Proto-transfer Learning in Markov Decision Processes Using Spectral Methods

Kimberly Ferguson   KFERGUSO@CS.UMASS.EDU
Sridhar Mahadevan   MAHADEVA@CS.UMASS.EDU
University of Massachusetts Amherst, Computer Science Department, 140 Governor's Drive, Amherst, MA 01002

Abstract

In this paper we introduce proto-transfer leaning, a new framework for transfer learning. We explore solutions to transfer learning within reinforcement learning through the use of spectral methods. Proto-value functions (PVFs) are basis functions computed from a spectral analysis of random walks on the state space graph. They naturally lead to the ability to transfer knowledge and representation between related tasks or domains. We investigate task transfer by using the same PVFs in Markov decision processes (MDPs) with different rewards functions. Additionally, our experiments in domain transfer explore applying the Nyström method for interpolation of PVFs between MDPs of different sizes.

1. Problem Statement

The aim of transfer learning is to reuse behavior by using the knowledge learned about one domain or task to accelerate learning in a related domain or task. In this paper we explore solutions to transfer learning within reinforcement learning (Sutton & Barto, 1998) through spectral methods. The new framework of proto-transfer learning transfers representations from one domain to another. This transfer entails the reuse of eigenvectors learned from one graph on another. We explore how to transfer knowledge learned on the source graph to a similar graph by modifying the eigenvectors of the Laplacian of the source domain to be reused for the target domain. Proto-value functions (PVFs) are a natural abstraction since they condense a domain by automatically learning an embedding of the

state space based on its topology (Mahadevan, 2005). PVFs lead to the ability to transfer knowledge about domains and tasks, since they are constructed without taking reward into account.

We define *task transfer* as the problem of transferring knowledge when the state space remains the same and only the reward differs. For task transfer, task-independent basis functions, such as PVFs, can be reused from one task to the next without modification. *Domain transfer* refers to the more challenging problem of the state space changing. This change in state space can be a change in topology (i.e. obstacles moving to different locations) or a change in scale (i.e. a smaller or larger domain of the same shape). For domain transfer, the basis functions may need to be modified to reflect the changes in the state space. (Foster & Dayan, 2002) study the task transfer problem by applying unsupervised, mixture model, learning methods to a collection of optimal value functions of different tasks in order to decompose and extract the underlying structure. In this paper, we investigate task transfer in discrete domains by reusing PVFs in MDPs with different reward functions. For domain transfer, we apply the Nyström extension for interpolation of PVFs between MDPs of different sizes (Mahadevan et al., 2006). Previous work has accelerated learning when transferring behaviors between tasks and domains (Taylor et al., 2005), but we transfer representation and reuse knowledge to learn comparably on a new task or domain.

2. Framework

Markov Decision Process.
A Markov decision process (MDP) $M = \langle S, A, P_{ss'}^a, R_{ss'}^a \rangle$ is defined by a set of states $S \subset \mathbb{R}^{|S|}$, a set of discrete actions A, a transition model $P_{ss'}^a$ specifying the distribution over future states s' when an action a is performed in state s, and a corresponding reward model $R_{ss'}^a$ specifying a scalar cost or

reward. The state-action value function $Q^\pi(s, a)$ of any policy π can be found for all state-action pairs by solving the linear system of the Bellman equations:

$$Q^\pi(s, a) = \sum_{s' \in S} P^a_{ss'} [R^a_{ss'} + \gamma \sum_{a' \in A} \pi(a', s') Q^\pi(s', a')]. \quad (1)$$

Proto-value Functions.
Proto-value functions (PVFs) are an orthonormal basis spanning all value functions on a state space. PVFs are constructed as follows: 1) from an initial random walk, create an adjacency matrix which reflects the topology of the state space; 2) compute the graph Laplacian of the adjacency matrix; 3) use the smoothest k eigenvectors (ranked by eigenvalue) of this graph Laplacian as PVFs. Thus, PVFs are a bases which respect the topology of the state space (See Figure 1).

