Transfer via Soft Homomorphisms

Jonathan Sorg
University of Michigan
2260 Hayward Street
Ann Arbor, Michigan
jdsorg@umich.edu

Satinder Singh
University of Michigan
2260 Hayward Street
Ann Arbor, Michigan
baveja@umich.edu

ABSTRACT

The field of transfer learning aims to speed up learning across multiple related tasks by transferring knowledge between source and target tasks. Past work has shown that when the tasks are specified as Markov Decision Processes (MDPs), a function that maps states in the target task to similar states in the source task can be used to transfer many types of knowledge. Current approaches for autonomously learning such functions are inefficient or require domain knowledge and lack theoretical guarantees of performance. We devise a novel approach that learns a stochastic mapping between tasks. Using this mapping, we present two algorithms for autonomous transfer learning – one that has strong convergence guarantees and another approximate method that learns online from experience. Extending existing work on MDP homomorphisms, we present theoretical guarantees for the quality of a transferred value function.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

transfer learning, Markov decision process, homomorphism

1. INTRODUCTION

Humans are adept at applying knowledge learned in previous task experiences to aid in accomplishing similar future tasks. The field of *transfer learning* aims to produce the same ability in artificial agents. Central to this ability to transfer knowledge from a *source task* to another *target task* is the existence of structural similarities between two tasks. In this work, we adopt the *Reinforcement Learning* (RL) formalism of Markov Decision Processes (MDPs) for specifying tasks.

The challenge of transfer learning can be divided into two conceptual steps. First, the structural similarities between source and target tasks must be identified. Second, they must be exploited in such a way that performance in the target task can be improved. These two problems are intricately related, however; the form of the structural similarities often determines the types of knowledge that may be transferred. One general form of structural similarity that can be exploited in many ways involves finding an equivalence relationship between states in one task and states in another. Given this relationship, we can transfer a multitude of knowledge structures including value functions [10], policies [11], and options [2], among others. Many current methods rely on hand-coded structural mappings. Although methods exist for learning a mapping, current methods still require hand-coded domain knowledge [11, 3] or they explicitly enumerate all potential state-mapping functions from a small candidate set [8, 9]. These methods generally lack theoretical guarantees of performance.

This idea of a mapping between states in one MDP to states in another is closely related to abstraction and model minimization in MDPs [1] and MDP homomorphisms [5]. The goal of transfer learning is slightly different than that of abstraction. In the abstraction setting, the method is free to design an arbitrary abstract (source) task, which it then subsequently solves for the optimal policy. In the transfer setting, the agent selects a solved source task from its past experience. During transfer, the agent does not have the freedom to design an arbitrary source task. However, it is assumed the source task has already been solved. In both settings, information learned in the *source* task – a value function, policy, etc. – is used to improve performance in the *target* task. This process of transferring a structure from source to target is known as *lifting*.

The novel aspect of our method is that we adopt a stochastic or *soft* mapping between states in the source and target MDPs. The advantages are three-fold. First, by extending existing work on MDP homomorphisms, we are able to provide theoretical bounds on performance. Second, the softness provides a natural means of approximation in the likely event that the two tasks do not have perfectly matching structure. Finally, the continuous parameters yield polynomial algorithms for finding good mappings.

1.1 Background on MDP Homomorphisms

An MDP is a tuple $\langle S, A, T, R, \gamma \rangle$ consisting of a state set S, action set A, transition function $\{T : S \times A \times S \to [0, 1]\}$, an expected reward function $\{R : S \times A \to \mathbb{R}\}$, and a discount factor $\gamma \in [0, 1]$. A policy is a mapping from each state to an action $\{\pi : S \to A\}$. A Q-function $Q^\pi(s, a)$ is the expected discounted reward obtained by taking action a in state s and following π thereafter. We define the m-step

Cite as: Transfer via Soft Homomorphisms, Jonathan Sorg, Satinder Singh, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 741–748

optimal Q function such that $\forall s \in S, a \in A$:

$$Q_m(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{b \in A} Q_{m-1}(s',b) \quad (1)$$

and define $Q_{-1}(s,a) = 0$. The optimal Q function $Q^*(s,a) = Q_\infty(s,a)$. A given Q-function yields a greedy action in each state according to $\arg\max_a Q(s,a)$. A policy that is greedy with respect to the optimal value function is an optimal policy.

Informally, a homomorphism of a system with transition dynamics is a transformation that preserves some aspects of its dynamics. Ravindran [5] defines an MDP homomorphism from MDP M to MDP M', denoted $h^{M \mapsto M'}$, as a surjection from the state-action pairs in M to the state-action pairs in M' that *commutes* with the dynamics of the MDPs and preserves the reward function. Throughout this paper, we simplify the problem by assuming the action portion of this mapping is the identity function. Henceforth, we will drop the superscript $M \mapsto M'$ from h when it is clear from context.

Definition 1. An MDP *homomorphism* $h^{M \mapsto M'}$ from a target MDP $M = \langle S, A, T, R, \gamma \rangle$ to a source MDP $M' = \langle X, A, T', R', \gamma \rangle$ is a surjection $h^{M \mapsto M'} : S \to X$ such that $\forall s \in S, a \in A, x' \in X$:

$$R'(h(s),a) = R(s,a) \quad (2)$$

$$T'(h(s),a,x') = \sum_{s' : h(s') = x', s' \in S} T(s,a,s'). \quad (3)$$

Given MDPs M, M', and homomorphism h, we can make strong statements relating the two systems. Perhaps most importantly, it can be shown that the values of the optimal policies in equivalent states are equal, and an optimal policy in one system can be easily transformed to produce an optimal policy in the other as follows. For all $(s,a) \in (S \times A)$,

$$Q^*(s,a) = Q^*(h(s),a), \quad (4)$$

where $Q^*(h(s),a)$ is the optimal value of state $h(s)$ in MDP M'.

