int_{S} \mathrm{d}s'\, P(s,a,s')\, V^{\pi}_{m-1}(s') \\ &= R(s,a) + \gamma \int_{S'} \mathrm{d}\underline{s'} \left(\int_{f^{-1}(\underline{s'})} \mathrm{d}s'\, P(s,a,s') \right) V^{\pi'}_{m-1}(\underline{s'}) \\ &= R'(h(s,a)) + \gamma \int_{S'} \mathrm{d}\underline{s'}\, P'(h(s,a), \underline{s'})\, V^{\pi'}_{m-1}(\underline{s'}) \\ &= Q^{\pi'}_{m}(h(s,a)). \end{aligned}$$

Given that $Q^{*}(s,a) = \lim_{m\to\infty} Q_{m}(s,a)$, we have $Q^{*}(s,a) = Q^{*}(h(s,a))$. □

4.2.1 Continuous Affine Homomorphisms

An interesting family of homomorphisms that we will consider in detail are that of continuous *affine* homomorphisms. These homomorphisms relate two spaces through a combination of rotation and translation. A continuous affine homomorphism is parameterised by A, an orthogonal matrix corresponding to rotation, and b a vector corresponding to translation. In addition, we can project the rotated vector to a smaller state space with the rectangular identity matrix, $I_{m,n}$, where m and n are the dimensions of M' (the image) and M respectively. Thus, $h(x) = I_{m,n}Ax + b$.

Example 2. *(Expressive Power of Continuous Affine Homomorphisms)*

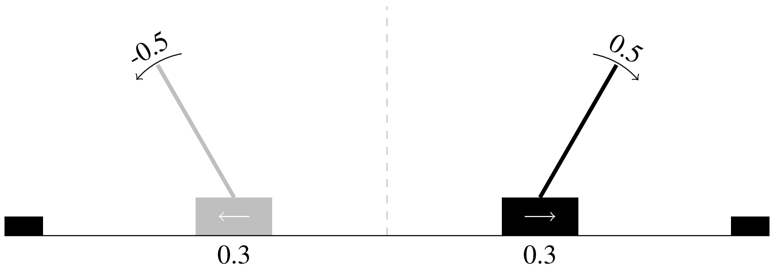


Figure 4.1: Cart Pole

Let us study the capabilities of the continuous affine homomorphism family, with the Cart Pole task as an example (Figure 4.1). In this task, the agent must balance a pole on a cart without moving the cart out of the boundaries; the only action the agent can perform is to push the cart forward or backward with some force. The state variables are the x position of the agent, the angular displacement from the vertical, and the linear and angular velocities of the cart and pole respectively.

In a bounded domain, there is a single exact automorphism; a reflection of all the state features about the centre of the track. This instance is trivially captured by the continuous affine family, $A = -I$ and $b = 0$. In an unbounded domain, however, the whole system is translation-invariant, i.e. the x position does not matter. In such a case, A could zero out the contribution of x, and b translate the system to any coordinate. This homomorphism is a reasonable approximation even in bounded domains.

An interesting reduction we could study in this domain would be to project the angular state space coordinates, and lift a policy from an inverted pendulum. A is simply the permutation matrix that shifts the angular components to the first two features.

4.3 Homomorphic Filters

A number of factors make searching for exact homomorphisms impractical. Rarely is the environment model exactly known, particularly in some closed form that it can be exploited. Furthermore, exact homomorphisms are extremely sensitive to noise, thus rendering them inapplicable to any learnt models. Given these conditions, it is more prudent to search for approximate homomorphisms. In this section, we describe an online algorithm to find homomorphisms from the current environment, M, to an environment the agent has encountered before, M'. We assume that we have a model of M', but not of M.

In principle, we would like to find a homomorphism that minimises the expected difference between $x' \sim P_M(x,a)$ and $\underline{x'} \sim P_{M'}(h(x,a))$, as well as $R_M(x,a)$ and $R_{M'}(h(x,a))$. To this end, we propose the following objective function,

$$\begin{aligned} C(h) &= \int_{S\times A} \mathrm{d}x\,\mathrm{d}a\, C(h,x,a) \\ C(h,x,a) &= \frac{1}{2}\mathbb{E}\left[\left(\underline{x'} - h(x')\right)^2\right] + \frac{1}{2}\mathbb{E}\left[\left(R_{M'}(\underline{x}) - R_M(x)\right)^2\right]. \end{aligned} \tag{4.4}$$

Without a model for M, the above expectations can not be evaluated, let alone minimised. However, we can use samples of x, a, x' and r collected by the agent's interaction with the world to perform stochastic gradient descent,

$$\begin{aligned} h_{t+1} &\leftarrow h_t - \alpha\langle\nabla_h C(h,x,a), \mathcal{H}\rangle \\ \nabla_h C &= \frac{1}{2}\nabla_h\left[\mathrm{tr}\left(K(h(x))\right) + \left(m(h(x)) - h(x')\right)^2 + \left(R(h(x')) - r\right)^2\right] \\ \nabla_h C &= \frac{1}{2}\nabla\,\mathrm{tr}\left(K(h(x))\right)\cdot\nabla_h h(x) + \left(m(h(x)) - h(x')\right)\left(\nabla m(h(x))\cdot\nabla_h h(x) - \nabla_h h(x')\right) \\ &+ \left(R(h(x')) - r\right)\nabla R(h(x'))\cdot\nabla_h h(x'). \end{aligned}$$

m and K are the mean and co-variance of M'. Note that we project the updates onto $\mathcal{H}$, $\langle\nabla_h C(h,x,a), \mathcal{H}\rangle$.

Finally, as $C(h)$ is in general a complex non-convex function, we perform the stochastic gradient descent simultaneously on a number of starting points, or particles. The algorithm is summarised in Algorithm 1. The Q-value at any point of the homomorphic filter is computed as an expectation over each particle, with probabilities proportional to $\exp\left(-\frac{C\left(\tilde{h}^j,x,a\right)}{\tau}\right)$.