More formally, let $G = (V, E, W)$ denote a weighted undirected graph with vertices V, edge set E and weights w_{ij} on edge $(i, j) \in E$. The degree of a vertex v is denoted as d_v. The adjacency matrix A can be viewed as a binary weight matrix describing the connectivity of the graph. Let D be the valency matrix–a diagonal matrix whose entries are the row sums of A. The nomalized Laplacian $\mathcal{L}$ of the graph G is defined as $\mathcal{L} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$. The states are the vertices, and edges connect states that are adjacent in the state space (i.e. a state that can be reached from that state); specifically,

$$\mathcal{L}(u, v) = \begin{cases} 1 - \frac{1}{d_v} & \text{if } u = v \text{ and } d_v \neq 0 \\ -\frac{1}{\sqrt{d_u d_v}} & \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$\mathcal{L}$ is a symmetric self-adjoint operator, and its spectrum (eigenvalues) lie in the interval $\lambda \in [0, 2]$. PVFs are the eigenvectors $\phi_i(\mathcal{L})$, such that $\mathcal{L}\phi_i = \lambda_i \phi_i$.

Case 1: Task Transfer.
For the task transfer problem, the graph Laplacian $\mathcal{L}$ of source graph G_S and target graph G_T are the same, since only the reward function has changed, and their adjacency matrix A is the same. Thus the eigenvectors $\phi_i(\mathcal{L})$ of G_S can be directly transferred to G_T (see Section 3.1).

Case 2: Domain Transfer (topology).
For the domain transfer problem, where the shape of the state space changes, the connectivity of the graph G_S is different from that of G_T and the adjacency matrix of the target A_T is the adjacency matrix of the source A_S perturbed by some matrix E, i.e. $A_T = A_S + E$. Thus, we can view the differences in the corresponding Laplacians of the source and target, $\mathcal{L}_S$

(a) 1st PVF   (b) 2nd PVF
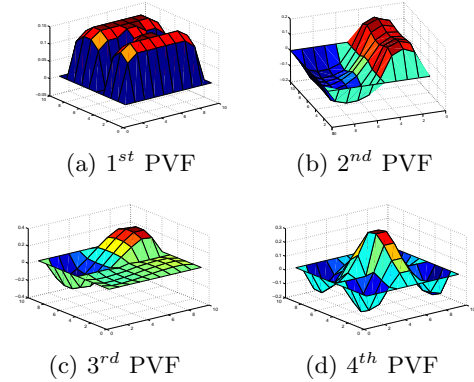
(c) 3rd PVF   (d) 4th PVF

Figure 1. Example PVFs of the 9x9 two-room gridworld in Figure 2(a) with Reward 1. Notice how the PVFs capture the structure inherent to the state space.

and $\mathcal{L}_T$ as:

$$\mathcal{L}_S = D_S^{-\frac{1}{2}}(D_S - A_S)D_S^{-\frac{1}{2}}$$
$$\mathcal{L}_T = D_T^{-\frac{1}{2}}(D_T - [A_S + E])D_T^{-\frac{1}{2}}$$

We are currently exploring matrix perturbation theory to quantify how the eigenvalues and eigenvectors $\phi_i(\mathcal{L}_T)$ change based on the perturbation E (i.e. changes in the connectivity of the graph) (Stewart & Sun, 1990). An example of topological domain transfer is shown in Figure 2, where Figure 2(a) is the source domain and Figure 2(b) is the target domain.
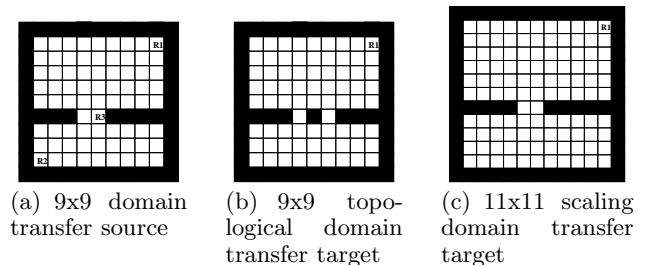

(a) 9x9 domain transfer source

(b) 9x9 topological domain transfer target

(c) 11x11 scaling domain transfer target

Figure 2. Two-room gridworld examples of topological and scaling domain transfer.

Case 3: Domain Transfer (scale).
The domain transfer problem where the size of the state space changes, focuses on the expansion of the connectivity of the graph G_S, where the pattern of the adjacency graph A_S is retained in A_T while the sizes of the matrices differ. We use the Nyström method to extend the eigenfunctions $\phi(\mathcal{L}_S)$ computed on A_S to the new states of A_T to create $\phi(\mathcal{L}_T)$ (see Section 3.2). An example of scaling domain transfer is shown in Figure 2, where Figure 2(a) is the source domain and Figure 2(c) is the target.

