A Framework for Transfer Learning in Sequential Decision-Making

Kimberly Ferguson

University of Massachusetts Amherst
Department of Computer Science

September 21, 2011

Research on transfer in reinforcement learning has used various terminology dependent on each author, with little commonality. The term transfer learning has been used to describe numerous different kinds of situations where any type of information is reused. The terms knowledge transfer and behavior transfer have been used interchangeably, and the claim of cross-domain transfer has shown little consistency. We propose a unified framework with a specific taxonomy for transfer learning in sequential decision-making, such that the terminology in future work can be clear and specific. We will describe four different *types of information transfer* which are dependent upon the kind of information being transferred: knowledge transfer (k-transfer), value transfer (v-transfer), behavior transfer (b-transfer), and representation transfer (r-transfer). We will also describe different *transfer situations* which are dependent on the difference between the source and target of transfer: task transfer, and domain transfer. Each type of information transfer can be combined with each transfer situation. We continue to use *transfer learning* as an umbrella term for this subset of machine learning.

For a concrete example, imagine a graduate student, or other autonomous agent, exploring a building such as a computer science department. There are different *types of information* that the agent can learn in this building which would be useful to reuse in a new building or new situation. The agent can learn a concept—a door links a room to another location; reuse of this information is *knowledge transfer*. The agent can learn a specific value—a door is eight feet tall; reuse of this information is *value transfer*. The agent can learn a skill—how to move to the door from any location in a room; reuse of this information is *behavior transfer*. The agent can learn a representation—doors, hallways, and rooms are useful features with which to represent a building; reuse of this information is *representation transfer*.

We can also reason about different *transfer situations*. This will encompass

both *task transfer* and *domain transfer*, for each of which we can specify specific subtypes. Again, consider the agent learning information for reuse in a building. The agent may originally learn the information with the intention of exiting the building, but can reuse the information if the goal changes to finding a certain room or person; the reuse of information in this situation is *goal-task transfer.* The same information can be reused to exit the building even if the student is now blindfolded and there is a non-zero probability that he will move a different direction than intended; the reuse of information in this situation is *dynamics-task transfer.* Information can also be reused if the student is given a scooter, and now is able to perform different actions than when just walking; the reuse of information in this situation is *action-task transfer.*

If the agent reuses the information in a different domain (i.e. a different building), then he is performing domain transfer. If the agent moves from a small square classroom to a large square auditorium, the new building is a different size but the same shape as the original; the reuse of information in this situation of transfer is *scaling-domain transfer.* If the agent moves from the computer science building to the campus dorms, the new building is a different shape, but the same size as the original; the reuse of information in this situation of transfer is *topological-domain transfer.* If there are several identical classrooms in the building, then the agent can reuse the information learned from one classroom for all the other classrooms; the reuse of information in this situation is *hierarchical-domain transfer.* If the agent moves from a building to a submarine or an amusement park, the new situation is a different type of domain (a different type of enclosure or not an enclosure at all); the reuse of information in this situation of transfer is *cross-domain transfer.*

1 Types of Information Transfer

The types of information transfer described below encompass many of the types of information that can be successfully used for transfer learning. This taxonomy directly speaks to one of the three important aspect of transfer learning: *what to transfer*. While these categories are sufficient for our purposes, it is evident that knowledge transfer may be divided into subtypes, which may be useful in the long run.

1.1 Knowledge Transfer (k-transfer)

Knowledge transfer (k-transfer) refers to when an agent begins by collecting a knowledge base of information and then reuses this information, either directly or with modifications, in a novel situation. Knowledge transfer combines observation and reasoning, encompassing broad conceptual knowledge. It can include negations, and multi-level rules. Examples of k-transfer include the transfer of facts, rules, expert advice, features, analogies, and relational information. For instance, relational knowledge about a factory manager and factory workers might be transferred to the novel but analogous situation of an advisor and

graduate students. As we have defined it, k-transfer is an extremely large category, and we acknowledge that it will likely require being broken into smaller subcategories.

1.2 Value Transfer (v-transfer)

Value transfer (v-transfer) refers to when specific scalar values are learned in one situation and then reused, either directly or with modifications, in a novel situation. Examples of v-transfer include the transfer of value functions, rewards, weights, parameters, scores, and probabilities. V-transfer, which pertains to the reuse of specific numerical values, can be viewed as a more specific type of knowledge transfer that is based on numerical observations only, with no high-level reasoning.

1.3 Behavior Transfer (b-transfer)

Behavior transfer (b-transfer) refers to when an agent collects a toolkit of one or more behaviors and then transfers these behaviors, either directly or with modifications, to a new situation, object, or agent. This information is based on experience and interactions, not high-level reasoning or knowledge. Within sequential decision-making, b-transfer encompasses decisions of which the agent itself has control. Examples include the transfer of actions, skills, options, policies, and samples.