Algorithm 1: Homomorphic Filtering

```
1 Initialise $M$ particles $h_0^{(i)}$ randomly from $\mathcal{H}$
2 while not converged do
3     foreach particle $i$ do $\tilde{h}_{t+1}^i \leftarrow h_t^i + \alpha\langle\nabla_h C, \mathcal{H}\rangle$
4     foreach particle $i$ do $h_{t+1}^i \sim \tilde{h}_{t+1}$ with probability $\propto \exp\left(-\frac{C\left(\tilde{h}^j,x,a\right)}{\tau}\right)$
```

4.3.1 Continuous Affine Homomorphisms

One possible choice for $\mathcal{H}$ is the set of affine transformations between the state spaces, i.e. $h(x) = I_{m,n}Ax + b$, where m and n are dimensions of M' and M respectively, A is an arbitrary orthogonal matrix, and b is an arbitrary translation. Below, we derive the necessary update equations for this family of homomorphisms,

$$\begin{aligned} \frac{\partial}{\partial A_{ij}} h(x) &= I_{m,n} 1_i x_j & \frac{\partial}{\partial A} f(h(x)) &= \nabla f(h(x))\, I_{m,n} x^T \\ \frac{\partial}{\partial b_i} h(x) &= 1_i & \frac{\partial}{\partial b} f(h(x)) &= \nabla f(h(x)). \end{aligned}$$

$$\begin{aligned} \frac{\partial}{\partial A} C &= \frac{1}{2}\nabla \operatorname{tr}(K(h(x)))\, x^T + \sum_i (m_i(h(x)) - h(x'))\left(\nabla m_i(h(x)) \cdot I_{m,n}^T x^T - 1 \cdot I_{m,n}^T x'^T\right) \\ &+ (R(h(x')) - r)\, \nabla R(h(x'))\, I_{m,n} x'^T \\ \frac{\partial}{\partial b} C &= \sum_i \left(\frac{1}{2}\nabla K_i(h(x)) + (m_i(h(x)) - h(x'))(\nabla m_i(h(x)) - 1)\right) \\ &+ (R(h(x')) - r)\, \nabla R(h(x')). \end{aligned}$$

The new A can be projected onto the set of orthogonal matrices, by finding the nearest orthogonal matrix, $A\left(A^T A\right)^{-\frac{1}{2}}$.

4.4 Experimental Results

We evaluated our algorithm on the Cart Pole domain. The domain describes an agent that must balance a pole on a cart by applying a linear force to the cart. The domain has four state variables, θ, the angular displacement of the pole from the vertical, x the linear displacement of the cart from the center, and their derivatives, $\dot{\theta}$ and $\dot{x}$. The linear force the agent can apply is discretised into 4 actions.

We used fitted Q-iteration with support vector regression through out the experiments to learn the value function. We initially chose Gaussian process regression to learn the domain models because it provided a covariance function. However, despite the uniform noise in our experiments, and hence independence from coordinates, the learnt models did not accurately predict the variance, and led to greater variability in performance. We decided to omit the contribution of variance to the gradient descent updates, and used the models learnt using support vector regression for the remainder of the experiments for efficiency reasons.

The Cart Pole domain has a single exact homomorphism which is a complete reflection of the coordinates, and reversal of the forces applied. Since the action space is discrete, we also added *an action permutation* to each particle. Our algorithm found homomorphisms which a combination of reflections of the angular components, and linear displacements of cart along the track, both of which are intuitive. When the track on which the cart moves is bounded, any two positions along the track, except for mirror reflections, are *not* equivalent. However, the difference in their values is reduces as the cart moves away from the wall; in limiting case, when the track is unbounded, the x displacement indeed does not matter.

l, m_p, m_c	**Using Original π**	**Using HF π**	**New Agent**
0.5, 0.1, 1.0	-11.003	1.612	2.079

Table 4.1: Lifted Policy on Perturbed Domain (Return after $1,600$ epochs)

We perturbed the domain by changing some of its parameters (e.g. length of the pole, masses, etc.), as well as by rotating the state space coordinates, and observed the performance of the algorithm. We present the return accumulated by the agent after $1,600$ epochs observations in Table 4.1; in general, the lifted policy of the homomorphic filter was competitive to a trained agent on the perturbed domain, considerably better than the trained agent in the original domain.

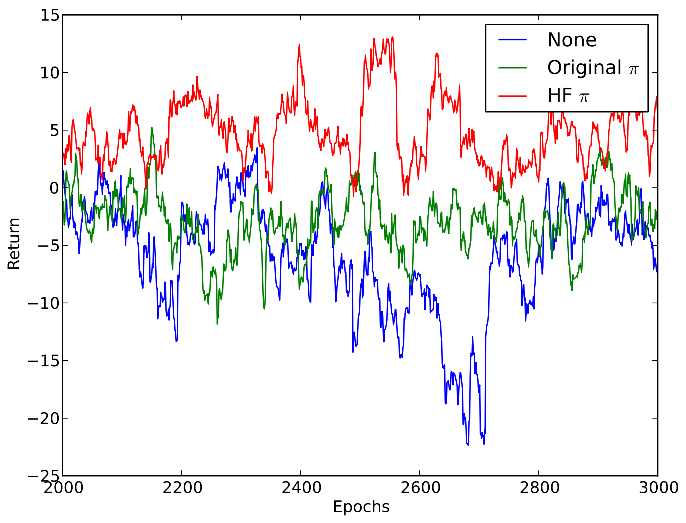


Figure 4.2: Bootstrapping Learning in a Perturbed Domain

We also evaluated the benefit of using the homomorphic filter to bootstrap the agent using a slight variant of multiple model reinforcement learning (Doya *et al.*, 2002). Initially, the agent relies more on the policy of the homomorphic filter, but as the agent's value function estimate improves, it gradually begins to use its value function more (Figure 4.2). We compared the performance of an agent learning without any other model (None), with the policy learnt in the original task (Original π), and the policy recommended by the homomorphic filter (HF π). The agent using the homomorphic filter significantly outperforms the other two agents. Note that the agent using the original policy without any homomorphisms actually negatively effects the learning of the agent.

CHAPTER 5

Small World Options

In this chapter, we describe how an MDP can be viewed as a graph, and how Kleinberg's small-world model can be adapted to this setting (Section 5.1). We then prove that small world options indeed guarantee a logarithmic bound on the number of decisions taken by the agent (Section 5.2). In line with our general theme of abstraction learning schemes relying only information readily available to the agent, we describe an efficient algorithm to learn small world options completely through trajectories taken by the agent (Section 5.3). Finally, we evaluate the small world options, comparing the options against another state of the art technique in Section 5.4.

