Discovering Continuous Homomorphisms for Transfer

Arun Tejasvi Chaganty
arunchaganty@gmail.com

Balaraman Ravindran
ravi@cse.iitm.ac.in
Department of Computer Science and Engineering,
Indian Institute of Technology, Madras,
Chennai, India 600036

Abstract

Generalisation across domains is a key problem for the reinforcement learning community. Spatial abstractions, such as MDP homomorphisms, allow an agent to relate its exploratory behaviour in a new domain to one it has already seen. Learning these abstractions remains a challenge, particularly in continuous state spaces. We extend MDP homomorphisms to continuous state and action spaces, and propose an online model-free algorithm for the agent to search for spatial abstractions using gradient descent over the space of continuous linear homomorphisms. We show how this approach can effectively be used to lift policies. We also demonstrate how it can be used to substantially reduce the exploration required by an agent in a new domain. Our framework easily extends to other families of homomorphisms.

Keywords: reinforcement learning, transfer learning, MDP homomorphisms.

1. Introduction

Many real world applications of reinforcement learning are described by complex continuous dynamical systems. Engel et al. (2006), for example, studies the problem of controlling a single octopus arm, with 88 state dimensions, and 22 action dimensions. In order to gracefully scale to the control of the remaining seven arms, we need to be able to generalise across continuous domains. Furthermore, it is impractical if not impossible to model such systems exactly, so we must also be able to learn abstractions using approximate models.

MDP homomorphisms (Ravindran and Barto, 2003) 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 Section 3, 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.

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.

The rest of the paper is organised as follows. We present an overview of MDP homomorphisms in discrete spaces in Section 2. Following a brief discussion on symmetries in continuous domains, we describe continuous MDP homomorphisms, and prove value equivalence of lifted policies in Section 3. We describe an algorithm that uses gradient descent to find an approximate continuous homomorphism in Section 4. We evaluate our technique on various transformations of the Cart Pole domain, and present our results in Section 5. Finally, we summarise our findings and describe future directions in Section 6.

2. Background

We will mainly be dealing with reinforcement learning in continuous state spaces with continuous or discrete actions. Our environment is represented, as usual, by a Markov Decision Process (MDP), i.e. the tuple, \((S, A, P, R, \gamma)\), where \(S\) and \(A\) are the state and action domains which are known to the agent. The state space is assumed to be a compact measurable subset of \(\mathbb{R}^n\); if the action space is also continuous, the same assumption applies for the action space. The transition dynamics of the world is described by \(P : S \times A \rightarrow \mathcal{X}(S)\), where \(\mathcal{X}(S)\) is the set of all probability measures over \(S\). For simplicity, we consider the reward function to be dependent only on the target state, i.e. \(R : S \rightarrow \mathbb{R}\). Finally, \(\gamma \in [0, 1]\) is the discount factor.

As usual, our objective is to find a policy, \(\pi : S \rightarrow \mathcal{X}(A)\) that maximises the expected return of the agent in the domain. This policy is found using a value-based optimal control algorithm that maintains an estimate of the \(Q\)-function. We use fitted Q-iteration (Riedmiller, 2005; Andra et al., 2008), wherein the \(Q\) function is estimated by regressing on a finite trajectory collected using a stationary policy \(\pi_b\). Let \([S_t, A_t, R_t]_{1 \leq t \leq N}\), be the dataset, then \(Q\) can be got by iterating,

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

A variety of regressors, from neural networks to regression trees to support vector regressors can be used to fit the \(Q\)-value function.

2.1. MDP Homomorphisms

In general, a homomorphism is a structure-preserving map; in the context of MDPs, we would like preserve the structure of the transition dynamics \((P)\) and rewards \((R)\).
Figure 1: Symmetries in Continuous MDPs

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

$$P' (h(s,a), s') = \sum_{s' \in h^{-1}(s)} P(s,a,s') \quad (1)$$

$$R' (f(s)) = R(s), \quad (2)$$

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

By allowing $h$ to range across the state-action space, we can capture symmetries requiring state-dependent action recoding. This allows us to capture, for example, the mirrored-equivalence between the two sides of an inverted pendulum that an agent is trying to balance.

MDP homomorphisms can be used to minimise a state space through the aggregation of equivalent states. They can also be used to transfer policies between domains through lifting. Given an MDP $M$, and it’s homomorphic image $\tilde{M}$, the lifted policy of $\pi'$ in $\tilde{M}$ in $M$ is,

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

The value function of this policy, $V^\pi$ is guaranteed to be equivalent to $V^{\pi'}$, i.e. $V^\pi (s) = V^{\pi'} (s)$. As a corollary, if $\pi'$ is optimal in $\tilde{M}$, then the lifted policy $\pi$ will be optimal in $M$.