1.4 Representation Transfer (r-transfer)

Representation transfer (r-transfer) refers to when the underlying representation of the state space is learned and then reused, either directly or with modifications, in a novel situation. This information is what is necessary to represent the structure of the environment. Examples of r-transfer include the transfer of categories (for supervised learning), basis functions, embeddings, or other representations of state, action, or state-action space. R-transfer relates to how we collect, organize, and represent basic information.

For an example from nature, consider the studies that have shown that there is a specific stage when children become able to correctly represent and generalize shapes. An experimental example is when researchers have young children try to fit blocks of different shapes into the correctly shaped slots. This deals with representations and not knowledge because the fact that this shape is a rectangle and that shape is a triangle is not the point. The purpose is to know that *Slot 1* represents the same shape as *Block 2* and then use that representation information to generalize that *Block 2* fits into *Slot 1*.

With further investigation, we believe it will become evident that certain types of information are more usefully transferred in specific kinds of situations. For example, it is likely that lower-level transfer such as representation transfer will be explicitly more useful for simpler tasks, such as task transfer and scaling-domain transfer, but higher-level transfer such as knowledge transfer will

be necessary for more complex situations like cross-domain transfer. Since all remain open problems in A.I., we will concentrate on the fundamental problem of representation transfer. We will explore representation transfer in a variety of transfer situations.

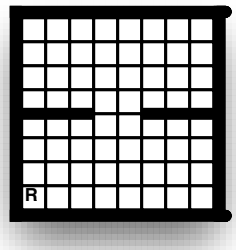
2 Transfer Situations

Our taxonomy of transfer situations, namely *task transfer* and *domain transfer*, described below and first discussed in (Ferguson & Mahadevan, 2006), can be linked to the ideas of *agent-space* and *problem-space* respectively, as presented in (Konidaris, 2006). Each *type of information transfer* discussed above can be used for each *transfer situation* described below. By categorizing the types of situations in which transfer learning can successfully take place, this taxonomy begins to address one of the three important aspects of transfer learning: *when to transfer*. The measuring of the similarity between transfer situations is another important aspect of deciding when to transfer and is still an open problem. In order to further specify how r-transfer occurs in each transfer situation, we assume that basis functions are the type of information being transferred. Specifically, using PVFs in (Ferguson & Mahadevan, 2006) as our basis function of choice, we will detail the construction of the basis function Φ_{Target} for the target domain, from the source basis functions Φ_{Source}.

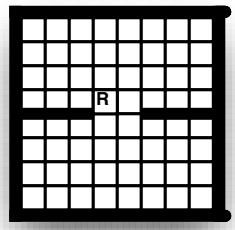
In the field of psychology, six levels of transfer learning have been defined based on the degree of similarity between the situations (Haskell, 2000). Level 1, nonspecific transfer, and Level 2, application transfer, are trivial within A.I., simply referring to the use of knowledge. Level 3, context transfer, refers to applying what one has learned in a slightly different situation, which we relate to task transfer. Level 4, near transfer, refers to when previous knowledge is transferred to new situations that are closely similar but not identical to previous situations, which we relate to domain transfer. Level 5, far transfer, refers to applying learning to situations that are quite dissimilar to the original learning situation, which we related to the harder types of cross-domain transfer. Level 6, creative transfer, refers to transferring learning in such a way that a newly discovered similarity creates a new concept, which is beyond what we describe for A.I. systems, but is the ultimate goal.

2.1 Task Transfer

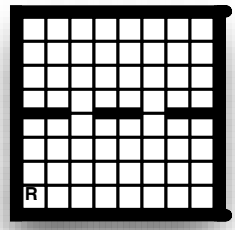
When an agent reuses information learned in one situation for a new situation with the same structure, then task transfer is being performed. There are several ways this can occur. The forms of task transfer we describe below include when the task is changed by modifying the reward function or the dynamics of the situation, or changing the action set. Other situations in which information is reused but the state space does not change are also task transfer. This includes situations such as when the agent's start state changes, which is often trivially performed at the start of each exploration episode.



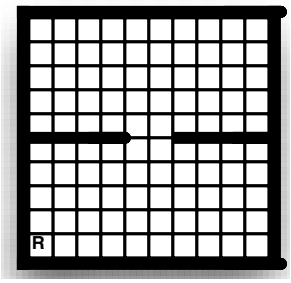
(a) 8x8 task and domain transfer source.



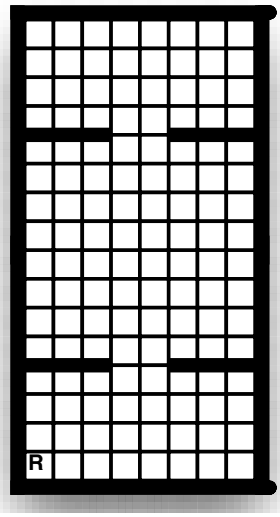
(b) 8x8 goal-task transfer target.



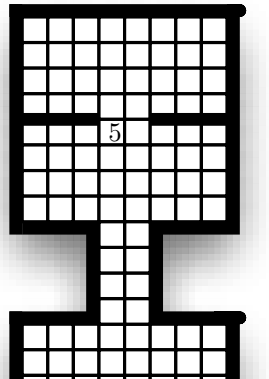
(c) 8x8 topological-domain transfer target.