5.1 Graph View of MDPs

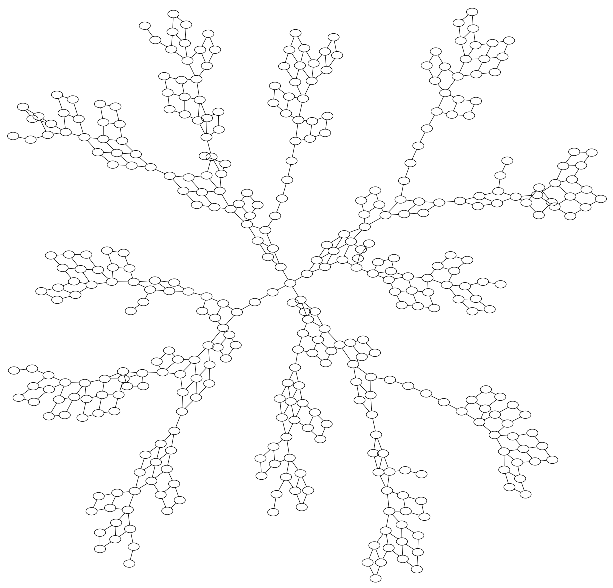


Figure 5.1: The State Space Graph for Taxi

It is easy to construct a graph $\mathcal{G}_M$ out of the state-space described by an MDP. The states S become the nodes of the graph, and actions A become the edges, with the transition probabilities as weights. Options can be viewed as paths along the graph. The Taxi domain defined earlier (Example 1) translates to the graph shown in Figure 5.1.

We draw a parallel to the network structure in Kleinberg's small-world network model (Figure 2.4), by adding a single 'path option' from each state s to another state s', where s' is chosen with probability $P_r(s, s') \propto \|s - s'\|^{-r}$. A path option $o_p(s, s')$ is an option with $\mathcal{I} = \{s\}$, $\beta = \{s'\}$, and an optimal policy to reach s' for π. Intuitively, it is an option that takes the agent from s to s'. In practice, we may generate path options for only a subset of $|S|$. Note that while this results in $O(|S|)$ options, only one additional option is available in any state, and thus the decision-space for the agent is not significantly larger.

5.2 Small World Structure in MDPs

Consider an MDP $M_{\mathcal{K}_r}$ with states connected in a r-dimensional lattice, and noisy navigational actions between states. We claim that by using *robust* path options distributed according to P_r, an ϵ-greedy agent can reach a state of maximal value using $O\left(\log(|S|)^2\right)$ options, using the value function V as a local property of the state.

Definition 8. *A* robust path option *$o(u, v)$, where $u, v \in S$ is an option that takes the agent from u to v 'robustly', in the sense that in each epoch, the agent moves closer to v with a probability $1 - \epsilon > \frac{1}{2}$.* [1]*. Note that this ϵ includes any environmental effects as well.*

In order to handle the ϵ-greedy nature of the algorithm, as well as the approximateness in the distance, we will need to extend Kleinberg's theorem (Theorem 3).

Theorem 9. *Let $f : V \to \mathbb{R}$ be a function embedded on the graph $\mathcal{G}(V, E)$, such that, $\kappa_1\|u - v\| - c_1 \leq \|f(u) - f(v)\| \leq \kappa_2\|u - v\| - c_2$, where $0 \leq \kappa_1 \leq \kappa_2$, and $0 \leq c_2 \leq \frac{c_1}{2}$. Let M_f be the global maxima of f. Let $\mathcal{GA}_\epsilon$ be an ϵ-greedy algorithm with respect to f, i.e. an algorithm which chooses with probability $1 - \epsilon$ to transit to the neighbouring state closest to M_f, i.e. $N(u) = \operatorname{argmin}_v \|f(v) - f(M_f)\|$.*

[1]This condition is equivalent to saying that the option takes the agent from u to v in finite time, and hence is not particularly strong.

If $\mathcal{G}(V,E)$ is r-dimensional lattice, and contains a long distance edge distributed according to $P_r : p(u,v) \propto \|u-v\|^{-r}$, then $\mathcal{GA}_\epsilon$ takes $O\left((\log|V|)^2\right)$ steps to reach M_f.

Proof. This result is a simple extension of Kleinberg's result in Kleinberg (2000), and follows the proof presented there, albeit with the somewhat cleaner notation and formalism of Martel and Nguyen (2004). We begin by defining the necessary formalism to present the proof.

Definition 10. *Let us define $\mathrm{B}_l(u)$ to be the set of nodes contained within a "ball" of radius l centered at u, i.e. $\mathrm{B}_l(u) = \{v \mid \|u-v\| < l\}$, and $\mathrm{b}_l(u)$ to be the set of nodes on its surface, i.e. $\mathrm{b}_l(u) = \{v \mid \|u-v\| = l\}$.*

Given a function $f : V \to \mathbb{R}$ embedded on $\mathcal{G}(V,E)$, we analogously define $\mathrm{B^f}_l(u) = \{v \mid |f(u) - f(v)| < l\}$. For notational convenience, we take $\mathrm{B^f}_l$ to be $\mathrm{B^f}_l(M_f)$.

The inverse normalised coefficient for $p(u,v)$ is,

$$\begin{aligned} c_u &= \sum_{v \neq u} \|u-v\|^{-r} \\ &= \sum_{j=1}^{r(n-1)} \mathrm{b}_j(u)\, j^{-r}. \end{aligned}$$

It can easily be shown that the $\mathrm{b}_l(u) = \Theta\left(l^{k-1}\right)$. Thus, c_u reduces to a harmonic sum, and is hence equal to $\Theta(\log n)$. Thus, $p(u,v) = \|u-v\|^{-r}\Theta(\log n)^{-1}$.

We are now ready to prove that $\mathcal{GA}_\epsilon$ takes $O\left((\log|V|)^2\right)$ decisions. The essence of the proof is summarised in Figure 5.2. Let a node u be in phase j when $u \in \mathrm{B^f}_{2^{j+1}} \setminus \mathrm{B^f}_{2^j}$. The probability that phase j will end this step is equal to the probability that $N(u) \in \mathrm{B^f}_{2^j}$.

