A Laplacian Framework for Option Discovery in Reinforcement Learning

Marlos C. Machado, Marc G. Bellemare, Michael Bowling

Presented by: Ameet Deshpande (CS15B001)

Indian Institute of Technology Madras

Table of contents

Introduction

Motivation

- Proto-value functions (PVFs) provide good representations for states in an MDP
- These PVFs can be used to implicitly define options by introducing intrinsic reward functions called *eigenpurposes* [2]
- These reward functions are task independent and hence result in useful options
- Two (out of the many) characteristics that are important for options
  - They should operate at different time scales
  - They can be easily sequenced

Proto-Value Functions

Brief Introduction

- PVFs are representations which capture the temporal (and structural) aspects of an environment
- They are obtained by diagonalizing a *diffusion model*, which captures the structure of the underlying graph in some sense (assumption, MDP $\approx$ graph)
- Common definitions include, the *combinatorial graph Laplacian matrix*, $L = D - A$ and the *normalized graph Laplacian*, $L = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$, where A is the adjacency matrix and D is a diagonal matrix with D_{ii}=rowsum(A_i)
- It is important to convince yourself that L indeed captures the information that the graph provides
- Before getting into this work, let's look into two other works on Proto-Value Functions so that it is easier to appreciate this framework [3] [4]

PVFs as basis

Recall the Bellman Optimality equation

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'}(R^a_{ss'} + \gamma V^*(s'))$$

- Value iteration and policy iteration represent the value functions using an orthonormal euclidean basis
- That is, the representation for state s_1 is $[1, 0, \ldots, 0]$ and for s_n is $[0, 0, \ldots, 1]$
- This is however a useless basis for any representation learning because the coefficient for each basis is the value function of that state itself

$$V^\pi = \phi^e_1 v^\pi(s_1) + \cdots + \phi^e_n v^\pi(s_n)$$

PVFs as basis

However, PVFs provide a basis $[V_1^G, V_2^G, \ldots, V_n^G]$ such that any value function on that MDP can be represented as a linear combination. Plausible claim right? Any basis should do the trick.

$$V^\pi = \alpha_1 V_1^G + \cdots + \alpha_n V_n^G$$

And as promised before, the basis is task independent

How are they constructed?

Consider the transition probability matrix P^π. If it can be diagonalized, it can be written as

$$P^\pi = \Phi^\pi \Lambda^\pi (\Phi^\pi)^T$$

$$P^\pi = \sum_{i=1}^{n} \lambda_i^\pi \phi_i^\pi (\phi_i^\pi)^T$$

The same Eigenvalues can be used for powers of P^π

Since the basis span the space of all $\mathbb{R}^n$ vectors, the reward can be written as

$$R^\pi = \Phi^\pi \alpha^\pi$$

Using the above, the value function can be written as,

$$V^\pi = \sum_{k=1}^{n} \frac{1}{1 - \gamma \lambda_k^\pi} \phi_k^\pi \alpha_k^\pi$$

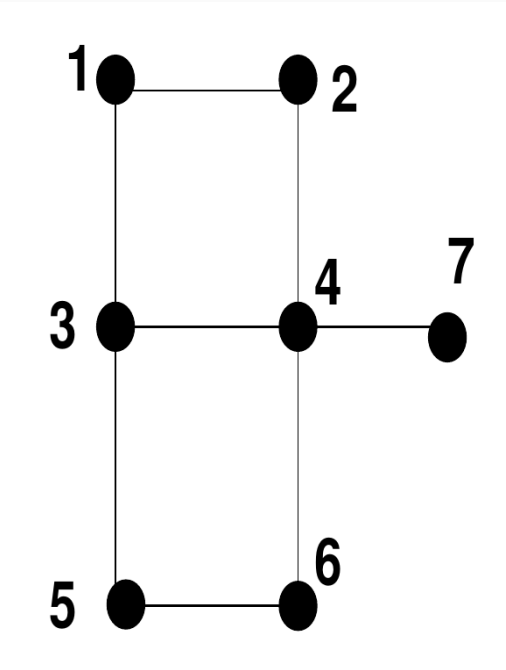
How are they constructed?

$$V^{\pi} = \sum_{k=1}^{n} \frac{1}{1 - \gamma \lambda_k^{\pi}} \phi_k^{\pi} \alpha_k^{\pi}$$

- But the idea was to make it independent of the rewards, so we choose the basis functions which have a large value of $\frac{1}{1-\gamma\lambda_k^{\pi}}$
- And this means we need to choose those Eigenvectors which have high Eigenvalues
- Pick $m < n$ number of functions for the best approximation, with **largest Eigen values** (stochastic matrix properties)

How are they constructed?