(d) 10x10 scaling-domain transfer target.



(e) One possible hierarchical-domain transfer target.



For a direct example from nature of task transfer situations, we take examples from a neuroscience study on primates, which showed that a rhesus monkey, Ivan, was capable of using tools for multiple goals (Veino & Novak, 2003). Ivan learns to use a rake to retrieve a treat in different locations (goal-task transfer) and to perform the same task with different tools (action-task transfer). Additionally, it has been shown that he can maneuver the rake over various barriers of his cage and even go around a corner to retrieve the treat (topological-task transfer).

2.1.1 Goal-Task Transfer

Goal-task transfer refers to when the MDP of the source $M_{Source} = \langle S, A, P, \mathbf{R_{Source}} \rangle$ and the MDP of the target $M_{Target} = \langle S, A, P, \mathbf{R_{Target}} \rangle$ differ only in their reward functions. For an example of goal-task transfer, consider the discrete two-room gridworld domain shown in Figure 1, the reward might be moved from the corner (Figure 1(a)) to the doorway (Figure 1(b)), or the reward function might be changed from 0 at the goal state and -1 in all other states to 10 at the goal state and 0 at all other states. Likewise, consider the continuous control task of the mountain car where the goal is to reach the top of the hill by oscillating to build enough momentum. The elevation of the top of the hill (the goal) might be raised or lowered, the goal might be moved to the other side of the hill, or given a different numerical value.

2.1.2 Dynamics-Task Transfer

Dynamics-task transfer refers to when the source MDP $M_{Source} = \langle S, A, \mathbf{P_{Source}}, R \rangle$ and the target MDP $M_{Target} = \langle S, A, \mathbf{P_{Target}}, R \rangle$ differ only in their transition functions. This will occur when only the dynamics of the situation are changed. For example, in the discrete two-room gridworld domain, the source task may be deterministic, where the action an agent selects is always successful, but the target task may be 10%-stochastic, where 10% of the time the action will fail, resulting in the agent transitioning (uniformly at random) according to one of the actions. In the extreme case, a transition $P(s'|s, a)$ may be zeroed out, for instance if a new wall has been added between two states in the gridworld. While the set of states S remains the same, the dynamics have changed, which is why this extreme case is considered topological-domain transfer and is discussed in Section 2.2.2.

For the goal-task or dynamics-task r-transfer problems, when reward-independent basis functions such as PVFs are used, the basis functions can be transferred directly without modification. In this case, the graph Laplacian $\mathcal{L} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$ of the source graph G_{Source} and target graph G_{Target} are the same, since only the reward function or model dynamics have changed, and their adjacency matrix A and valency (or degree matrix) D is the same for both graphs. Thus the k eigenvectors $\Phi_{Source} = \phi_{i=1...k}(\mathcal{L}(G_{Source}))$ can be reused directly as Φ_{Target}. For these types of task transfer, the states and actions of both domains are the same, so the domain mapping is trivial. This transfer is referred to as

literal transfer in psychology literature, since intact information transfers to a new task. We will examine these cases of goal-task and dynamics-task r-transfer in our preliminary experiments in Section 4.2 & 4.3, respectively.

2.1.3 Action-Task Transfer

Action-task transfer refers to when the source MDP $M_{Source} = \langle S, \mathbf{A_{Source}}, P_{Source}, R_{Source} \rangle$ and the target MDP $M_{Target} = \langle S, \mathbf{A_{Target}}, P_{Target}, R_{Target} \rangle$ differ in their action sets, which will also change P and R as a byproduct, since each are based on a current state and action. This will occur when the action set is modified by adding, subtracting, or changing actions. For example, in the discrete two-room gridworld domain, the source task may only include the directional actions North, South, East, and West, but the target task may include a don't move action, or a pick-up action.

For action-task r-transfer, the change in the action set can cause a difference in the adjacency matrix of the source A_{Source} and target A_{Target}, commonly constructed based on when an action connects two states. The domain mapping of states is trivial, but an action mapping is also needed. Keeping with our gridworld example, this may be as simple as mapping: {West $\rightarrow$ left, East $\rightarrow$ right, North $\rightarrow$ up, South $\rightarrow$ down}, but may be more complicated should the number of actions differ. This case will not be part of our focus.

2.2 Domain Transfer

Task transfer differs from all the variations of domain transfer because the state space in the source and target always remains the same. When the state space changes, an agent must reuse information learned in one environment to perform some task in a new environment, and domain transfer is being performed. In practice, task and domain transfer can be combined, but in our descriptions we will consider them independently. We define four different types of domain transfer below.