The size of $\mathrm{B^f}_{2^j}$ is at least $|\,\mathrm{B}_{\frac{2^j+c_2}{\kappa_2}}\,| = \Theta\left(\frac{2^j+c_2}{\kappa_2}\right)$. The distance between u and a node in $\mathrm{B^f}_{2^j}$ is at most $\frac{2^{j+1}+c_1}{\kappa_1} + \frac{2^j+c_2}{\kappa_2} < 2\left(\frac{2^{j+1}+c_2}{\kappa_2}\right)$. The probability of a link between these two nodes is at least $\left(\frac{2^{j+2}+2c_1}{\kappa_1}\right)^{-r}\Theta(\log n)^{-1}$. Thus,

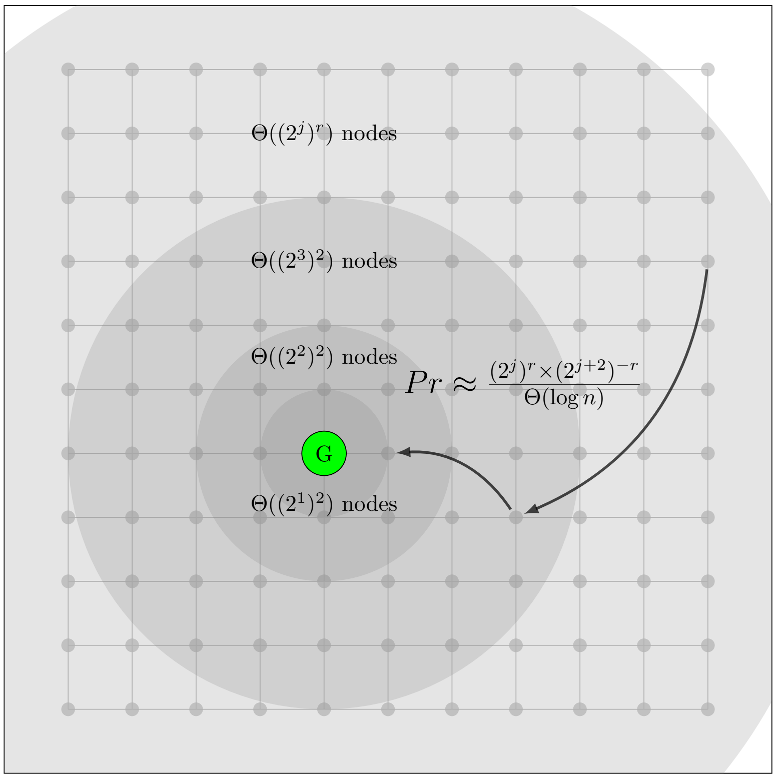


Figure 5.2: Exponential Neighbourhoods

$$\begin{aligned}
P\left(u, \mathrm{B^f}_{2^j}\right) &\geq \frac{(1-\epsilon)}{\Theta(\log n)} \left(\frac{2^j + c_2}{\kappa_2}\right)^r \times \left(\frac{2^{j+2} + 2c_1}{\kappa_1}\right)^{-r} \\
&\geq \frac{(1-\epsilon)}{\Theta(\log n)} \times \left(\frac{\kappa_1}{4\kappa_2}\right)^r \times \left(\frac{1 + \frac{c_2}{2^j}}{1 + \frac{c_1}{2 \times 2^j}}\right)^r \\
&\geq \frac{(1-\epsilon)}{\Theta(\log n)} \times \left(\frac{\kappa_1}{4\kappa_2}\right)^r \times \left(\frac{1 + c_2}{1 + \frac{c_1}{2}}\right)^r .
\end{aligned}$$

Let number of decisions required to leave phase j be X_j. Then,

$$\begin{aligned}
\mathbb{E}\left[X_j\right] &\leq \sum_{i=0}^{\infty} \left(1 - P\left(u, \mathrm{B^f}_{2^j}\right)\right)^i \\
&\leq \frac{1}{P\left(u, \mathrm{B^f}_{2^j}\right)} \\
&\leq \Theta(\log n) \frac{1}{(1-\epsilon)} \left(\frac{4\kappa_2}{\kappa_1}\right)^r \left(\frac{1 + \frac{c_1}{2}}{1 + c_2}\right)^r \\
&\leq \Theta(\log n) .
\end{aligned}$$

Thus, it takes at most $O(\log n)$ decisions to leave phase j. By construction, there are at most $\log n$ phases, and thus at most $O\left((\log n)^2\right)$ decisions. $\square$

All that remains is to bound the value function, V, in terms of the graph distance to the value maxima. We show this in the following lemma.

Lemma 11. *Let $o(u,v)$ be the preferred option in state u, and let $\|u-v\|_V = |\log \mathrm{V}(v) - \log \mathrm{V}(u)|$. Then,*

$$k_1\|u-v\| - c_1 \leq \qquad \|u-v\|_V \leq \qquad k_2\|u-v\|,$$

where $k_1 = \log\frac{1}{\gamma}$, $k_2 = \log\frac{1}{(1-\epsilon)\gamma}$, and $c_1 = \log\frac{1}{1-\gamma}$.

Proof. From the Bellman optimality condition, we get the value of $o(u,v)$ to be,

$$\mathrm{Q}(u, o(u,v)) \quad = \quad \mathbb{E}_l\left[\gamma^l\,\mathrm{V}(v) + \sum_{i=1}^{l}\gamma^{i-1}r_i\right],$$

where l is the length of the option, and r_i is the reward obtained in the i-th step of following the option.

If o(u,v) is the preferred option in state u, then $\mathrm{V}(u) = \mathrm{Q}(u, o(u,v))$. Using the property that $0 \leq r_i \leq 1$,

$$\begin{aligned}
\mathbb{E}_l\left[\gamma^l\,\mathrm{V}(v)\right] \leq \qquad & \mathrm{V}(u) \leq \qquad \mathbb{E}_l\left[\gamma^l\,\mathrm{V}(v) + \sum_{i=1}^{l}\gamma^{i-1}\right] \\
\mathbb{E}_l\left[\gamma^l\right]\mathrm{V}(v) \leq \qquad & \mathrm{V}(u) \leq \qquad \mathbb{E}_l\left[\gamma^l\right]\mathrm{V}(v) + \frac{1}{1-\gamma}. \qquad (5.1)
\end{aligned}$$

$\mathbb{E}_l$ is an expectation over the length of the option. Using the property that $o(u,v)$ is robust, we move closer to v with probability $\bar{\epsilon} = 1-\epsilon$; this is exactly the setting of the well-studied gambler's ruin problem, where the gambler begins with a budget of $\|u-v\|$, and wins with a probability of ϵ. Using a standard result from FellerFeller (1968), with $m = \|u-v\|$, we have,

$$E_l\left[x^l\right] = \sum_{l=0}^{\infty} P(L=l)\,x^l \quad = \quad \frac{1}{\lambda_1^m(x) + \lambda_2^m(x)},$$

where $\lambda_1(x) = \frac{1+\sqrt{1-4\epsilon\bar{\epsilon}x^2}}{2\bar{\epsilon}x}$, and $\lambda_2(x) = \frac{1-\sqrt{1-4\bar{\epsilon}\epsilon x^2}}{2\bar{\epsilon}x}$. When $x \le 1$,

$$\begin{aligned}
\frac{1}{\left(\lambda_1(x) + \lambda_2(x)\right)^m} \le \quad & E_l\left[x^l\right] \le \quad & \sum_{l=m}^{\infty} P(L=l)\, x^l \\
\frac{1}{\left(\frac{2}{2\bar{\epsilon}x}\right)^m} \le \quad & E_l\left[x^l\right] \le \quad & \sum_{l=m}^{\infty} P(L=l)\, x^m \\
(\bar{\epsilon}x)^m \le \quad & E_l\left[x^l\right] \le \quad & x^m.
\end{aligned}$$