Note that the direction of the mapping is typically opposite that of transfer. Given a homomorphism from the target MDP to the source MDP, we use it to transfer knowledge from the source MDP back to the target MDP.

Although the homomorphism result is powerful, the preconditions are too restrictive for our purposes. In practice, it is unlikely that an agent will have experience in a system for which there exists a surjection satisfying the constraints. Even when an exact match does exist, the task of searching for a corresponding satisfactory function h is difficult. In fact, a closely-related abstraction problem has been shown to be NP-hard [5].

2. SOFT HOMOMORPHISMS

Our method provides a solution to both these shortcomings by replacing the discrete function $h^{M \mapsto M'}(s)$ with a continuous function $f^{M \mapsto M'} : S \times X \to [0,1]$ such that $\forall s \in S, \sum_{x \in X} f_s^{M \mapsto M'}(x) = 1$. Throughout this paper, we drop the superscript $M \mapsto M'$ from f when it is clear from context.

The function $f_s(x)$ can be loosely interpreted as $P(X|S)$, or the probability that state $s \in S$ maps to state $x \in X$. It has the important property that if we were to restrict $f_s(x)$ to be deterministic, we would restore the semantics of the homomorphism mapping. However, in general the resulting structure can no longer be called a homomorphism as it is no longer a simple surjection. We refer to it throughout as a *soft homomorphism*.

This more general class of soft equivalence relationships can potentially do well in broader settings that may not work well using a discrete mapping. Furthermore, the relaxation from a discretely parameterized function to a continuously parameterized distribution yields polynomial algorithms with guaranteed solutions. This is akin to approximately solving an integer programming problem by relaxing the constraint to integers and converting the problem to a linear program. Because we have relaxed this constraint, we will no longer be guaranteed exact optimal value function equivalence. However, we can bound the transfer error as a function of how "close" the stochastic mapping is to determinism.

2.1 Soft Constraints

The constraints in the MDP homomorphism definition can be extended to handle this more general form of mapping. In short, everywhere $h(s)$ appears in Definition 1, we replace it with an expectation with respect to $f_s(x)$.

Definition 2. A *soft MDP homomorphism* $f^{M \mapsto M'}$ from a target MDP $M = \langle S, A, T, R, \gamma \rangle$ to a source MDP $M' = \langle X, A, T', R', \gamma \rangle$ is a probability distribution $P(X|S)$ denoted $f_S(X)$ such that $\forall s \in S, a \in A, x' \in X$:

$$\sum_{x \in X} R'(x,a) f_s(x) = R(s,a) \quad (5)$$

$$\sum_{x \in X} T'(x,a,x') f_s(x) = \sum_{s' \in S} T(s,a,s') f_{s'}(x'). \quad (6)$$

We can similarly extend the lifted Q-function in equation (4) to the stochastic mapping using expectation. Define the soft-homomorphism lifted Q-function to be $\forall s \in S, a \in A$,

$$\hat{Q}(s,a) \stackrel{\text{def}}{=} \sum_{x \in X} Q^*(x,a) f_s(x). \quad (7)$$

Throughout this paper, we refer to the greedy policy with respect to the lifted Q-function as the *lifted policy*.

Notice that if we were to restrict $f_s(x)$ to be deterministic, equations (5-6) are equivalent to equations (2-3). Clearly, if we were to solve for $f_s(x)$ and find a deterministic solution, the discrete homomorphism result (4) still holds: $\hat{Q}(s,a) = Q^*(s,a)$. In the event that f is stochastic, we cannot guarantee lifted value equality.

2.2 Error Bounds

Although we know that a deterministic soft homomorphism results in a perfect lifted value function, a natural question to ask is whether a small deviation from determinism is catastrophic. The remainder of this section demonstrates that a small deviation from a deterministic f results in a small lifted Q-function error. Lemma 1 bounds from below and Lemma 3 bounds from above, resulting in inequalities of the form

$$Q^*(s,a) \leq \hat{Q}(s,a) \leq Q^*(s,a) + K/(1-\gamma),$$

where K will be defined in Lemma 2.

LEMMA 1. *Let Q^* be the optimal Q-function for MDP $M = \langle S, A, T, R, \gamma \rangle$. Let $\hat{Q}$ be the lifted Q-function obtained from a given $M' = \langle X, A, T', R', \gamma \rangle$ and soft MDP homomorphism function f, then $\forall s \in S, a \in A$,*

$$\hat{Q}(s,a) \geq Q^*(s,a) \tag{8}$$

PROOF. Using induction, first we show the base case:

$$\begin{aligned} Q_0(s,a) &= R(s,a) \\ &= \sum_{x \in X} R'(x,a) f_s(x) \\ &= \sum_{x \in X} Q_0(x,a) f_s(x). \end{aligned}$$

Next we show the induction step.