2.2.1 Scaling-Domain Transfer

Scaling-domain transfer refers to when the source MDP $M_{Source} = \langle \mathbf{S_{Source}}, A, P_{Source}, R_{Source} \rangle$ and the target MDP $M_{Target} = \langle \mathbf{S_{Target}}, A, P_{Target}, R_{Target} \rangle$ differ in their state set such that $|S_{Source}| \neq |S_{Target}|$, which will also change P and R as a byproduct since each are based on a current state and action. In particular, the state space in the target domain is just a scaled version of the source domain. A classic RL example for scaling-domain transfer would be transferring information from an 8×8 gridworld (Figure 1(a)) to a 10×10 (Figure 1(d)) or larger gridworld. The aim is that if an agent has explored a small office with four walls and a door, much of that situation can be transferred to a new larger lecture hall. This specific transfer situation has been of particular interest to many machine learning researchers who would like to be able to train more quickly in the smaller source domain and transfer that information to the larger target

domain with a reduction in training time. A popular example of scaling-domain transfer in RL is in the robot-soccer domain where an agent might transfer from 3 vs. 2 keep-away to 5 vs. 3 keep-away. Even transfer of information to a smaller domain can lead to interesting results. For a real-world example consider the olympic sport of volleyball. Indoor volleyball has 6 players on each team and is the version that many of us were taught to play. However, beach volleyball only has 2 players on each team. When trying to modify your play from indoor volleyball to beach volleyball the transfer is not trivial, but clearly information can be reused.

The scaling-domain r-transfer problem, focuses on the expansion of the connectivity of the graph G_{Source}, where the pattern of the adjacency graph in the source A_{Source} is retained in target A_{Target}, while the sizes of the matrices differ. For this case, we can use various interpolation techniques to extend the basis functions to the novel states. For PVFs specifically, we extend the k eigenfunctions $\Phi_{Source} = \phi_{i=1...k}(\mathcal{L}(G_{Source}))$ to each new state $\{\dot{s} \in S_{new} \subset S_{Target} | \dot{s} \in S_{Target}, \dot{s} \notin S_{Source}\}$ to create Φ_{Target}. The domain mapping in this case is not obvious. Results using a trivial mapping are shown in Section 4.4. Future work will include further investigation of manifold alignment techniques that can be used to align and stretch the eigenfunctions of the source to perform scaling-domain r-transfer.

2.2.2 Topological-Domain Transfer

Topological-domain transfer refers to when the shape of the state space in the target domain is a modified version of the shape of the state space in the source domain. This is similar to topological-task transfer, where the MDP of the source $M_{Source} = \langle S, A, \mathbf{P_{Source}}, R \rangle$ and the MDP of the target $M_{Target} = \langle S, A, \mathbf{P_{Target}}, R \rangle$ differ only in their transition functions, but in topological-domain transfer the state space changes. Topological-domain transfer includes transferring information between differently shaped mazes when the set of states itself is not changed, only the transitions between them. If the basic goal is to find a way to the end of the maze, then the fact that the mazes are shaped differently should not mean the agent has to start learning everything from scratch. A simpler example would be an obstacle gridworld where the location, shape or size of the obstacle differs from source (Figure 1(a)) to target (Figure 1(c)) . Only a portion of the situation is different and the rest is unchanged. Most of the information learned in the source domain can be reused directly, and methods can be developed to identify and modify the rest. Matrix perturbation theory (Stewart & Sun, 1990), which has already been applied to MDPs (Cao, 2003), is one area that may provide ideas for such methods. A more extreme example of topological-domain transfer is transferring from a 10x10 one-room gridworld, to a 2x50 grid hallway. The set of states is the same, they have just been rearranged, which greatly changes their transition probabilities. Quantifiable limits for how modified a domain can be while still being similar enough to avoid negative transfer is related to the open problem of *when to transfer*.

For the topological-domain r-transfer problem, the connectivity of the graph G_{Source} is different from that of G_{Target} and the adjacency matrix of the target A_{Target} is the adjacency matrix of the source A_{Source}, perturbed by some matrix E, i.e. $A_{Target} = A_{Source} + E$. Thus, we can view the differences in the corresponding Laplacians of the source and target, $\mathcal{L}(G_{Source})$ and $\mathcal{L}(G_{Target})$ as:

$$\mathcal{L}(G_{Source}) = D_{Source}^{-\frac{1}{2}}(D_{Source} - A_{Source})D_{Source}^{-\frac{1}{2}}$$
$$\mathcal{L}(G_{Target}) = D_{Target}^{-\frac{1}{2}}(D_{Source} - [A_{Source} + E])D_{Target}^{-\frac{1}{2}}$$

where D_{Source} is the valency matrix of G_{Source} and D_{Target} is the valency matrix of G_{Target}.

Matrix perturbation theory may provide ways to quantify how the eigenvalues and eigenvectors $\Phi_{Target} = \phi_{i...k}(\mathcal{L}(G_{Target}))$ change based on the perturbation E (e.g., changes in the connectivity of the graph). The theory is that for quantifiably small enough perturbations, a trivial domain mapping can be used without modification with no detriment to learning. For larger perturbations, the domain mapping can help determine which basis functions need modification based on which mapped states are different. Beyond that, a measure of when a state space is perturbed enough to cause negative transfer is necessary.