Substituting $x = \gamma$ and $m = \|u-v\|$ into Equation (5.1), we get,

$$\begin{aligned}
\mathbb{E}_l\left[\gamma^l\right] \mathrm{V}(v) \le \quad & \mathrm{V}(u) \le \quad & \mathbb{E}_l\left[\gamma^l\right] \mathrm{V}(v) + \frac{1}{1-\gamma} \\
(\bar{\epsilon}\gamma)^{\|u-v\|} \mathrm{V}(v) \le \quad & \mathrm{V}(u) \le \quad & \gamma^{\|u-v\|} \mathrm{V}(v) + \frac{1}{1-\gamma} \\
\|u-v\| \log\frac{1}{\gamma} - \log\frac{1}{1-\gamma} \le \quad & \|u-v\|_V \le \quad & \|u-v\| \log\frac{1}{\bar{\epsilon}\gamma}.
\end{aligned}$$

□

Thus, an ϵ-greedy agent acting with respect to its value function can reach the maxima of the value function using just $O\left((\log|S|)^2\right)$ decisions. Though this result strictly applies only to the lattice setting, we observe that many MDPs are composed of lattice-like regions of local connectivity connected via bottleneck states. The presence of such bottleneck states would only increase the expected time by a constant factor.

5.3 Efficiently Constructing Small World Options

In Section 5.2, we remarked that we needed $O(|S|)$ options. In order to be practical, we require an algorithm to efficiently generate these options within a budget of training epochs. The proof of Theorem 9 provides us with a crucial insight – our options only need bring the agent into an exponentially smaller *neighbourhood* of the maximal value state. This suggests that cheaply generated options may still be acceptable.

The algorithm (Algorithm 2) we propose takes a given MDP M, and trains an agent to learn T different tasks (i.e. different R) on it, evenly dividing the epoch budget amongst them. With each learned task, we certainly will have a good policy for path options from any state to the state of maximal value, M_v. However, we observe that will also have a good policy for path options from u to v is the path is 'along the gradient' of Q, i.e. when $V(u) < V(v) < V(M_v)$. Observing that $V(s) \approx \text{argmax}_v\, Q(s, \pi(s))$, we detail the algorithm to construct many options options from a single Q-value function in Algorithm 3.

Algorithm 2: Small World Options from Experience

```
input : $M$, $\mathcal{R}$, $r$, $n$, epochs, $T$
1 $O \leftarrow \emptyset$
2 for $i \leftarrow 0$ to $T$ do
3     $R \sim \mathcal{R}$
4     $Q \leftarrow$ Solve $M$ with $R$ using $\frac{\text{epochs}}{T}$ epochs
5     $O' \leftarrow$ QOptions( $Q$, $r$, $\frac{n}{T}$ )
6     $O \leftarrow O \cup O'$
7 return A random subset of $n$ options from $O$
```

Algorithm 3: QOptions: Options from a Q-Value Function

```
input : $Q$, $r$, $n$
1 $O \leftarrow \emptyset$
2 $\pi \leftarrow$ greedy policy from $Q$
3 for $s$ in $S$ do
4     Choose an $s'$ according to $P_r$
5     if $Q(s', \pi(s')) > Q(s, \pi(s))$ then
6         $O \leftarrow O \cup \langle \{s\}, \pi, \{s'\} \cup \{t \mid Q(s', \pi(s')) < Q(t, \pi(t))\} \rangle$
7 return A random subset of $n$ options from $O$
```

We note here except for sampling s' from P_r, we do not require any knowledge of the MDP, nor do we need to construct a local model of the same. In fact, s' can be sampled approximately using the expected path length instead of the graph distance in P_r. As the expected path length $\mathbb{E}[l]$ is only a constant factor greater than l ($\frac{l}{\epsilon}$), Lemma 11 continues to hold.

5.4 Experimental Results

We trained MacroQ learning agents on several standard domains, and measured the cumulative return obtained using the following option generation schemes:

- **None**: No options were used.
- **Random**: Options were generated by randomly connecting two nodes in the domain (this is equivalent to P_0).
- **Betweenness**: As a representative of bottleneck-based schemes, options were generated to take any node to a local maxima of betweenness centrality, as described in Şimşek and Barto (2008).
- **Small World**: Options were generated randomly connecting two nodes of the domain using an inverse square law, as described in Section 5.2.

Each experiment, unless mentioned otherwise, was run for 10 randomly generated tasks in the domain; each task ran for $40,000$ epochs, and was averaged over an ensemble of 20 agents.

5.4.1 Optimal Options

The agents were run on the following three domains using the algorithm sketched in Section 5.2:

- **Arbitrary Navigation**: The agent must reach an arbitrary goal state in an obstacle-free $x \times y$ grid-world.
- **Rooms**: The agent must navigate a floor plan with 4 rooms to reach an arbitrary goal state.
- **Taxi**: This is the domain described in Example 1.

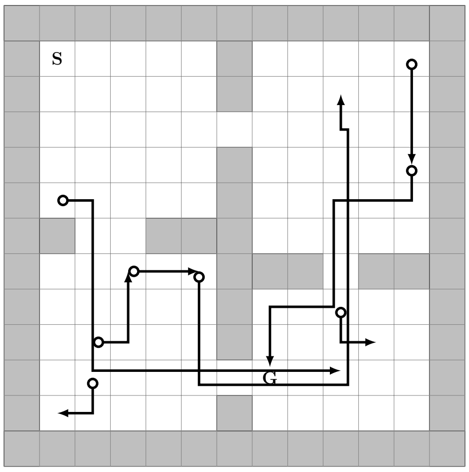
Optimal policies were given to the options generated according to the schemes described above.