$$\begin{aligned} &Q_m(s,a) \\ &= R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{b \in A} Q_{m-1}(s',b) \\ &\leq R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{b \in A} \sum_{x' \in X} Q_{m-1}(x',b) f_{s'}(x') \\ &\leq R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \sum_{x' \in X} \max_{b \in A} Q_{m-1}(x',b) f_{s'}(x') \\ &= R(s,a) + \gamma \sum_{x' \in X} \max_{b \in A} Q_{m-1}(x',b) \sum_{s' \in S} T(s,a,s') f_{s'}(x') \\ &= R(s,a) + \gamma \sum_{x' \in X} \max_{b \in A} Q_{m-1}(x',b) \sum_{x \in X} T'(x,a,x') f_s(x) \\ &= \sum_{x \in X} f_s(x) \left[R'(x,a) + \gamma \sum_{x' \in X} T'(x,a,x') \max_{b \in A} Q_{m-1}(x',b) \right] \\ &= \sum_{x \in X} f_s(x) Q_m(x,b) \end{aligned}$$

Taking the limit as $m \to \infty$ gives us the bound stated above. □

The lone source of inequality results from "swapping" the max operation with the summation in the induction step in the proof of Lemma 1 above. We put an upper bound on the error incurred by this "swap" in Lemma 2, and in Lemma 3, we use this result to complete the upper bound for a lifted Q-function.

In order to bound error as a function of deviation of f from determinism, we must first define a measure of deviation from determinism. The measure we use is $(1 - f_s(\tilde{x}_s))$ where $\tilde{x}_s = \arg\max_x f_s(x)$. In words, this is the probability that target state s does not map to its most likely source state. If the mapping is deterministic, this value will be 0.

In what follows we will use the definitions:

$$\begin{aligned} Q_m^-(s) &\stackrel{\text{def}}{=} \min_{x : f_s(x) > 0, a \in A} Q_m(x,a) \\ Q_m^+(s) &\stackrel{\text{def}}{=} \max_{x : f_s(x) > 0, a \in A} Q_m(x,a). \end{aligned}$$

In words, $Q_m^-(s)$ is the minimum value over all actions $a \in A$ and all states $x \in X$ that state s maps to with positive probability. $Q_m^+(s)$ is defined similarly.

LEMMA 2. *Let MDP $M = \langle S, A, T, R, \gamma \rangle$, let MDP $M' = \langle X, A, T', R', \gamma \rangle$, and let $f^{M \mapsto M'}$ be a soft homomorphism. Then $\forall s \in S$,*

$$\max_{a \in A} \sum_{x \in X} Q_m(x,a) f_s(x) \geq \sum_{x \in X} \max_{a \in A} Q_m(x,a) f_s(x) - K_{m,s}$$

where $K_{m,s} = (1 - f_s(\tilde{x}_s)) \left(Q_m^+(s) - Q_m^-(s) \right)$.

PROOF.

$$\begin{aligned} &\sum_{x \in X} \max_{a \in A} Q_m(x,a) f_s(x) - \max_{a \in A} \sum_{x \in X} Q_m(x,a) f_s(x) \\ &= \sum_{x \in X} \max_{a \in A} Q_m(x,a) f_s(x) \\ &\quad - \max_{a \in A} \left(Q_m(\tilde{x}_s,a) f_s(\tilde{x}_s) + \sum_{x \in X : x \neq \tilde{x}_s} Q_m(x,a) f_s(x) \right) \\ &\leq \sum_{x \in X : x \neq \tilde{x}_s} \max_{a \in A} Q_m(x,a) f_s(x) - (1 - f_s(\tilde{x}_s)) Q_m^-(s) \\ &\leq (1 - f_s(\tilde{x}_s))(Q_m^+(s) - Q_m^-(s)) = K_{m,s} \end{aligned}$$

□

In words, $K_{m,s}$ is the maximum difference between the values of the best and worst actions in a source state under an m-step optimal policy times the probability that the given target state s does not map to the most likely source state.

LEMMA 3. *Let Q^* be the optimal Q-function for MDP $M = \langle S, A, T, R, \gamma \rangle$. Let $\hat{Q}$ be the lifted Q-function obtained from a given $M' = \langle X, A, T', R', \gamma \rangle$ and soft MDP homomorphism function f, then $\forall s \in S, a \in A$,*

$$\hat{Q}(s,a) \leq Q^*(s,a) + K/(1-\gamma) \tag{9}$$

where $K = \sup_{s,m} (1 - f_s(\tilde{x}_s)) \left(Q_m^+(s) - Q_m^-(s) \right)$.

PROOF. This is an induction proof showing that $\forall m \geq 0$,

$$Q_m(s,b) \geq \sum_{x \in X} Q_m(x,b) f_s(x) - \sum_{i=1}^{m} \gamma^i K. \tag{10}$$

First we show the base case:

$$\begin{aligned} Q_0(s,a) &= R(s,a) \\ &= \sum_{x \in X} R'(x,a) f_s(x) \\ &= \sum_{x \in X} Q_0(x,a) f_s(x) - \sum_{i=1}^{0} \gamma^i K. \end{aligned}$$

Then we follow with the induction proof.

$$\begin{aligned} Q_m(s,a) &= R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{b \in A} Q_{m-1}(s',b) \\ &= R(s,a) + \sum_{s' \in S} T(s,a,s') \gamma \max_{b \in A} Q_{m-1}(s',b) \end{aligned} \tag{11}$$

Focus on the right hand side of (11):

$$\begin{aligned}
&\gamma \max_{b\in A} Q_{m-1}(s',b) \\
&\quad \geq \gamma \max_{b\in A} \left[\sum_{x'\in X} Q_{m-1}(x',b) f_{s'}(x') - \sum_{i=1}^{m-1} \gamma^i K \right] \\
&\quad = \gamma \max_{b\in A} \left[\sum_{x'\in X} Q_{m-1}(x',b) f_{s'}(x') \right] - \sum_{i=2}^{m} \gamma^i K \\
&\quad \geq \gamma \sum_{x'\in X} \max_{b\in A} \left[Q_{m-1}(x',b) f_{s'}(x') - K \right] - \sum_{i=2}^{m} \gamma^i K \\
&\quad = \gamma \sum_{x'\in X} \max_{b\in A} \left[Q_{m-1}(x',b) f_{s'}(x') \right] - \sum_{i=1}^{m} \gamma^i K. \qquad (12)
\end{aligned}$$

Substituting (12) back into the right hand side of (11) and proceding similarly to Lemma 1 yields the induction invariant (10). Taking the limit as $m \to \infty$ produces the upper bound (9) stated above. □

We combine these results to produce the following theorem.