### 3. Continuous Homomorphisms

Symmetries in continuous spaces can be of two types, (a) discrete symmetries, wherein the dimensionality of the state space is preserved, and (b) continuous symmetries which can map an infinite number of states to a single state. To motivate this distinction, consider the following examples,

1. **Discrete Symmetry**: Consider a task where an agent must balance an inverted pendulum. Any force applied when the rod is on one side of the vertical is equivalent to the same force applied in a reverse sense on the other side. This allows the state space can be “folded” about the vertical, and there are at most two states equivalent to each other (Figure 1(a)).
2. **Continuous Symmetry:** Imagine a circular disc-world wherein an agent must navigate to the centre of a disc. The only relevant variable here is the radial distance, and the infinite set of points along a circle with a particular radius are homomorphically equivalent to each other (Figure 1(b)). Note that the dimensionality of the homomorphic space has reduced from 2 to 1.

In order to capture continuous symmetries in our framework, we need a definition of continuous MDP homomorphisms that capture 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. However, conveniently, a continuous map is exactly one that takes closed sets to closed sets; thus, we may proceed using continuous functions between the two state-action spaces. Note that points are always closed sets.

**Definition 2 (Continuous MDP Homomorphism)** A continuous MDP homomorphisms $h$ from a continuous MDP $M = \langle S, A, P, R, \gamma \rangle$ to a continuous MDP $M_0 = \langle S_0, A_0, P_0, R_0, \gamma_0 \rangle$ is a continuous surjection $S \times A \rightarrow S_0 \times A_0$ such that,

$$
P'(h(s,a),s') = \int_{f^{-1}(s')} ds' P(s,a,s') \tag{3}
$$

$$
R'(f(s)) = R(s), \tag{4}
$$

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

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

The lifted policy of a continuous MDP is defined as follows.

**Definition 3 (Continuous Lifted Policy)** Given $M \xrightarrow{h} M$, and a policy $\pi'$ in $M$ we can define a lifted policy $\pi = h^{-1}(\pi')$ in $M$ as follows,

$$
\pi(s,a) \triangleq \frac{\pi'(h(s,a))}{\int da' \delta(h(s,a') - h(s,a))}. \tag{5}
$$

where $\delta$ is the Dirac delta function. The denominator represents the count of equivalent actions.

It is now straightforward to prove the value equivalence of lifted policies. We state the crucial lemma and theorem, with only a brief sketch of the proof in the interest of space.

**Lemma 4** Let $M \xrightarrow{h} M$. Let $\pi'$ be any policy in $M$, and $\pi$ be its lifted policy in $M$. Define $V_m^\pi(s) = \int_A da \pi(s,a) Q_m^\pi(s,a)$. If $Q_m^\pi(s,a) = Q_m^{\pi'}(h(s,a))$, then $V_m^\pi(s) = V_m^{\pi'}(s)$.

**Proof** Decompose the integral $\int_A da$ into $\int_A' da \int da \delta(h(s,a) - (f(s),a))$, and apply the definition of the lifted policy $\pi$. The result directly follows. □
**Theorem 5** (Continuous Value Equivalence) Let $M \xrightarrow{h} M$. Let $\pi'$ be any policy in $M$, and $\pi$ be its lifted policy in $M$. For any $(s, a) \in S \times A$, $Q^\pi_m(s, a) = Q^{\pi'}_m(h(s, a))$.

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

$$Q^\pi_m(s, a) = R(s) + \gamma \int_S ds' P(s, a, s') \int_A da' \pi(s', a') Q^\pi_{m-1}(s', a'),$$

with $Q^\pi_{-1}(s, a) = 0$ for any $(s, a) \in M$. Express $Q^\pi_m$ in terms of the value function $V^\pi_m$, and apply the previous lemma to show that $Q^\pi_m = Q^{\pi'}_m$ for all $m$. Given that $Q^\pi(s, a) = \lim_{m \to \infty} Q^\pi_m(s, a)$, we have $Q^\pi(s, a) = Q^{\pi'}(h(s, a))$. \hfill \blacksquare

### 4. 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, $\tilde{M}$. We assume that we have a model of $\tilde{M}$, but not of $M$.