The results of these experiments are summarised in Table Table 5.1. Small world options perform significantly better than the other schemes in navigation-oriented tasks like Rooms or Arbitrary Navigation. In the Taxi domain, options generated by the betweenness scheme outperform the small world op-

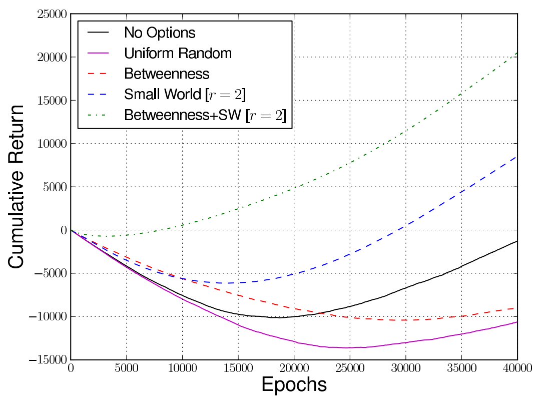
	Arbt. Navi	Rooms	Taxi
None	-31.82	-1.27	-16.90
Random	-31.23	-10.76	-18.83
Betw.	-18.28	-8.94	**80.48**
Sm-W	**-14.24** [$r = 4$]	**8.54**[$r = 2$]	0.66 [$r = 0.75$]

Table 5.1: Cumulative Return

tions. This is expected because the goal states in this domain lie at betweenness maxima.



(a) Rooms: Options learnt



(b) Rooms: Cumulative Return with 200 options

Some of the small world options preferred in Rooms domain are shown in Figure 5.3a. The graph shows several examples of options that compose together to arrive near the goal state. We have also plotted the learning behaviour in Figure 5.3b. The option scheme "Betweenness + SW" combines options to betweenness maxima with small world options. Expectedly, it significantly outperforms all other schemes. The options to betweenness maxima help take the agent between strongly connected regions, while the small world options help the agent navigate within the strongly connected region.

5.4.2 Sensitivity of r

Figure 5.3 plots r versus the cumulative return on the Rooms domain. We do not yet have a clear understanding of how the exponent r should be chosen. The performance of the agent without options after $20,000$ epochs is also plotted for reference. There is a range of r ($\approx$ 0.75 to 1.5) with good performance, after

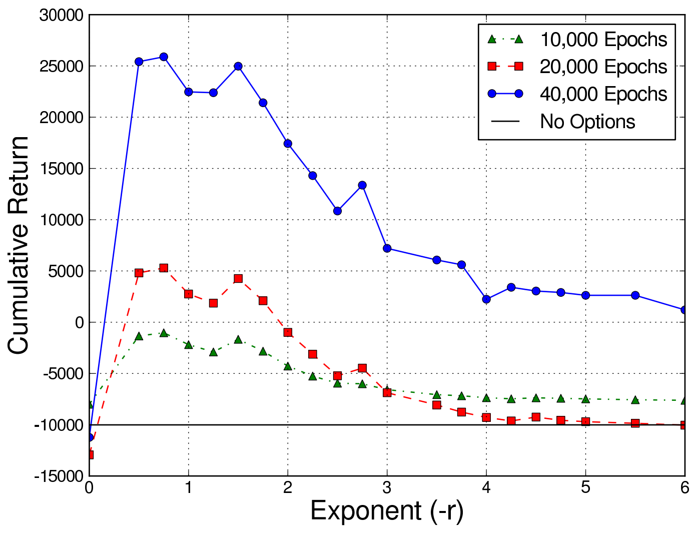


Figure 5.3: Rooms: r vs Cumulative Return

which the performance steadily drops. This behaviour is easily explained; as the exponent goes up, the small world options generated are very short, and do not help the agent get nearer to the maximal value state. The optimal range of r is slightly counter-intuitive because the Rooms domain is a two dimensional lattice with some edges removed. As a consequence of the reduced connectivity, and perhaps due to stochastic factors, longer range options are preferred.

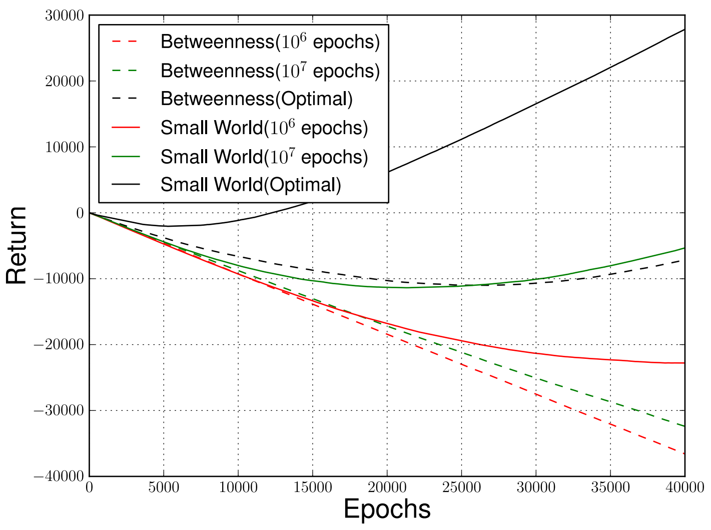


Figure 5.4: Rooms: Options Learnt on a Budget

5.4.3 Options Learnt on a Budget

In Section 5.3, we described an algorithm to construct small world options efficiently when given a limited number of learning epochs. We compared the performance of these options with betweenness options learnt with the same total number of epochs, and have plotted our results in Figure 5.4. Despite using many more options, the small world options thus created significantly outperform betweenness options learnt with the same budget, and are even comparable to the optimal betweenness options.

CHAPTER 6

Conclusions and Future Directions

6.1 Homomorphic Filters

We described an online algorithm to learn MDP homomorphisms in continuous state spaces through gradient descent in the family of homomorphisms, and evaluated the same using the family of affine homomorphisms. This family of homomorphisms subsumes many existing selection algorithms which only consider variable remappings. When run on the Cart Pole domain, the algorithm finds intuitively obvious approximate homomorphisms which an exact homomorphism solver could not find. The lifted policy in perturbed domains performs comparably to an agent trained to learn in that domain. We also used the lifted policy to bootstrap an agent in the perturbed domain, and observed that the agent performed better than its counterpart without the lifted policy.

We believe the homomorphic filter is a novel approach to finding continuous homomorphisms, backed by a solid theoretical foundation in MDP homomorphisms. Of particular interest to the authors would be to study if complex tasks could be solved given models of simpler subtasks. For example, could we learn how to behave in the Cart Pole domain faster if we were given a model of an inverted pendulum. This approach motivates the use of self-paced learning in reinforcement learning. Though it is straightforward to extend the current work to continuous action spaces, it remains to be seen how well homomorphic filtering performs in such domains. Finally, though we have restricted ourselves to the class of affine homomorphisms, the method described is general enough to capture other differentiable families, for example regression trees. Studying alternative homomorphism classes is planned future work.