THEOREM 1. *Let Q^* be the optimal Q-function for MDP $M = \langle S, A, T, R, \gamma \rangle$. Let $\hat{Q}$ be the lifted Q-function obtained from a given $M' = \langle X, A, T', R', \gamma \rangle$ and soft MDP homomorphism function f, then $\forall s \in S, a \in A$,*

$$Q^*(s,a) \leq \hat{Q}(s,a) \leq Q^*(s,a) + K/(1-\gamma) \qquad (13)$$

where $K = \sup_{s,m} (1 - f_s(\tilde{x}_s)) \left(Q_m^+(s) - Q_m^-(s) \right)$.

PROOF. The result follows from Lemmas 1 and 3. □

We know from past work that a bound in error of an approximate value function can be used to bound the loss of the resulting greedy policy [6]. Therefore, given $\gamma < 1$, if the soft mapping is close to deterministic, the value of the greedy policy with respect to the lifted value function can also be shown to be close to the value of the true optimal policy. Additionally, there are several conditions under which a lifted Q-function using a soft homomorphism results in zero error. First, as stated before, a deterministic mapping results in 0 error.

COROLLARY 1. *Let Q^* be the optimal Q-function for MDP $M = \langle S, A, T, R, \gamma \rangle$. Let $\hat{Q}$ be the lifted Q-function obtained from a given $M' = \langle X, A, T', R', \gamma \rangle$ and soft MDP homomorphism function f. If f is deterministic,*

$$\hat{Q}(s,a) = Q^*(s,a). \qquad (14)$$

PROOF. If f is deterministic, $K = 0$ as defined in Theorem 1. Therefore $Q^*(s,a) \leq \hat{Q}(s,a) \leq Q^*(s,a)$. □

Determinism is not the only condition under which the lifted Q-function has 0 error. The major distinction between a deterministic mapping and a stochastic mapping is the potential for a one-to-many relationship from a target state to source states. Errors occur only when a target state maps to source states with different optimal actions. Corollary 2 formalizes this notion. Note that Corollary 1 is a special case of this result.

COROLLARY 2. *Let M be an MDP with state space S and action space A. Let $\hat{Q}$ be the lifted Q-function obtained from MDP M' with state space X and soft MDP homomorphism function f. If $\forall s \in S, x \in X, x' \in X$ such that $f_s(x) > 0, f_s(x') > 0$:*

$$\arg\max_{a\in A}(Q^*(x,a)) = \arg\max_{a\in A}(Q^*(x',a)), \qquad (15)$$

then,

$$\hat{Q}(s,a) = Q^*(s,a).$$

PROOF. (Sketch) If the conditions in (15) hold, then the reordering of the max and summation operations in the proof of Lemma 1 has no effect:

$$\max_{a\in A} \sum_{x\in X} Q_m(x,a) f_s(x) = \sum_{x\in X} \max_{a\in A} Q_m(x,a) f_s(x).$$

We use this fact to remove the lone source of inequality in the proof of Lemma 1. □

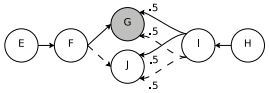

Figure 1: Source MDP M'

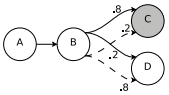

Figure 2: Target MDP M

In practice, we expect many domains to exhibit a form of locality: states that exhibit similar dynamics under the model are likely to have the same optimal action. States that exhibit similar dynamics are likely to be grouped together in a soft-homomorphism. This is one reason we expect a lifted policy using a non-deterministic soft homomorphism to perform well.

Theorem 2 demonstrates the versatility of soft homomorphisms using a small example system.

THEOREM 2. *There exists a source MDP M' and a target MDP M such that a soft homomorphism $f^{M \mapsto M'}$ exists with which the optimal Q-function can be lifted exactly, but no discrete homomorphism exists.*

PROOF. Figures 1 and 2 illustrate two MDPs, M and M'. Each state has up to two actions which we refer to as (S)olid and (D)ashed. All transitions are deterministic except where annotated. A transition to the goal state (shaded) results in a reward of 1. Other transitions result in 0 reward.

In MDP M', there are two paths to the termination states G and J. On one path (F), the agent is confronted with a deterministic choice between the goal state G and the non-goal state J. On the other path (I), both actions have a 50% chance of reaching the goal. In MDP M, there is one path to the termination states. The agent is confronted at state B with a choice between two action with transition probabilities indicated in the figure.

Clearly, the state C can be mapped to state G and similarly, state D maps to J deterministically. State B has dynamics and reward that share properties of both states F and I. The discrete homomorphism cannot handle this

situation, but the soft homomorphism allows $f_B(F) = 0.6$ and $f_B(I) = 0.4$. The strength of both homomorphism variants is that they allow dynamics to *commute*. In this example, this allows state A to be mapped stochastically between states E and H: $f_A(E) = 0.6, f_A(H) = 0.4$. Explicit enumeration of all possible discrete mappings reveals that no mapping is a discrete homomorphism.