Nyström Method.
The Nyström method interpolates the value of eigenvectors computed on known sample states to novel states, and is an application of a classical method used in the numerical solution of integral equations (Baker, 1977). We use a nearest neighbor distance metric (euclidean for gridworlds) to determine which states are close to another, while respecting the topology of the state space. The Nyström method is applied to the approximation of the eigenfunctions of the graph Laplacian where x is a new vertex in the graph and $\phi_i(y)$ are the eigenvectors of a known state (vertex) which is close to x:

$$\phi_i(x) = \frac{1}{1 - \lambda_i} \sum_{y \sim x} \frac{w(x, y)}{\sqrt{d(x)d(y)}} \phi_i(y), \qquad (3)$$

where $d(z) = \sum_{y \sim z} w(z, y)$, and $w(z, y)$ measures how close z is to y.

3. Algorithmic Details and Experimental Results

We use *source* and *target* to describe the domain we transfer knowledge from and to, respectively. We include 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. Least-squares Policy Iteration (LSPI) (Lagoudakis & Parr, 2003) is used to learn the control policy (although other methods, such as temporal-difference, can be used), where the underlying subspace for approximating the value function is spanned by the learned PVFs. The algorithmic details are provided in Figure 3.

3.1. Task Transfer

These 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 transfer problem. These 12x12 two-room gridworlds have zero reward for non-goal states; the goal has reward of 100. We use the 'lsqfast' algorithm in LSPI, a discount of 0.9, 130 eigenvectors, 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. The learned policy is evaluated allowing a maximum of 50 steps, and

---

Proto-transfer $(dom_S, dom_T, S_S, S_T, J, N, \epsilon, k, P)$:

1. **Representation Learning Phase**: Perform a random walk of J trials, each of maximum N steps on the source domain dom_S, and store the states visited in the dataset $\mathcal{D}_S$.

   (a) *Create PVFs for the source domain*: Build an undirected weighted graph G from $\mathcal{D}$ where edges can be inserted between a pair of points x_i and x_j if , x_j is among the k nearest neighbors of x_i and all edges have weight 1. Construct the normalized Laplacian $\mathcal{L}$ on graph G as in Equation 2.

   (b) Compute the k smoothest eigenvectors of $\mathcal{L}$ on the graph G, and collect them as columns of the basis function matrix Φ, a $S_S \times k$ matrix, where S_S is the number of states in the source. The embedding of a state action pair $\phi(s, a)$ where $s \in \mathcal{D}$ is given as $e_a \otimes \phi(s)$, where e_a is the unit vector corresponding to action a, $\phi(s)$ is the sth row of Φ, and $\otimes$ is the tensor product.

2. **Control Learning Phase**: Perform a random walk of J trials, each of maximum N steps on the target domain dom_T, and store the states visited in the dataset $\mathcal{D}_t$. Initialize $w^0 \in \mathcal{R}^k$ to a random vector.
   Repeat the following steps:

   (a) *Transfer PVFs from source to target domain*: Set $i \leftarrow i + 1$. For each transition $(s_t, a_t, s'_t, a'_t, r_t) \in \mathcal{D}_T$, compute low rank approximations of matrix A and b as follows:

   $$\tilde{A}^{t+1} = \tilde{A}^t + \phi(s_t, a_t)(\phi(s_t, a_t) - \gamma\phi(s'_t, a'_t))^T$$
   $$\tilde{b}^{t+1} = \tilde{b}^t + \phi(s_t, a_t)r_t$$

   where $\phi(s_t, a_t)$ is approximated using the Nyström extension (Equation 3) when $s_t \notin \mathcal{D}_S$ (necessary for domain transfer only).

   (b) Solve the system $\tilde{A}w^i = \tilde{b}$

3. **until** $\|w^i - w^{i+1}\|^2 \leq \epsilon$.

4. Return $\hat{Q}^\pi = \sum_i w^i \Phi$ as the approximation to the optimal value function.

Figure 3. Pseudo-code of the proto-transfer algorithm for both task and domain transfer.