6.2 Small World Options

We have devised a new scheme to generate options based on small world network model. The options generated satisfy an intuitive criteria, that the subtasks learnt should be easily composed to solve any other task. The options greatly improve the connectivity properties of the domain, without leading to a state space blow up.

Experiments run on standard domains show significantly faster learning rates using small world options. At the same time, we have shown that learning small world options can be cheaper than learning bottleneck options, using a natural algorithm that extracts options from a handful of tasks it has solved. Another advantage of the scheme is that is does not require a model of the MDP.

As future work, we would like to characterise what the exponent r should be in a general domain. There are some technicalities to be worked out in extending our results to the continuous domain; however, as most real-life applications are continuous in nature, this is an important further direction we are looking at. Given the ease with which options can be discovered, it would be interesting to experiment with a dynamic scheme that adds options on the fly, while solving tasks. Liben-Nowell *et al.* (2005) extend Kleinberg's results to arbitrary graphs by using rank instead of lattice distance. It would be interesting to extend this approach to the reinforcement learning setting. The logarithmic bounds on the number of decisions presented may have some interesting consequences on theoretical guarantees of sample complexity as well.

CHAPTER 7

Publications from this Work

1. Poster at the 9th European Workshop for Reinforcement Learning (EWRL 2011), titled "Learning in a Small World" described preliminary work of Chapter 5.
2. Full Paper for Oral Presentation at the 11th international conference on Autonomous Agents and Multi-agent Systems (AAMAS 2012), titled "Learning in a Small World" described the work in Chapter 5.
3. Paper submitted for review at the 10th European Workshop for Reinforcement Learning (EWRL 2012), titled "Discovering Continuous Homomorphisms" described the work in Chapter 4.

APPENDIX A

An Alternate Approach to Find Continuous Homomorphisms

In this appendix, we describe a technique we attempted to find continuous homomorphisms. We did not continue, as the approach soon became intractable.

A.1 Modelling the State Space with a Gaussian Process

We would like to unify our representation of continuous MDPs, using Gaussian Processes (GP) as standard model; GPs can be easily learnt from trajectories through the state space, and also model the stochasticity of the world dynamics. A major limitation of using a Gaussian process to model action behaviours is uni-modality; only actions which move the agent in one direction, with noise, can be modelled. This limitation could be overcome using a mixture of Gaussian processes, but that is out of the scope of this thesis.

Consider a state space S. Let us assume that S is a manifold, with a mapping into $\mathbb{R}^n$, ξ. For each point $x \in \mathbb{R}^n$, there is an associated *vector of Gaussian processes*, $\vec{a}$, representing the behaviour of the actions. The transition dynamics can be described by the following equation,

$$T\left(s, a, s'\right) = \frac{1}{Z} \exp\Big(\left(\xi\left(s'\right) - m_a\left(\xi\left(s\right)\right)\right)^T K_a^{-1}\left(\xi\left(s\right)\right)\left(\xi\left(s'\right) - m_a\left(\xi\left(s\right)\right)\right)\Big),$$

where m and K, are the mean and covariance of the Gaussian process. Continuous actions can also be handled, using $m\left(\xi\left(s\right), \xi\left(a\right)\right)$ and $K^{-1}\left(\xi\left(s\right), \xi\left(a\right)\right)$ in place of the respective subscripted versions.

For the sake of notational convenience, we'll take $S = \mathbb{R}^n$ in the sequel. The results we present can be applied to the general continuous state space through

ξ.

$$\begin{tikzcd}
x \arrow[r, dotted, "f"] \arrow[d, "T_a"] & y \arrow[d, "T_a"] \\
x' \arrow[r, dotted, "f"] & y' \\
& \mathbf{S}
\end{tikzcd}$$

Figure A.1: Bijective Automorphisms

Now, stating our problem in this framework, we would like to find bijective automorphisms $f : S \to S$ that preserve the behaviour of the transition dynamics described by T_a for all actions $a \in A$. At a high level, the diagram Figure A.1 describes what we hope to achieve. Mathematically,

$$\begin{aligned}
T_a\left(f\left(s\right), f\left(s'\right)\right) - T_a\left(s, s'\right) &= 0 \\
\frac{1}{\left|2\pi K_a\left(f\left(s\right)\right)\right|}\exp\Big(\left(f\left(s'\right) - m_a\left(f\left(s\right)\right)\right)^T K_a^{-1}\left(f\left(s\right)\right)\left(f\left(s'\right) - m_a\left(f\left(s\right)\right)\right)\Big) &- \\
\frac{1}{\left|2\pi K_a\left(s\right)\right|}\exp\Big(\left(s' - m_a\left(s\right)\right)^T K_a^{-1}\left(s\right)\left(s' - m_a\left(s\right)\right)\Big) &= 0.
\end{aligned}$$

Aside from being in general intractable to solve, the above definition is too strict for any practical domain. We will now explore two relaxations that will let us find approximate homomorphisms.

A.2 Bayesian Approach

In this section, we will consider a Bayesian approach wherein the mapping f is itself a Gaussian process,

$$f\left(x, y\right) = \frac{1}{Z}\exp\Big(\left(y - n\left(x\right)\right)^T L^{-1}\left(x\right)\left(y - n\left(x\right)\right)\Big)$$

. We know that the probability of y transitioning to y', given x and x' are the respective pre-images of y and y', is $T_a\left(x, x'\right)$; $P\left(y \to y' | x \to y, x' \to y'\right) = P\left(x \to x'\right)$. Thus, we have,

$$\begin{aligned} P\left(y \to y'\right) &= \int \mathrm{d}x\,\mathrm{d}x'\, P\left(y \to y' | x \to y, x' \to y'\right) P\left(x \to y\right) P\left(x' \to y'\right) \\ T_a\left(y, y'\right) &= \int \mathrm{d}x\,\mathrm{d}x'\, T_a\left(x, x'\right) f\left(x, y\right) f\left(x', y'\right). \end{aligned}$$

In the approximate setting, we would like to minimise the KL divergence between the left and right-hand sides.

$$\begin{aligned} T_a\left(y, y'\right) &\approx \int \mathrm{d}x f\left(x, y\right) \mathbb{E}_{x'}\left[f\left(x', y'\right)\right]. \\ \mathcal{L}\left(y\right) &= \mathbb{E}_{y'}\left[\log T_a\left(y, y'\right) - \log \int \mathrm{d}x f\left(x, y\right) \mathbb{E}_{x'}\left[f\left(x', y'\right)\right]\right]. \end{aligned}$$

If $\int \mathrm{d}x f\left(x, y\right) = 1$, then we can apply Jensen's inequality to get,

$$\mathcal{L}\left(y\right) \geq \mathbb{E}_{y'}\left[\log T_a\left(y, y'\right) - \int \mathrm{d}x f\left(x, y\right) \mathbb{E}_{x'}\left[\log f\left(x', y'\right)\right]\right].$$

Note that $\log f\left(x', y'\right)$ contains terms of complexity at most $\exp\left(x^T A x + b^T x + c\right)$, which can be evaluated by Gaussian integral.