Suppose the agent would like to transfer the value function from M' to M. Given a discount factor $\gamma = 1$, the optimal value of state A is $Q^*(A, S) = 0.8$. This matches the lifted value function $\hat{Q}(A, S) = f_A(E)Q^*(E, S)+f_A(H)Q^*(H, S) = 0.6 \times 1 + 0.4 \times 0.5 = 0.8$. This is guaranteed to be the case, because Corollary 2 holds. □

2.3 Action Sequence Equivalence

Although the lifted Q-function is not always guaranteed to match the true optimal value function, we can show equality results for state-independent sequences of actions of arbitrary length. In other words, given a soft homomorphism, the values of open-loop policies can always be lifted exactly. This is what Givan, Dean and Greig [1] refer to as *action sequence equivalence*. We include this result to demonstrate another reason why, even in the face of a very stochastic mapping, we have reason to expect the lifted policy will yield reasonable performance.

Let us define a finite sequence of actions $\xi_0^n = (a_0, a_1, ..., a_n)$. A subsequence is denoted $\xi_i^j = (a_i, a_{i+1}, ..., a_j)$. The ith action is denoted ξ_i. The value of taking the actions in order from action sequence ξ_0^n starting in state s, denoted $Q(s, \xi_0^n)$ is defined recursively as $Q(s, \xi_n^n) = R(s, \xi_n^n)$ and $Q(s, \xi_i^n) = R(s, \xi_i) + \gamma \sum_{s' \in S} T(s, \xi_i, s')Q(s', \xi_{i+1}^n)$.

THEOREM 3. *If $f^{M \mapsto M'}$ is a soft homomorphism from a target MDP $M = \langle S, A, T, R, \gamma \rangle$ to a source MDP $M' = \langle X, A, T', R', \gamma \rangle$, then $\forall s \in S$,*

$$Q(s, \xi_0^n) = \sum_{x \in X} f_s(x) Q(x, \xi_0^n). \tag{16}$$

PROOF. First we prove the base case.

$$\begin{aligned} Q(s, \xi_n^n) &= R(s, \xi_n) \\ &= \sum_{x \in X} R'(x, \xi_n) f_s(x) \\ &= \sum_{x \in X} Q(x, \xi_n^n) f_s(x) \end{aligned}$$

Next we show the induction step.

$$\begin{aligned} Q(s, \xi_i^n) &= R(s, \xi_i) + \gamma \sum_{s' \in S} T(s, \xi_i, s') Q(s', \xi_{i+1}^n) \\ &= R(s, \xi_i) + \gamma \sum_{s' \in S} T(s, \xi_i, s') \sum_{x' \in X} Q(x', \xi_{i+1}^n) f_{s'}(x') \\ &= R(s, \xi_i) + \gamma \sum_{x' \in X} Q(x', \xi_{i+1}^n) \sum_{s' \in S} T(s, \xi_i, s') f_{s'}(x') \\ &= R(s, \xi_i) + \gamma \sum_{x' \in X} Q(x', \xi_{i+1}^n) \sum_{x \in X} T'(x, \xi_i, x') f_s(x) \\ &= \sum_{x \in X} f_s(x) \left[R'(x, \xi_i) + \gamma \sum_{x' \in X} T'(x, \xi_i, x') Q(x', \xi_{i+1}^n) \right] \\ &= \sum_{x \in X} f_s(x) Q(x, \xi_i^n) \end{aligned}$$

□

3. LEARNING ALGORITHMS

Another important advantage of soft homomorphisms is that they are *learnable*. Current state of the art in discrete homomorphism learning involves explicit enumeration and evaluation of all candidate mappings. Past work has made this tractable by considering mappings from a very limited set.

In this section, we present two algorithms with different advantages and requirements. The first algorithm, which uses convex quadratic programming, works in the restrictive case that the agent has access to models of both the source and target tasks. It has the advantage that if a soft homomorphism exists, the algorithm is guaranteed to find one in time polynomial in the sizes of the source and target tasks. The second algorithm removes the requirement that a model of the target task be known a priori. We also introduce a compact parameterization of f that allows transfer to large and continuous state spaces. Both methods work by minimizing sum of squared error of the form

$$\varepsilon = \sum_{s,a,x'} (\varepsilon_R(s, a)^2 + \varepsilon_T(s, a, x')^2) \tag{17}$$

where,

$$\varepsilon_R(s, a) = \sum_{x \in X} R'(x, a) f_s(x) - R(s, a) \tag{18}$$

$$\varepsilon_T(s, a, x') = \sum_{x \in X} T'(x, a, x') f_s(x) - \sum_{s' \in S} T(s, a, s') f_{s'}(x'). \tag{19}$$

If $\varepsilon = 0$, the mapping is a soft homomorphism. In many practical scenarios, there will not exist a mapping such that the constraints can be exactly satisfied. In such cases, $\varepsilon > 0$ and we call the stochastic mapping an *approximate soft homomorphism*.

Equation (17) weighs transition constraints and reward constraints equally. This is not a requirement of either algorithm, but for lack of a better method, we leave it this way for simplicity. In our experiments, rewards are normalized to the range $[0, 1]$ to compensate for disparities in scale; however, there are a greater number of transition constraints than reward constraints.