- There is still a problem though, the Eigen values depend on the transition probabilities, which are pretty hard to estimate. Why? It is equivalent to finding a part of the underlying model
- We define the combinatorial Laplacian as $L = D - A$



$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 3 & -1 & -1 & 0 & 0 \\ & & & \cdots & & & \end{bmatrix}$$

What is the Laplacian?

- The Laplacian has many interesting properties. It is related to the random walk operator, which is in turn related to the transition probability matrix P^{π}
- It is symmetric and positive semi-definite $\implies$ Positive and real Eigen Values
- It is also connected to spectral clustering
- Note that a function f refers to a mapping $f : S \to R$ and $L = D - A$
- $Lf(x) = \sum_{y \sim x}(f(x) - f(y))$, where y denotes vertices adjacent to x

What does the Laplacian do?

$Lf(x) = \sum_{y \sim x}(f(x) - f(y))$, where y denotes vertices adjacent to x

Consider a chain graph, let's see what the combinatorial Laplacian is doing

$$\begin{aligned} Lf(v_i) &= (f(v_i) - f(v_{i-1})) + (f(v_i) - f(v_{i+1})) \\ &= (f(v_i) - f(v_{i-1})) + (f(v_i) - f(v_{i+1})) \\ &= \nabla f(v_i, v_{i-1}) - \nabla f(v_{i+1}, v_i) \\ &= \Delta f(v_i) \end{aligned}$$

What does the Laplacian do?

Solving the Laplace operator on a graph means finding the eigenvalues and eigenfunctions of the following equation.

$$Lf = \Delta f = \lambda f$$

If the domain is 2-D, what are the functions which have $\frac{\partial^2 f}{\partial x^2} = \lambda f$?
$\sin(x)$ and $\cos(x)$

So far we have established the following

- Laplacian fills in for Transition probability matrix
- It is related to the Laplace Partial Differential equation (in fact, it is the discrete form)
- Choose the smallest Eigenvalues to get the best approximation. Why?
- PVFs are abstract Fourier basis functions that represent an orthonormal basis set for approximating any value function. Why orthonormal?

What does the Laplacian do?

Solving the Laplace operator on a graph means finding the eigenvalues and eigenfunctions of the following equation.

$$Lf = \Delta f = \lambda f$$

If the domain is 2-D, what are the functions which have $\frac{\partial^2 f}{\partial x^2} = \lambda f$?
$\sin(x)$ and $\cos(x)$

So far we have established the following

- Laplacian fills in for Transition probability matrix
- It is related to the Laplace Partial Differential equation (in fact, it is the discrete form)
- Choose the smallest Eigenvalues to get the best approximation. Why? $L = D - \boldsymbol{A}$
- PVFs are abstract Fourier basis functions that represent an orthonormal basis set for approximating any value function. Why orthonormal?

What does the Laplacian do?

Solving the Laplace operator on a graph means finding the eigenvalues and eigenfunctions of the following equation.