---

averaged over 20 runs. Transferring the PVFs learned from a grid with reward in the upper right-hand corner to grids with different rewards (reward in the lower left-hand corner and with reward in the middle) retains high probability of success without increasing the average number of steps to the goal, and keeping the number of iterations to convergence low (See Table 1).

Table 1. Two-room gridworld results comparison between experiments in which the PVFs for the current grid are learned and used (pure) versus when the learned PVFs for Reward 1 placement are used in grids with different rewards (transfer).

	Rew 1 (pure)	Rew 2 (pure)	Rew 2 (transfer)	Rew 3 (pure)	Rew 3 (transfer)
Prob. of success	98%	94%	98%	97%	98%
Ave. number of steps	9.87 ± 1.4226	11.77 ± 2.1485	14.9 ± 2.9551	6.94 ± 1.58	6.52 ± 1.304
Ave. total d. reward	25.873 ± 5.11	26.894 ± 5.751	23.496 ± 4.654	40.888 ± 4.691	39.522 ± 3.998
Iterations to converge	9	7	11	8	11

Table 2. Scaling domain transfer results for experiments in which the PVFs for the 10x10 grid (pure) are learned and used in grids with different sized state spaces (transfer): 4 nearest neighbors, 100 PVF, and Reward 1 (lower left)

	10x10 (pure)	11x11 (transfer)	12x12 (transfer)	15x15 (transfer)	20x20 (transfer)
Prob. of success	100%	91.6%	94.2%	96%	100%

3.2. Domain Transfer (scale)

These experiments investigate transfer learning using PVFs, where the reward function is constant and the basis functions are interpolated to span a larger state space. This is an important type of transfer learning since the dynamics of a gridworld with no obstacles are the same regardless of scale; the basic topology is a square. An agent should be able to transfer the representation it has learned in one gridworld to another. The Nyström extension is performed during this domain transfer (see Figure 3).

The PVFs are learned on a 10x10 one-room gridworld and interpolated using the Nyström extension to be transfered to larger domains, up to a 20x20 gridworld. Other parameters are identical to the task transfer experiment. Table 2 shows that extending the basis functions to larger state spaces using the Nyström method works well (100% for larger magnifications). The results are consistent as long as the reward is not in or adjacent to the state space region being extended. For more complicated domains, such as two-room grids we are exploring manifold alignment techniques to allow appropriate mappings from the source to target domain (Ham et al., 2005).

4. Conclusions and Future Work

We have introduced a new framework for transfer learning called proto-transfer. The contribution of this paper is in using spectral methods to successfully transfer representation between domains with different reward functions and different state spaces. Using spectral methods allows reward-independent learning which naturally leads to task transfer. This method works well because PVFs reflect the topology of the state which is barely modified (if at all) by a change in the reward function. However, when the state space is scaled up in domain transfer, the PVFs must be extended using the Nyström method to estimate the PVFs of new states based on that of near-by known states.

Future work includes further experiments with domain transfer, including using matrix perturbation theory to explore the case in which the shape of the domain changes and manifold alignment to advance the results for scaling domain transfer. We are also working on task and domain transfer in continuous domains, where the dynamics of the domain may change.

References

Baker, C. T. H. (1977). *The numerical treatment of integral equations.* Academic Press.

Foster, D., & Dayan, P. (2002). Structure in the space of value functions. *Journal of Machine Learning Research, 49,* 325–346.

Ham, J., Lee, D., & Saul, L. (2005). Semisupervised alignment of manifolds. *Proceedings of the 10th International Workshop onArtificial Intelligence and Statistics.*

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

Mahadevan, S. (2005). Proto-value functions: Developmental reinforcement learning. *Proceedings of the 22nd International Conference on Machine Learning.*

Mahadevan, S., Maggioni, M., Ferguson, K., & Osentoski, S. (2006). Learning representation and control in continuous markov decision processes. *Proceedings of the 21st National Conference on Artificial Intelligence.*

Stewart, G., & Sun, J. (1990). *Matrix perturbation theory.* Oxford: Clarendon Press.

Sutton, R., & Barto, A. G. (1998). *An introduction to reinforcement learning.* MIT Press.

Taylor, M., Stone, P., & Liu, Y. (2005). Value functions for rl-based behavior transfer: A comparative study. *Proceedings of the 20th National Conference on Artificial Intelligence.*