Note 2. *We can also take,*

$$\mathcal{L}\left(y\right) \geq \left[\log T_a\left(y, y'\right) - \int \mathrm{d}x f\left(x, y\right) \mathbb{E}_{x'}\left[\log \mathbb{E}_{y'}\left[f\left(x', y'\right)\right]\right]\right].$$

Note that $\mathbb{E}_{y'}$ here is over a different distribution*, namely $T_a\left(y, y'\right)$ than $f\left(x', y'\right)$, and hence not equal to 1.*

To proceed in this direction, we need a prior on $P\left(x\right)$, which is hard to define. More importantly, an implict condition is that $\int \mathrm{d}x P\left(x\right) f\left(x, y\right) = 1$. This involves integrating over the mean function of a Gaussian process; using the typical Gaussian kernel, the form of the integrand is e^{e^x}, which is intractable.

REFERENCES

1. **Andra, A.**, **R. Munos**, and **C. Szepesvári**, Fitted Q-iteration in continuous action-space MDPs. *In NIPS*. 2008.

2. **Barto, A. G.** and **S. Mahadevan** (2003). Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems*, 1–28.

3. **Şimşek, O.** and **A. G. Barto**, Skill characterization based on betweenness. *In NIPS*. 2008.

4. **Doya, K.**, **K. Samejima**, **K.-i. Katagiri**, and **M. Kawato** (2002). Multiple model-based reinforcement learning. *Neural computation*, **14**(6), 1347–69.

5. **Engel, Y.**, **P. Szabo**, and **D. Volkinshtein**, Learning to Control an Octopus Arm with Gaussian Process Temporal Difference Methods. *In Advances in Neural Information Processing Systems*, volume c. 2006.

6. **Feller, W.**, *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 1968.

7. **Guestrin, C.**, **D. Koller**, **R. Parr**, and **S. Venkataraman** (2003). Efficient Solution Algorithms for Factored MDPs. *Journal of Machine Learning Research*, **19**, 399–468.

8. **Jerrum, M.** and **A. Sinclair**, Conductance and the rapid mixing property for markov chains: the approximation of permanent resolved. *In Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88. ACM, New York, NY, USA, 1988. ISBN 0-89791-264-0. URL `http://doi.acm.org/10.1145/62212.62234`.

9. **Kleinberg, J.** (2000). The Small-World Phenomenon : An Algorithmic Perspective. *ACM Theory of Computing*, **32**, 163–170.

10. **Li, L.**, **T. J. Walsh**, and **M. L. Littman**, Towards a Unified Theory of State Abstraction for MDPs. *In In Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*. 2006*a*.

11. **Li, L.**, **T. J. Walsh**, and **M. L. Littman**, Towards a Unified Theory of State Abstraction for MDPs. *In Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*. 2006*b*.

12. **Liben-Nowell, D.**, **J. Novak**, **R. Kumar**, **P. Raghavan**, and **A. Tomkins** (2005). Geographic routing in social networks. *PNAS*, 1–6.

13. **Martel, C.** and **V. Nguyen**, Analyzing Kleinberg's (and other) Small-world Models. *In PODC*, volume 2. 2004. ISBN 1581138024.

14. **McGovern, A.** and **A. G. Barto**, Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. *In ICML*. 2001.

15. **Menache, I.**, **S. Mannor**, and **N. Shimkin**, Q-Cut - Dynamic Discovery of Sub-Goals in Reinforcement Learning. *In ECML*. 2002.

16. **Narayanamurthy, S. M.** and **B. Ravindran**, On the hardness of finding symmetries in Markov decision processes. *In Proceedings of the 25th international conference on Machine learning - ICML '08*. ACM Press, New York, New York, USA, 2008. ISBN 9781605582054. URL `http://portal.acm.org/citation.cfm?doid=1390156.1390243`.

17. **Ravindran, B.** (2004). *An Algebraic Approach To Abstraction In Reinforcement Learning*. Ph.D. thesis, University of Massachusetts, Amherst.

18. **Ravindran, B.** and **A. G. Barto**, Relativized Options : Choosing the Right Transformation. *In International Conference on Machine Learning*. 2003.

19. **Ravindran, B.** and **A. G. Barto**, Approximate Homomorphisms : A framework for non-exact minimization in Markov Decision Processes. *In Knowledge Based Computer Systems*. 2004.

20. **Soni, V.** and **S. Singh**, Using Homomorphisms to Transfer Options across Continuous Reinforcement Learning Domains. *In AAAI*. 2006.

21. **Sorg, J.** and **S. Singh**, Transfer via Soft Homomorphisms. *In AAMAS*. 2009.

22. **Sutton, R. S.**, **D. Precup**, and **S. Singh** (1999). Between MDPs and Semi-MDPs : Learning , Planning , and Representing Knowledge at Multiple Temporal Scales at Multiple Temporal Scales. *Artificial Intelligence*, **112**, 181–211.

23. **Tahbaz-Salehi, A.** and **A. Jadbabaie**, Small world phenomenon, rapidly mixing markov chains, and average consensus algorithms. *In Decision and Control, 2007 46th IEEE Conference on*. 2007. ISSN 0191-2216.

24. **Taylor, J. J.**, **D. Precup**, and **P. Panangaden**, Bounding Performance Loss in Approximate MDP Homomorphisms. *In NIPS*. 2009.

25. **Taylor, M. E.**, **G. Kuhlmann**, and **P. Stone**, Autonomous Transfer for Reinforcement Learning. *In Autonomous Agents and Multi-Agent Systems*, May. 2008.

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

27. **Taylor, M. E.**, **S. Whiteson**, and **P. Stone**, Transfer via Inter-Task Mappings in Policy Search Reinforcement Learning. *In Autonomous Agents and Multi-Agent Systems*, May. 2007. ISBN 1595930949.

28. **Thrun, S.** and **A. Schwartz**, Finding Structure in Reinforcement Learning. *In Advances in Neural Information Processing Systems 7*. 1995.