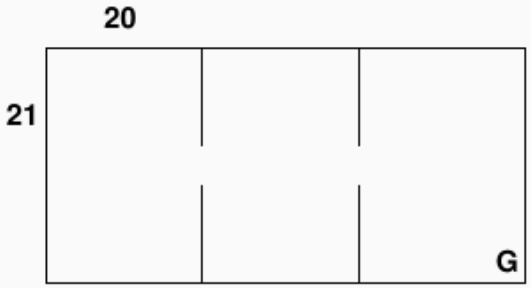
$$Lf = \Delta f = \lambda f$$

If the domain is 2-D, what are the functions which have $\frac{\partial^2 f}{\partial x^2} = \lambda f$?
$\sin(x)$ and $\cos(x)$

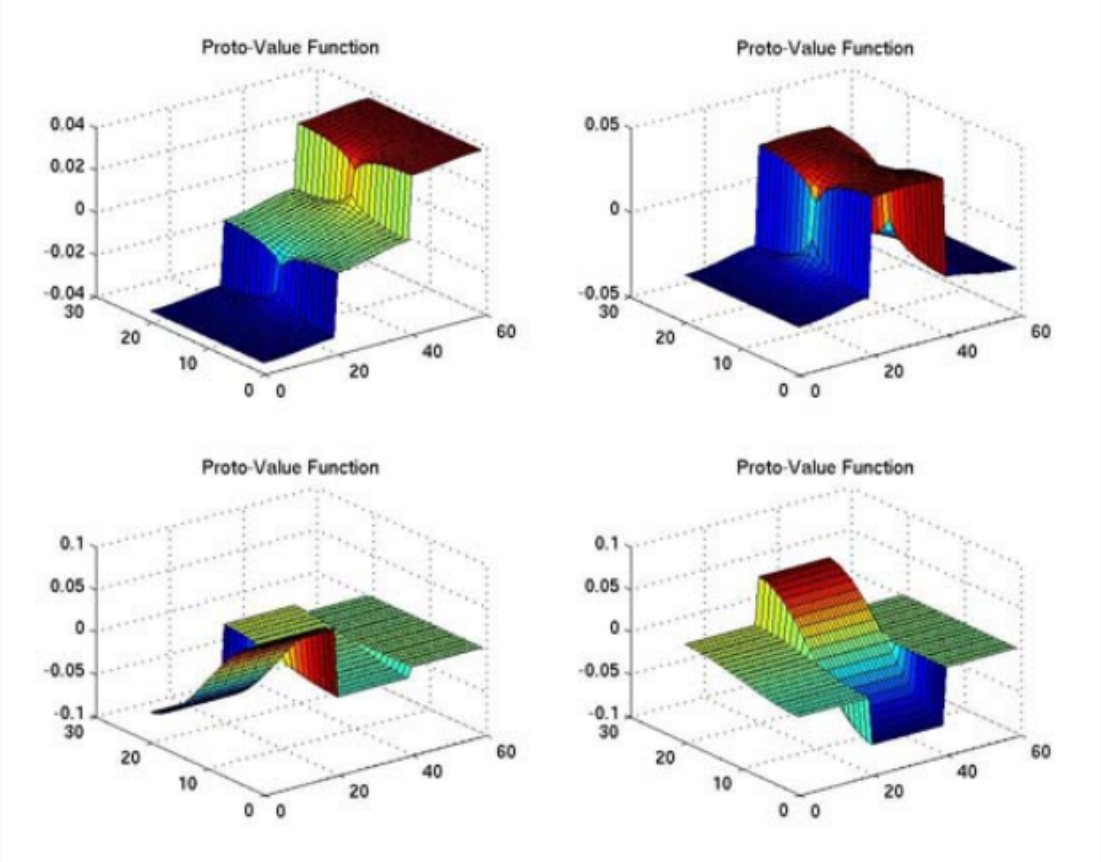
So far we have established the following

- Laplacian fills in for Transition probability matrix
- It is related to the Laplace Partial Differential equation (in fact, it is the discrete form)
- Choose the smallest Eigenvalues to get the best approximation. Why? $L = D - \boldsymbol{A}$
- PVFs are abstract Fourier basis functions that represent an orthonormal basis set for approximating any value function. Why orthonormal? **Symmetric Positive Semi-Definite!**

An Example



Total = 1260 states



Approximating the value function

- What can we do with the PVF basis?