3.1 Quadratic Programming Solution

If the agent has models of both the source task and the target task, the convex Quadratic Programming (QP) solution developed in this section has excellent theoretical properties. This algorithm begins with the observation that the soft homomorphism constraints (Definition 2) can be expressed as a set of linear constraints of the form $\mathbf{A}y = b$ and $y \geq 0$, where $\mathbf{A}$ is a matrix and y and b are vectors. We would like to satisfy $\mathbf{A}y = b$ in a least-squares sense such that $y \geq 0$. This is the well-understood linear least squares problem with nonnegative variables [4]. Although specialized algorithms exist, it can be solved using a standard quadratic program solver. Methods for solving it are guaranteed to find a global minimum in polynomial time. If $\mathbf{A}$ has full column rank, the solution will be unique.

We begin by defining the model parameters and soft homomorphism function f as sets of matrices. The superscipt

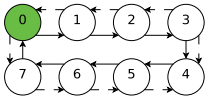

Figure 3: Counter domain, 3 bits

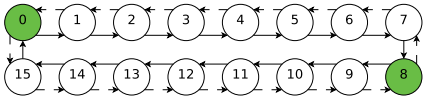

Figure 4: Counter domain, 4 bits

a indicates a unique matrix for each action.

$$\mathbf{F} : |S| \times |X| = f_s(x) \tag{20}$$
$$\mathbf{T}^a : |S| \times |S| = T(s, a, s') \tag{21}$$
$$\mathbf{R}^a : |S| \times 1 = R(s, a) \tag{22}$$
$$\mathbf{T}^a_X : |X| \times |X| = T'(x, a, x') \tag{23}$$
$$\mathbf{R}^a_X : |X| \times 1 = R'(x, a) \tag{24}$$

Given these definitions, here are the soft homomorphism constraints expressed in matrix form:

$$\forall_a \mathbf{F}\mathbf{T}^a_X = \mathbf{T}^a\mathbf{F} \tag{25}$$
$$\forall_a \mathbf{F}\mathbf{R}^a_X = \mathbf{R}^a \tag{26}$$
$$\mathbf{F}\mathbf{1}_x = \mathbf{1}_s \tag{27}$$
$$\mathbf{F}_{ij} \geq 0, \tag{28}$$

where $\mathbf{1}_s$ is an column vector of 1's with length $|S|$.

The vec operator explicitly converts a matrix $\mathbf{B}$ into a column vector $b = \operatorname{vec}\mathbf{B}$ by stacking its columns in order. The Kronecker product of two matrices $\mathbf{A}_{p\times q} \otimes \mathbf{B}_{r\times s}$ results in a matrix $\mathbf{C}_{pr\times qs} = \{a_{ij}\mathbf{B}\}$. The vec operator can be related to the Kronecker product using the identity $\operatorname{vec}(\mathbf{ABC}) = (\mathbf{C}' \otimes \mathbf{A}) \operatorname{vec}\mathbf{B}$. We can use this fact to re-express constraints (25 - 28) as

$$\forall_a \left[{\mathbf{T}^a_X}' \otimes \mathbf{I}_s - \mathbf{I}_x \otimes \mathbf{T}^a\right] \operatorname{vec}\mathbf{F} = \mathbf{0} \tag{29}$$
$$\forall_a \left[{\mathbf{R}^a_X}' \otimes \mathbf{I}_s\right] \operatorname{vec}\mathbf{F} = \operatorname{vec}\mathbf{R}^a \tag{30}$$
$$\left[\mathbf{1}'_x \otimes \mathbf{I}_s\right] \operatorname{vec}\mathbf{F} = \mathbf{1} \tag{31}$$
$$\operatorname{vec}\mathbf{F} \geq 0, \tag{32}$$

where $\mathbf{I}_s$ is the identity matrix of size $|S|$ and A' denotes the matrix transpose of A. To get the form $\mathbf{A}y = b$ as desired, the matrices on the left hand side of $\operatorname{vec}\mathbf{F}$ in equations (29-31) are stacked to form $\mathbf{A}$ and the vectors on the right hand side are similarly stacked to form b.

Quadratic Programming Example

Figures 3 and 4 illustrate a variant of the binary integer counter domain adapted from [12]. This domain was developed to demonstrate an abstraction algorithm, but we use it here to demonstrate transfer. The state is represented by an n bit integer. There are two actions: INCREMENT and DECREMENT. Each action attempts to adjust the integer in the named direction. Each succeeds with probability 0.8 and fails with probability 0.2, resulting in no change. The agent receives reward when the lowest three bits are all 0. The agent observes a unique identifier (not the binary integer representation) from each state. Although this domain is not a challenging RL task, it provides an easy mechanism for producing transfer domains of different sizes. None of the bits greater than 3 are necessary for predicting reward – they can be ignored. Therefore, any task larger than 3 bits can be exactly abstracted to the 3-bit task.

We attempted transfer problems from a source task of size 2^3 to a target of size 2^n. Using a commercially available quadratic program solver, we were able to solve counter domain transfer problems upwards of size 2^{16} in a matter of minutes. The method found the optimal deterministic homomorphism for all solved problems. This was guaranteed to be the case for these problems, because a deterministic solution exists and the computed A matrix had full column rank.

Given two MDPs with state spaces S and X, the size of the solution grows $O(|S| \cdot |X|)$. For example, the 16 bit transfer problem contains 2490368 constraints and $f_s(x)$ requires $|S| \cdot |X| = 2^{19}$ parameters. The QP is able to be solved quickly by taking advantage of the sparse structure of the $\mathbf{A}$ matrix. Clearly, however, for large problems the tabular representation of $f_s(x)$ used here is impractical. Although the online algorithm presented in the next section does not have the same convergence guarantees, it allows for a compact parameterization of the mapping function.