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

$$C(h) = \int_{S \times A} ds \int_A da C(h, s, a)$$

$$C(h, s, a) = \frac{1}{2} \mathbb{E}_{s', f(s), s'} \left( (s' - f(s'))^2 \right) + \frac{1}{2} \left(R_M(f(s)) - R_{\tilde{M}}(s)\right)^2.$$

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

$$h_{t+1} \leftarrow h_t - \alpha \langle \nabla_h C(h, s, a), H \rangle$$

$$\nabla_h C = \frac{1}{2} \nabla_h \left[ \text{tr} (K(f(s))) + (m(f(s)) - f(s'))^2 + (R(f(s')) - r)^2 \right]$$

$$\nabla_h C = \frac{1}{2} \nabla \text{tr} (K(f(s))) \cdot \nabla_f s + (m(f(s)) - f(s')) \left( \nabla m(f(s)) \cdot \nabla_h f(s) - \nabla_h f(s') \right)$$

$$+ \left(R(f(s')) - r\right) \nabla R(f(s')) \cdot \nabla_h f(s').$$

$m$ and $K$ are the expected values of the first and second moments of $P(h(s, a))$, i.e. the mean and co-variance of the transition function in the MDP image. Finally, as $C(h)$

---

1. Though not included in the original submission, we have been able to use this objective function to bound the difference between the values of the original and lifted policies.
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 $\exp\left(-\frac{C(h^j, s, a)}{\tau}\right)$.

**Algorithm 1: Homomorphic Filtering**

Initialise $M$ particles $h_0^{(i)}$ randomly from $\mathcal{H}$;

while not converged do

foreach particle $i$ do

$h^i_{t+1} \leftarrow h^i_t - \alpha \langle \nabla h C, \mathcal{H} \rangle$;

endforeach

foreach particle $i$ do

$h^i_{t+1} \sim \tilde{h}^i_{t+1}$ with probability $\propto \exp\left(-\frac{C(\tilde{h}^j, s, a)}{\tau}\right)$

end

4.1. 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 we have used $x$ to denote a column vector with $s$ stacked on top of $a$. $m$ and $n$ are dimensions of $M$ and $M$ respectively, $I_{m,n}$ is an $M \times N$ identity matrix, $A$ is an arbitrary orthogonal matrix belonging to the subspace $S \otimes A^2$, and $b$ is an arbitrary translation. The derivatives of $h(x)$ with respect to the parameters are,

$$
\frac{\partial}{\partial A_{ij}} h(x) = I_{m,n}1_i x_j \\
\frac{\partial}{\partial b_i} h(x) = 1_i
$$

where $g$ is an arbitrary function. Finally, the derivatives of the cost function with respect to the parameters are,

$$
\frac{\partial}{\partial A} C = \frac{1}{2} \nabla \text{tr} (K(h(x))) x^T + \sum_i (m_i(h(x)) - h(x)) (\nabla m_i(h(x)) \cdot I_{m,n} x^T - 1 \cdot I_{m,n} x'^T)
$$

$$
+ \left( R\left(f\left(x'\right)\right) - r \right) \nabla R\left(f\left(x'\right)\right) I_{m,n} x'^T
$$

$$
\frac{\partial}{\partial b} C = \sum_i \left( \frac{1}{2} \nabla K_i(h(x)) + (m_i(h(x)) - h(x)) (\nabla m_i(h(x)) - 1) \right)
$$

$$
+ \left( R\left(f\left(x'\right)\right) - r \right) \nabla R\left(f\left(x'\right)\right).
$$

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

2. This is to ensure that $f$ is well defined.
5. 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, $\theta$, 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 throughout the experiments to learn the value function. Due to convergence issues while using Gaussian process regression to learn the mean and variance of the environmental models, we ignored the contribution of variance and used support vector regression instead.

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 used an action permutation to each particle. Our algorithm found homomorphisms which is 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 reduced as the cart moves away from the wall; in limiting case, when the track is unbounded, the $x$ displacement indeed does not matter.

Table 1: Performance of Lifted Policy on Perturbed Domain (Return after 1600 epochs)

<table>
<thead>
<tr>
<th>$l$, $m_p$, $m_c$</th>
<th>Using Original $\pi$</th>
<th>Using HF $\pi$</th>
<th>New Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5, 0.1, 1.0</td>
<td>-11.003</td>
<td>1.612</td>
<td>2.079</td>
</tr>
</tbody>
</table>

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 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.

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. We compared the performance of an agent learning without any other model (None), with the policy learnt in the original task (Original $\pi$), and the policy recommended by the homomorphic filter (HF $\pi$). 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.

6. Conclusions

We describe 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.

References