Approximating the value function

- What can we do with the PVF basis?
- Least-Squares Approximation can be used

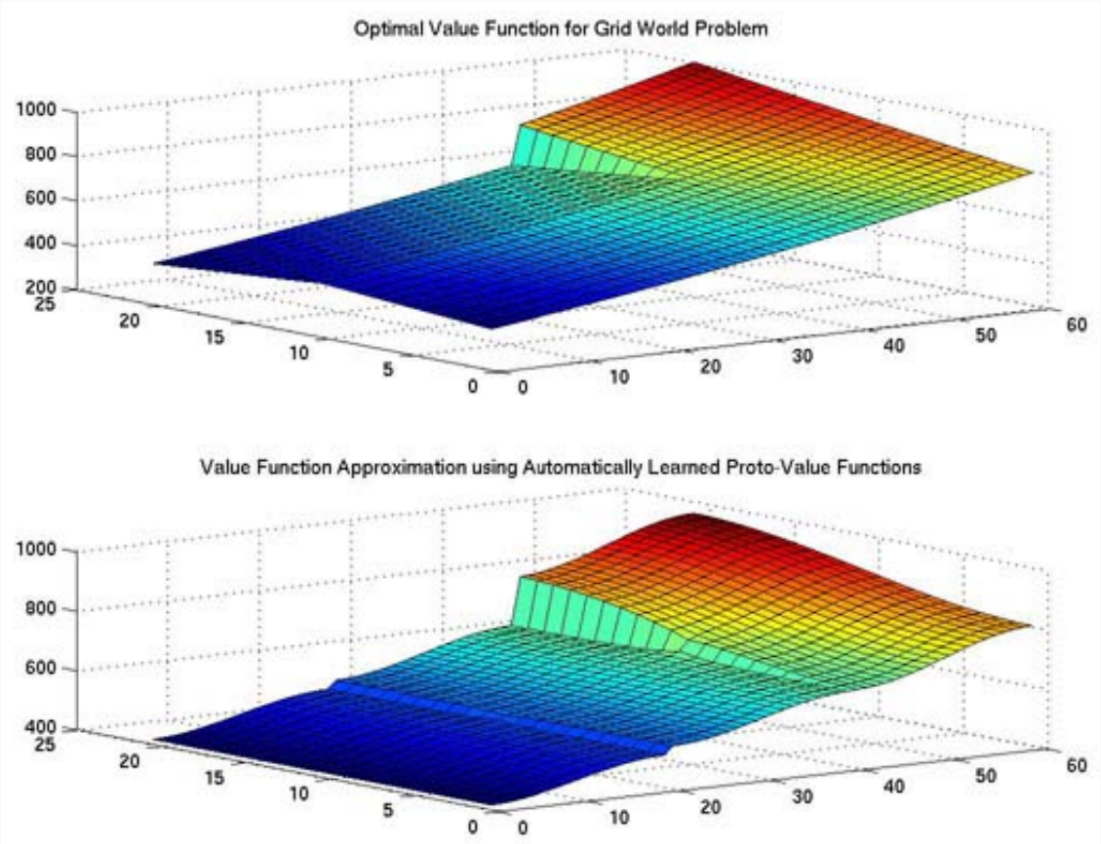


Figure 1: Using just 10 basis functions

Option Discovery

Option Discovery through Laplacian

PVFs capture the large-scale geometry of the environment, such as symmetries and bottlenecks. Why?

Option Discovery through Laplacian

PVFs capture the large-scale geometry of the environment, such as symmetries and bottlenecks. Why?

3.6 Proto-Value Functions and Large-Scale Geometry

We now formalize the intuitive notion of why PVFs capture the large-scale *geometry* of a task environment, such as its symmetries and bottlenecks. A full discussion of this topic is beyond the

3. Option Discovery through the Laplacian

PVFs capture the large-scale geometry of the environment, such as symmetries and bottlenecks. They are task inde-