2.2.3 Hierarchical-Domain Transfer

Hierarchical-domain transfer refers to when the state space of the source domain is a subset of the larger target domain, which is repeated one or more times. The MDP of the source $M_{Source} = \langle \mathbf{S_{Source}}, A, P_{Source}, R_{Source} \rangle$ and the MDP of the target $M_{Target} = \langle \mathbf{S_{Target}}, A, P_{Target}, R_{Target} \rangle$ differ in their state set such that $S_{Source} \subset S_{Target}$, which will also change P and R as a byproduct since each are based on a current state and action. The section of the target that is identical to the source can be reused directly, so that only the information from unique sections of the target need to be learned from scratch. This transfer is hierarchical, and can be extended to reusing information learned in a section of the target domain to another part of that domain, if the agent can recognize that the section is repeated elsewhere. How an agent would autonomously recognize that a section of a domain is similar to something it has already learned is another possible application of representation discovery. For a simple example of hierarchical-domain transfer, consider a 10×10 gridworld as the source with the target domain a 20×20 4-room world (which is made up of four 10×10 rooms with connecting doors). More examples are displayed in Figure 1, where Figure 1(a) is the source and Figures 1(e) & 1(f) are possible targets. For a more real-world example, consider an agent whose target domain is a computer science building. One subset of the building that is repeated often is professors' offices. If the agent initially learns information from an office as its source domain, then much of this information can be reused for each office in the building and should not need to be repeatedly learned.

In the hierarchical-domain r-transfer problem, the state space of the source is a portion of the state space of the target. The graph G_{Source} and the adjacency matrix A_{Source} of the source is a portion repeated one or more times within the graph G_{Target} and adjacency matrix A_{Target} of the target. The domain mapping from the source to the corresponding portion of the target domain is trivial. Thus, the basis functions only need be calculated on the novel sections of the target domain, saving time. Future work will include further investigation into techniques that can be used for hierarchical-domain r-transfer.

2.2.4 Cross-Domain Transfer

Cross-domain transfer may be viewed as the most advanced and difficult of the transfer situations, and therefore the most interesting and hardest to define. Cross-domain transfer refers to the case when in addition to the state space, the actual domain of the source and target is different. One example of cross-domain transfer in sequential decision-making would be transferring information from the mountain car domain to the inverted pendulum domain. A stereotypical real-world example of cross-domain transfer might be transferring information and strategies learned from playing a chess game to planning a course of action for army infantry groups. A key factor in successful cross-domain transfer is figuring out when two situations are similar, which is an open problem. It is likely that if two domains are radically different, higher-level information transfer, such as k-transfer, will be more useful than lower-level representation transfer. Yet, a difference in representation may be the main part of the definition of when two domains are different enough to be considered as cross-domain transfer. In this case, a change of representation will need to be part of the transfer process.

For example, preliminary work by (Taylor & Stone, 2007) shows that transfer of information from the source can help to enhance the new representation in the target domain, resulting in faster learning. The authors learn a new representation by transferring value functions learned while training one basis to learn a new basis for the same problem. These initial results could be viewed as a simplified version of task transfer, where the state space remains the same, but so does the reward function and the world dynamics. Essentially, all elements of the MDP stay the same and only the representation is changed. While the authors call this representation transfer, since the source and target problems are the same in every way, we will simply refer to it as a change of representation. This technique is promising in terms of transferring information between various agents that would need to use different representations. Another application of this change of representation would be in a cross-domain transfer situation. Since the domains are different, a new representation might be necessary for the target domain. Nonetheless, relevant information can still be transferred from the target to the source domain.

3 Domain Mappings

Most transfer techniques require some kind of mapping between the states in the source and target domain. This may be a mapping between a set of corresponding states, an action mapping, or a complete homomorphism, to name a few. Many domain mappings that have appeared in the literature have been hand-coded mappings provided a priori. In our proposed work we will focus on domain mappings that are learned by the agent. Classic work on analogous learning may also provide additional insights to domain mapping techniques.

4 R-Transfer Experiments in Discrete MDPs

These experiments perform representation transfer—the transfer of a basis. We use proto-value functions as the basis, which are automatically constructed. For evaluation purposes we compare experiments where transfer is not used to those where it is to show that results do not decline. For r-transfer specifically, we do not expect the success of experiments to increase, instead it is assumed that time and space is saved since the basis functions are not recomputed. The goal is to show that representations can be transferred without the learning results being ill-effected.

4.1 Algorithmic Details

Algorithmic details for representation transfer learning in discrete MDPs are given in Figure 2. This can be done using any basis. PVFs are a basis on the function of the state space which are automatically constructed by the agent based upon what it experiences during exploration. All of our experiments use PVFs as the information transferred, the construction of which is given in Figure 3. We use the term *pure* when the PVFs are created from and used for learning on the same (source) graph, while *transfer* will refer to the case in which the PVFs are created on a (source) graph and transferred to be used for learning on another (target) graph. LSPI (Lagoudakis & Parr, 2003) is used to learn the control policy, where the underlying subspace for approximating the value function is spanned by the learned PVFs. The reinforcement learning domain that we will be experimenting on is the obstacle gridworld—in particular, a two-room gridworld (see Figure 4). Sometimes we will use the gridworld with no obstacles, also known as a one-room gridworld. The possible actions are North, South, East, and West. If an agent tries to move into a wall, it will remain in the same state. To estimate the value function, we collect samples using a random walk of a maximum of 200 episodes, each with a maximum of 150 steps and random start state. Non-goal states have zero reward; the goal has reward of 100. This is a discrete domain with 144 states.

