# Proto-transfer Learning in Markov Decision Processes Using Spectral Methods

Keywords: transfer learning, reinforcement learning, spectral learning, proto-value functions, manifolds

## 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. The new framework of proto-transfer learning transfers representation from one domain to another. This transfer entails the reuse of eigenvectors learned from one graph in 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.

In this paper we explore solutions to transfer learning within reinforcement learning (Sutton & Barto, 1998) through spectral methods. (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. 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.

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) $M = \langle S, A, P^a_{ss'}, R^a_{ss'} \rangle$ is defined by a set of states $S \subset \mathbb{R}^d$, a set of discrete actions $A$, a transition model $P^a_{ss'}$ specifying the distribution over future states $s'$ when an action $a$ is performed in state $s$, and a corresponding reward model $R^a_{ss'}$ specifying a scalar cost or reward. The state-action value function $Q^\pi(s,a)$ of any policy $\pi$ can be found for all state-action pairs by solving the linear system of the Bellman equations:

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

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

*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 3).

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,

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

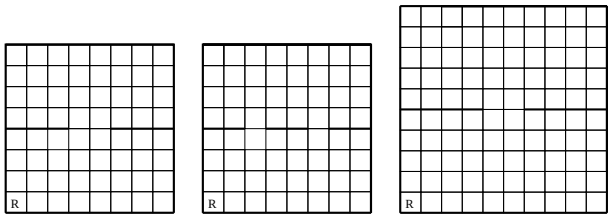
$\mathcal{L}$ is a symmetric self-adjoint operator, and its spectrum (eigenvalues) lie in the interval $\lambda \in [0, 2]$. PVFs are the eigenvectors $\phi_i(\mathcal{L})$, such that $\mathcal{L}\phi_i = \lambda_i \phi_i$.

*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, $\mathcal{L}_S$ and $\mathcal{L}_T$ as:

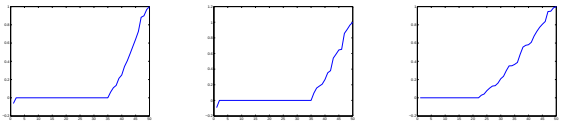
$$\mathcal{L}_S = D_S^{-\frac{1}{2}}(D_S - A_S)D_S^{-\frac{1}{2}}$$
$$\mathcal{L}_T = D_T^{-\frac{1}{2}}(D_T - [A_S + E])D_T^{-\frac{1}{2}}$$

We are currently exploring matrix perturbation theory to quantify how the eigenvalues and eigenvectors $\phi_i(\mathcal{L}_T)$ change (Figure 2) 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 1, where Figure 1(a) is the source domain and Figure 1(b) is the target domain.



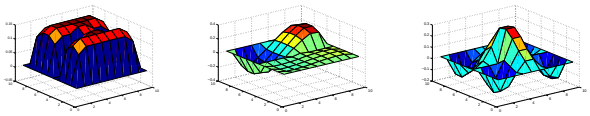
(a) 8x8 domain transfer source. (b) 8x8 topological domain transfer target. (c) 10x10 scaling domain transfer target.

*Figure 1.* Two-room gridworld examples of topological and scaling domain transfer.



(a) Spectrum of 8x8 domain transfer source. (b) Spectrum of 8x8 domain transfer target. (c) Spectrum of 10x10 domain transfer target.

*Figure 2.* Spectrums (eigenvalues) of the two-room gridworld examples of topological and scaling domain transfer.



(a) $1^{st}$ PVF (b) $2^{nd}$ PVF (c) $3^{rd}$ PVF

*Figure 3.* Example PVFs of the 10x10 two-room gridworld (Figure 1(c)). Notice how the PVFs capture the structure inherent to the state space.

*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 1, where Figure 1(a) is the source domain and Figure 1(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 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, where the underlying subspace for approximating the value function is spanned by the learned PVFs. The algorithmic details are provided in Figure 4.

### 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 one-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 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 100% probability of success. (Results not shown.)

`Proto-transfer` $(dom_S, dom_T, S_{\{S,T\}}, J, N, \epsilon, k, P)$:

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 $\Phi$, 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 $s^{th}$ row of $\Phi$, 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 4.* Pseudo-code of the proto-transfer learning algorithm for both task and domain transfer learning.

### 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 rep-

*Table 1.* 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 in state 1 (lower left)

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

resentation it has learned in one gridworld to another. The Nyström extension is performed during this domain transfer (see Figure 4).

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 1 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 area being extended.

## 4. Conclusions

We have introduced a new framework for transfer learning called proto-transfer learning. 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 which estimates the PVFs of new states based on that of near-by known states. The contribution of this paper is in using spectral methods to successfully transfer representation between domains with different reward functions and different state spaces.

## 5. Future Work

Future work includes further experiments with domain transfer, including using matrix perturbation theory to explore the casein which the shape of the domain changes. We are examining homomorphisms between graphs to further formalize proto-transfer learning. We are also working on task and domain transfer in continuous domains, where the dynamics of the domain may change (i.e. the mass or length of the inverted pendulum).

We have shown that using PVFs, an agent can transfer knowledge learned in a smaller gridworld to larger gridworlds. One exciting application for this transfer is representation learning. Can an agent learn which type of domain it is in? If we have different domains as sources for transfer (i.e. a one or two room gridworld, a torroid, a chain, etc. or more abstract domains) the agent can attempt proto-transfer from each of these domains, and the matching domain should have a high probability of success. Therefore, which ever source domain's interpolated PVFs yield the best results on our new domain, we can conclude this new domain is of that the same type. This can give us a reusable representation of different domains, which can save an agent time from relearning representations and knowledge about domains it has already visited. For example, imagine a robot exploring a school. Once it has learned the representations of an office, a classroom, and a closet, it can recognize a new room and reuse the knowledge it already has about that sub-domain.

## 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.

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* $22^{nd}$ *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* $21^{st}$ *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* $20^{th}$ *National Conference on Artificial Intelligence*.