3.2 Approximate Online Algorithm

In practice, it is unrealistic to expect an agent to have a complete model of the target system. The algorithm developed in this section drops this requirement, needing only samples of experience from the world. Also, it allows a compact parameterization to be used to represent f. We illustrate both points by demonstrating the algorithm on a transfer problem from a discrete state source task to a continuous vector-valued target task.

First we need to compactly parameterize $f_s(x)$. There are many potential representations. The algorithm presented below assumes that the representation always produces a valid probability distribution regardless of its parameters. The following example parameterization, based popular Boltzmann distribution, has this property.

$$f_s(x|\Theta) = \frac{e^{\langle\theta_x, \phi(s)\rangle}}{\sum_y e^{\langle\theta_y, \phi(s)\rangle}} \tag{33}$$

Each state x in the source MDP has an associated parameter vector θ_x which together form the parameter matrix Θ. The function $\phi(s)$ extracts an arbitrary feature vector from state s. Angle brackets $\langle , \rangle$ represent a dot product.

Next we adjust the soft constraints in Definition 2 to handle the continuous target MDP. This can be done by simply replacing the sum on the right hand side of (6) with an integral. We then take the gradient of (17) with respect to θ_y. The resulting gradient computation still contains expectations with respect to s. The online algorithm approximates these expectations using samples. Given a state transition (s, a, r, s') (state, action, reward, next state), and a learning rate α, it updates the parameters a small step in the negative

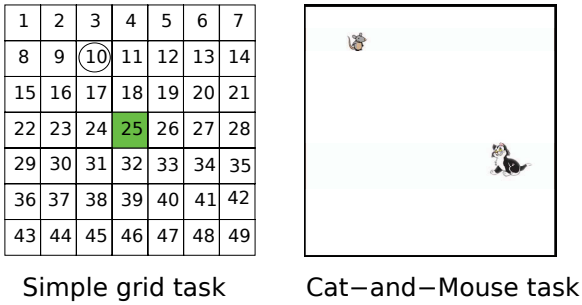

Figure 5: Grid world to Cat-and-Mouse world transfer problem

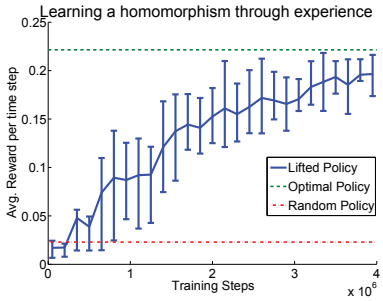

Figure 6: Continuous Transfer Performance

gradient direction (for each x', y) given by $-\Delta\theta_{x',y} =$

$$\alpha\hat{\varepsilon}_T(s,a,s',x')\left(\sum_{x\in X} T_x(x,a,x')\frac{\partial f_s(x)}{\partial\theta_y} + \frac{\partial f_{s'}(x')}{\partial\theta_y}\right) + \alpha\hat{\varepsilon}_R(s,a,r)\left(\sum_{x\in X} R_x(x,a)\frac{\partial f_s(x)}{\partial\theta_y}\right), \quad (34)$$

where $\hat{\varepsilon}_R(s,a)$ and $\hat{\varepsilon}_T(s,a,x')$ are sample approximations of equations (18-19):

$$\hat{\varepsilon}_R(s,a,r) = \sum_{x\in X} R'(x,a)f_s(x) - r \quad (35)$$

$$\hat{\varepsilon}_T(s,a,s',x') = \sum_{x\in X} T'(x,a,x')f_s(x) - f_{s'}(x'). \quad (36)$$

Discrete to Continuous Transfer Example

The gridworld in Figure 5 was chosen to isolate one important skill that is shared in a large number of common tasks: that of navigating to a fixed point. The agent (represented by a circle) starts in a random location. The agent has four actions: N, S, E, and W. Each moves in the corresponding direction and stops against the boundaries. With 0.4 probability, the action fails and the agent moves in a uniformly random direction. When the agent reaches the goal state (shaded) in the center of the grid, it receives 1 reward and the agent resets to a random new start state. The agent receives 0 reward otherwise. The value of γ is 0.95. The agent observes a unique identification integer for each grid location as demonstrated in the diagram.

In the continuous Cat-and-Mouse world in Figure 5, the agent, represented by the cat, must learn how to catch a mouse. The cat is represented by a 1×1 rectangle in bounded (10×10) two dimensional space. The cat has N, S, E, and W actions. Each action moves the cat 1 unit in the corresponding direction plus noise in both dimensions drawn from independent Gaussian random variables with $\mu = 0$ and $\sigma = 0.2$. The mouse is represented by a 0.5×0.5 unit rectangle and takes random cardinal direction steps of length 0.5 plus independent Gaussian noise with the same distribution as the cat's. When either the cat or mouse attempt to move across a boundary, movement is cancelled. When the cat's bounding box overlaps that of the mouse, the agent gets 1 reward and both the cat and mouse reset to new positions. Otherwise, the agent receives 0 reward. The value of γ is 0.95. The cat has a sensor that provides it with a vector observation equal to the (x, y) offset between it and the mouse.

In both the grid world and the Cat-and-Mouse world, the goal for the agent is to navigate to a location in 2D space using actions that move it in the cardinal directions. In the case of Cat-and-Mouse, the goal location in moving. We would like to capture the common structure in such a way that knowledge learned in the simple grid world can be applied to improve the agent's performance in Cat-and-Mouse world. This transfer task is challenging in a number of ways: the continuous state space requires an approximate mapping, the dynamics of a moving goal differ from those of a stationary one, and the observation spaces share no structure. We do not expect that the lifted Q-function will match exactly to the optimal Q-function. However, we will show that the method will produce a lifted Q-function with a similar "shape" as the optimal Q-function, resulting in a greedy policy that is nearly optimal.