`R-transfer Policy Iteration` ($J_{Source}, J_{Target}, M_{Source}, M_{Target}, \epsilon, k$):
// J_{Source}: Number of initial random walk trials for source domain sampling
// J_{Target}: Number of initial random walk trials for target domain sampling
// M_{Source}: Maximum length of each trial for the source domain sampling
// M_{Target}: Maximum length of each trial for target domain sampling
// ϵ: Convergence condition for policy iteration
// k: Number of basis functions to use

Sample Collection Phase (Source and Target)

1. Perform a random walk of J_{Source} trials, each of maximum M_{Source} steps on the source domain, and store the samples in the dataset $\mathcal{D}_{Source}$.
2. Perform a random walk of J_{Target} trials, each of maximum M_{Target} steps on the target domain, and store the samples in the dataset $\mathcal{D}_{Target}$.

Representation Learning Phase (Source)

3. *Create basis functions for the source domain.* The basis function matrix Φ_{Source}, is a $N \times k$ matrix, where N is the number of states in the source domain. The state action bases $\phi(s,a)$ where $s,a \in \mathcal{D}_{Source}$ can be generated from rows of this matrix by duplicating the state bases $\phi(s)|A|$ times, and setting all the elements of this vector to 0 except for the ones corresponding to the chosen action. (See figure 3 for PVF construction.)

Representation Transfer Phase (Source to Target)

4. *Transfer bases from source to target domain.* Transfer Φ_{Target}=Φ_{Source} either directly or with modification. For each new state $\{\dot{s} \in S_{new} \subset \mathcal{D}_{Target} | \dot{s} \in \mathcal{D}_{Target}, \dot{s} \notin \mathcal{D}_{Source}\}$, the row vector $\phi(\dot{s},a)$ of Φ_{Target} is approximated using a pre-selected interpolation method (not necessary for task transfer).

Control Learning Phase (Target)

5. Using a standard parameter estimation method (e.g. Q-learning or LSPI), find an ϵ-optimal policy π for the target domain that maximizes the action value function $\hat{Q}^{\pi} = \Phi w^{\pi}$ within the linear span of the bases Φ_{Target} using the samples in $\mathcal{D}_{Target}$.
6. Return $\hat{Q}^{\pi}$ as the approximation to the optimal value function for the target domain.

Figure 2: Pseudo-code of the r-transfer policy iteration (RTPI) algorithm which incorporates the transferring of basis functions into representation policy iteration.

`Construction of proto-value functions in discrete RL domain:`

1. Build an undirected weighted graph G from collected samples $\mathcal{D}$ where edges are inserted between a pair of points x_i and x_j if there is an action that causes a direct transition from x_i to x_j and all edges have weight 1. Construct the normalized Laplacian $\mathcal{L}$ on graph G as in Equation **??**.
2. Compute the k smoothest eigenvectors of $\mathcal{L}$, and collect them as columns of the basis function matrix Φ, a $N \times k$ matrix, where N is the number of states in the graph.

Figure 3: PVFs as example of basis function construction to be used in Step 2 of the r-transfer algorithm given in Figure 2.

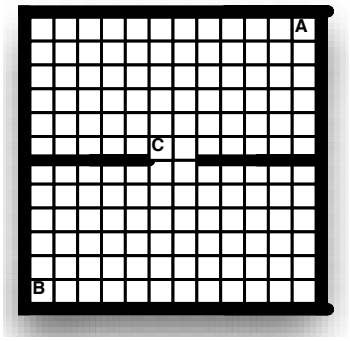
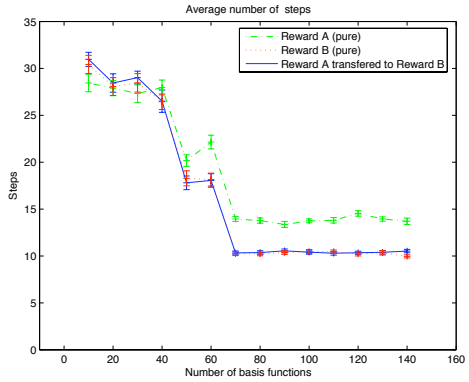


Figure 4: Goal-Task Transfer Example: 12x12 two-room gridworld with various reward functions.

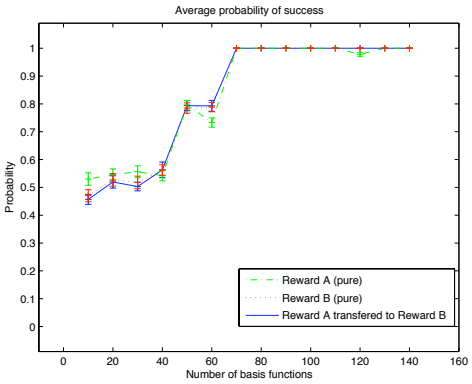
4.2 Goal-Task R-Transfer Results

These goal-task r-transfer experiments investigate transfer learning using PVFs, where the state space and basis functions are constant, but the reward function is varied. Since this method creates basis functions based on the actual topology of the state space, it is a natural solution to this task r-transfer problem. The domain mapping is trivial and the basis functions transfer directly.

In Table 1 we see that transferring the basis functions from one grid (Exp 1.a, with reward in the upper right-hand corner) to grids with different reward functions (Exp 1.b, with reward in the lower left-hand corner and Exp 1.c, with reward in the middle) does not affect performance. Thus we have shown that proto-value functions are reward independent. We use a discount of 0.9, 130 PVFs, and allow 20 iterations. We collect samples using a random walk of a maximum of 200 episodes, each with a maximum of 150 steps and random start state. During testing, the learned policy is evaluated allowing a maximum of 50 steps, and averaged over 20 runs. The reward function assigns zero reward to all states except for the goal which has reward of 10. The results in Figures 5 are for r-transfer experiments set up the same as above except for the following:



(a) Reward Experiment: Number of Steps to goal.



(b) Reward Experiment: Percentage of trials that reach the goal.

Figure 5: Goal-task r-transfer experiments on 12x12 two-room gridworld.

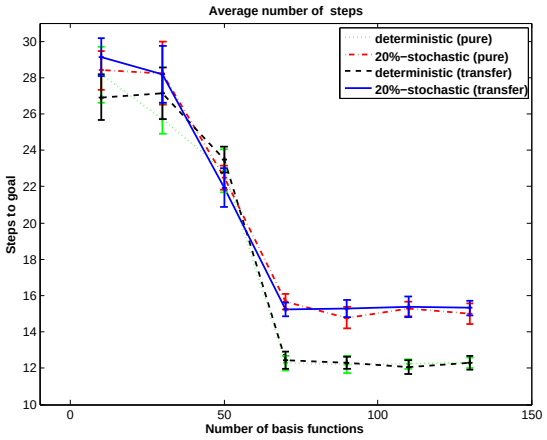
Table 1: Results comparison between experiments in which the eigenfunctions for the current grid are learned and used (pure) versus when the learned eigenfunctions for Exp 1.A are used in grids with different goal location (transfer).

	Exp 1.A (pure)	Exp 1.B (pure)	Exp 1.B (transfer)	Exp 1.C (pure)	Exp 1.C (transfer)
Prob. of success	100%	100%	100%	100%	100%
Avg # of steps	14.8 ± 2.1	13.6 ± 2.1	14.9 ± 3.0	7.3 ± 1.2	7.4 ± 1.2
Min/Max steps	$[5, 27]$	$[4, 22]$	$[5, 24]$	$[3, 13]$	$[2, 11]$
Avg discounted reward	26.2 ± 5.6	30.0 ± 7.1	29.2 ± 8.8	53.5 ± 6.5	53.1 ± 7.3
Iterations to converge	19	16	11	12	12

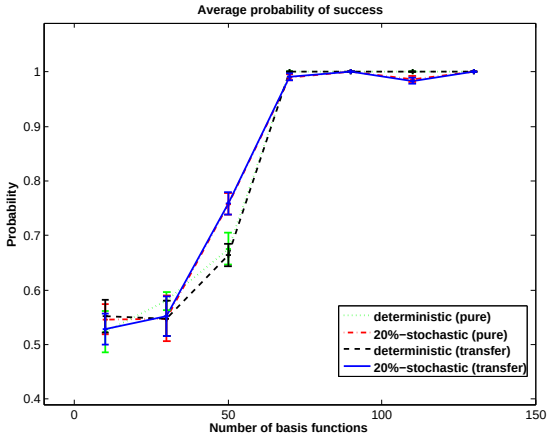
we use a discount of 0.8, and vary the number of eigenvectors to show how many PVFs are needed for good accuracy. Transferring the basis functions learned from a grid with reward in position A to a grid with the reward in position B retains 100% probability of success. We see that the transferred-task performance matches, but is no better than the best of the pure experiments. Results are similar with other reward functions.