In this experiment, the agent is first placed in a 10x10 grid world and is given enough time to learn an approximate model and an approximately optimal value function. The agent is then moved to the Cat-and-Mouse task. The agent starts with a uniformly distributed f parameterized as in equation (33) with $\phi(s) = \langle x, y, 1\rangle$, where x and y are the offset values observed by the agent. To learn the soft homomorphism, the agent takes random actions and observes the outcomes. These outcomes are then used to produce samples of the gradient, equation (34), and the parameters are updated accordingly. Note that although we used a random policy for ease of exposition, this is by no means a requirement. In fact, if the world were sampled according to a distribution more closely representing the stationary distribution under the optimal policy, the gradient descent method will focus on states most relevant to an optimal agent.

To evaluate the quality of f, we periodically froze learning and computed the greedy policy with respect to the lifted value function. Each frozen policy was evaluated by computing the average reward obtained over 1 million evaluation steps. These results are compared against a random policy baseline and an approximately optimal policy. Figure 6 diplays the results of an average of 20 learning curves. The error bars represent the 1st and 3rd quartiles of the data at each evaluation step. Even though these two problems differ in a number of ways, the best lifted value functions produce greedy policies that are close to optimal.

4. DISCUSSION

Figure 6 demonstrates the ability to learn an approximate soft homomorphism from an unfamiliar continuous environment to a known discrete world online using a stream of experience. This knowledge structure was used to transfer knowledge sufficient to produce a nearly optimal policy. There are few methods that can claim to achieve autonomous transfer in this setting.

We make no claims about the data efficiency of the online algorithm. This work largely focused on the theoretical results, and the experiment is meant as a proof of concept. We claim that in the larger context of an agent presented with a number of tasks, the cost of learning a homomorphism is an amortized cost. When confronted with multiple tasks in both the source and target domains, it is likely that one mapping will work well across a number of transfer problems. This is especially true in the case of skill acquisition, which involves multiple subgoals (reward functions) being defined in the same state space.

The guarantees provided by MDP homomorphisms break down when the reward function is changed; however, it was shown in [12] that the reward constraint in MDP homomorphisms may be replaced by a constraint on an arbitrary feature of the MDP. A similar extension may be made to soft homomorphisms, allowing soft homomorphisms to be defined in terms relevant to transition dynamics that are independent of a particular goal. In this way, a soft homomorphism may be made *reusable* by allowing solutions of multiple tasks to be transferred using a single state mapping.

Although this work focused on the transfer problem, it is important to note that the results apply directly to abstraction methods as well. In fact, the precise form of the soft mapping function we use has been used in previous work on soft state aggregation [7]. In that work, $f_s(x)$ was used as a soft partitioning function and it was improved using Bellman error. They were able to show improvement in clustering, but show no optimality results. Our work specifies precise conditions under which a soft partitioning scheme can result in an error-free Q-function. In future work, we plan to develop efficient abstraction algorithms using the results developed here. As we stated in the introduction, the abstraction problem can be seen as an extention to the transfer problem – one in which a simple source domain is chosen and solved on-the-fly.

Past work on MDP homomorphisms and transfer learning has often included an action translation mechanism in addition to the state mapping. Our work does not preclude action abstraction; for example, the action mapping from discrete MDP homomorphisms needs only to be modified slightly to be used in our current results. However, the challenge of efficiently learning an action mapping has not yet been solved. Future work will examine soft action mappings for efficient learning.

This work is similar to the concept of an approximate MDP homomorphism [5], but there is one key difference. An approximate homomorphism allows for an inexact match, but it is still a discrete mapping. The power of soft homomorphisms comes from the fact that the mapping itself is continuously parameterized. Approximate homomorphisms do not yield the efficient algorithms given above, nor do they allow a class of mappings as flexible.

5. ACKNOWLEDGMENTS

This work is supported by the Air Force Office of Scientific Research under grant FA9550-08-1-0418. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the sponsors.

6. REFERENCES

[1] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[2] G. Konidaris and A. G. Barto. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 895–900, 2007.

[3] Y. Liu and P. Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *Procedings of the Twenty-First National Conference on Artificial Intelligence*, pages 415–20, July 2006.

[4] L. F. Portugal, J. J. Judice, and L. N. Vicente. A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables. *Mathematics of Computation*, 63(208):625–643, 1994.

[5] B. Ravindran. *An Algebraic Approach to Abstraction in Reinforcement Learning*. PhD dissertation, University of Massachusetts, Amherst MA, 2004.

[6] S. Singh and R. Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

[7] S. P. Singh, T. Jaakkola, and M. I. Jordan. Reinforcement learning with soft state aggregation. In *Advances in Neural Information Processing Systems*, volume 7, pages 361–368, 1995.

[8] V. Soni and S. Singh. Using homomorphisms to transfer options across reinforcement learning domains. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, 2006.

[9] M. E. Taylor, G. Kuhlmann, and P. Stone. Autonomous transfer for reinforcement learning. In *Procedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2008.

[10] M. E. Taylor, P. Stone, and Y. Liu. Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 880–885, July 2005.

[11] M. E. Taylor, S. Whiteson, and P. Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–8, May 2007.

[12] A. P. Wolfe and A. G. Barto. Decision tree methods for finding reusable mdp homomorphisms. In *Procedings of the Twenty-First National Conference on Artificial Intelligence*, pages 530–535, 2006.