4.3 Dynamics-Task R-Transfer Results

These dynamics-task r-transfer experiments investigate transfer learning using PVFs, where the state space and basis functions are constant, but the dynamics of the environment is varied. The four directional actions in the gridworld can be either deterministic or stochastic. If the dynamics are deterministic, an action is successful 100% of the time. When the dynamics are nondeterministic, an action is successful with probability β%, which we refer to as $(\beta - 1)$%-stochastic, and a failed action results in the agent transitioning (uniformly at random) according to one of the other three actions. In these dynamic-task r-transfer experiments, we modify the percentage of stochasticity. Since this method creates basis functions based on the actual topology of the state space, it is a natural solution to this task r-transfer problem. The domain mapping is trivial and the basis functions transfer directly. The results in Figures 6 are for transferring the representation of two 12x12 two-room gridworlds with different dynamics. These experiments use a discount of 0.8, and allow 20 iterations. We collect samples using a random walk of a maximum of 200 episodes, each with a maximum of 150 steps and random start state. During testing, the learned policy is evaluated allowing a maximum of 50 steps, and averaged over 10 runs. The reward function assigns -1 reward to all states except for the goal (reward position A) which has reward of 0. Transferring the basis functions learned from a deterministic grid to a 20%-stochastic grid retains 100% probability of success when using a sufficient number of PVFs, and vice versa. We see that the transferred-task performance matches, but is no better than the best of the pure experiments.



(a) Dynamics Experiment: Number of Steps to goal.



(b) Dynamics Experiment: Percentage of trials that reach the goal.

Figure 6: Dynamics-task r-transfer experiments on 12x12 two-room gridworld.

4.4 Scaling-Domain R-Transfer Results

These scaling-domain r-transfer experiments investigate r-transfer using PVFs, where the size of the state space changes and the the basis functions must be modified (interpolated) to span a larger state space. For simplicity, the reward function and state dynamics will remain constant. This is an important type of transfer learning since the dynamics of a one-room gridworld with no obstacles are the same regardless of scale; the basic topology is a square, the actions are North, East, South, West, and the agent cannot walk through the walls on the border. An agent should be able to transfer the representation it has learned from one gridworld to another. We will focus on the harder case: transferring from a smaller domain to a larger domain. In these experiments, the domain mapping is as it would be for hierarchical-domain transfer—the smaller source graph is trivially mapped by aligning the first state (lower, left corner) in both domains. The basis functions for these corresponding states transfer directly and interpolation is used to estimate the PVFs for the extra states in the target domain. In these experiments, the Nyström extension is used to transfer the PVFs from a smaller one-room gridworld to a larger one-room gridworld. This method interpolates PVFs for new states in the larger domain by basically averaging the nearby known PVFs taken from the smaller domain. Even though when considering probability of success, extending the basis functions to larger state spaces using the Nyström method is working well (100% for larger magnifications), the learned value functions show that the interpolation may not be correct (see Figure 7). The transfer works well as long as the reward is not in the area being extended. It is likely that the Nyström's method of averaging is too simple for this problem and another method may be a better fit.

References

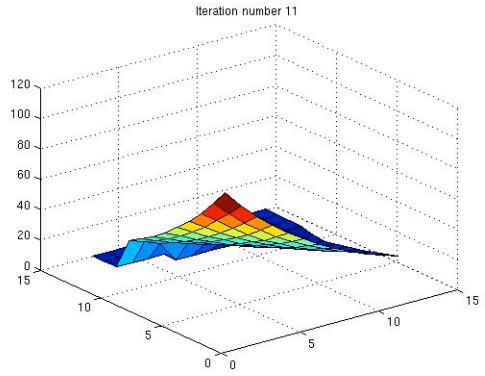
Cao, X. (2003). From perturbation analysis to markov decision processes and reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications* (pp. 9–39). The Netherlands: Kluwer Academic Publishers.

Ferguson, K., & Mahadevan, S. (2006). Proto-transfer learning in markov decision processes using spectral methods. *ICML Workshop on Structural Knowledge Transfer for Machine Learning*.

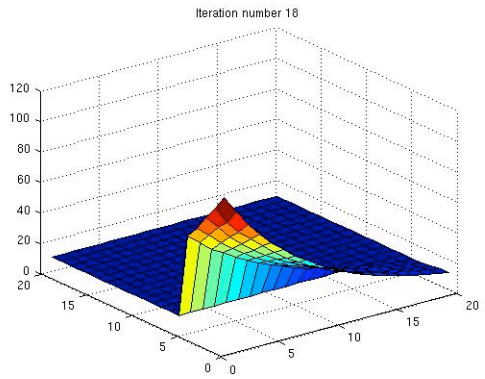
Haskell, R. E. (2000). *Transfer of learning: Cognition, instruction, and reasoning*. Academic Press.

Konidaris, G. D. (2006). A framework for transfer in reinforcement learning. *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*.

Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4*, 1107–1149.



(a) Learned value function when PVFs are transferred directly from a 10x10 one-room gridworld to a 12x12 one-room gridworld.



(b) Learned value function when PVFs are transferred directly from a 10x10 one-room gridworld to a 20x20 one-room gridworld.

Figure 7: Scaling-Domain R-Transfer Experiments: Learned value functions where r-transfer uses the Nyström extension to interpolate PVFs. Reward position A.

Stewart, G. W., & Sun, J. (1990). *Matrix perturbation theory.* Academic Press.

Taylor, M. E., & Stone, P. (2007). Towards reinforcement learning representation transfer. *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 683–685).

Veino, C., & Novak, M. (2003). *Tool use in juvenile rhesus macaques (macaca mulatta)* (Technical Report). University of Massachusetts, Department of Neurocience.