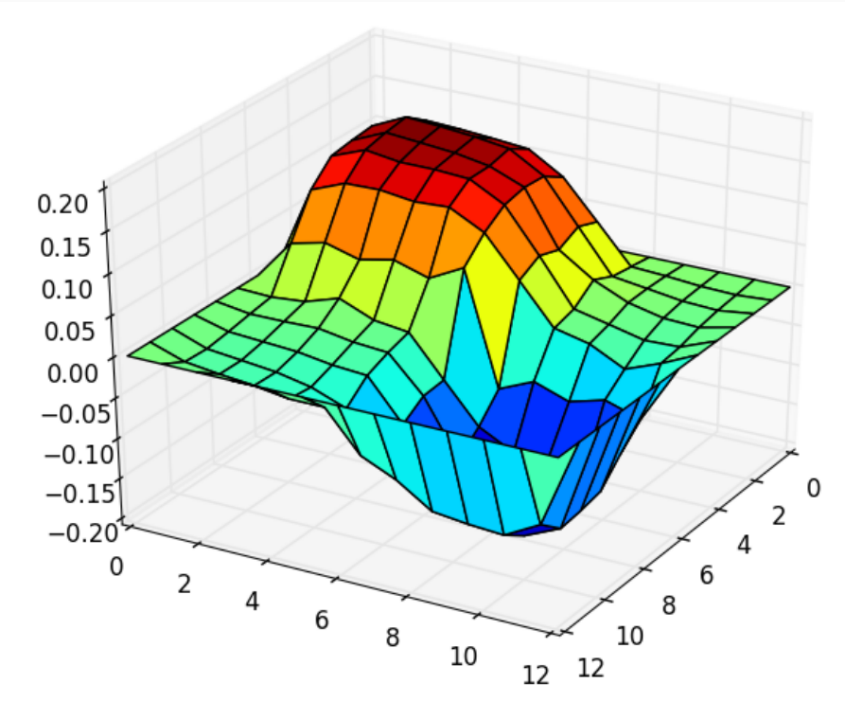
Symmetries because the Laplacian has spectral clustering properties

Bottlenecks and more

A PVF can be interpreted as the desire to reach the highest or the lowest point. *My intuition is that, when the value function is written as*

$$V^{\pi} = \alpha_1 V_1^G + \cdots + \alpha_n V_n^G$$

the points with the highest and lowest proto-value are where the change will be the most. Hence, they probably represent something important.



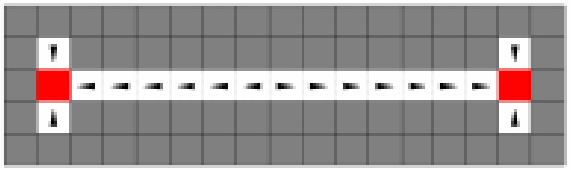
Eigenpurpose

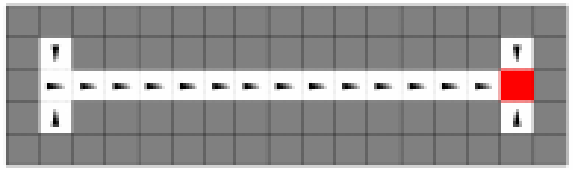
Define the following reward function, which represents a *eigenpurpose*

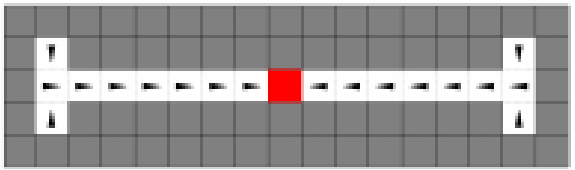
$$r_i^e(s, s') = e^T(\phi(s') - \phi(s))$$

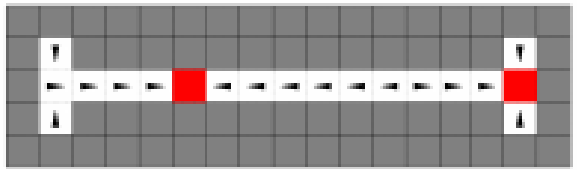
- This is a potential function. In the tabular case, the optimal option will just be a policy to go to the highest or the lowest point
- An *Eigenbehavior* is a policy which is optimal with respect to the eigenpurpose
- An action called terminate is augmented with the original set so that the agent can terminate when it is not able to get any more positive rewards
- They prove that for any option there is at least one state in which it terminates. This can be seen intuitively (the highest point, for example)

Learned Options

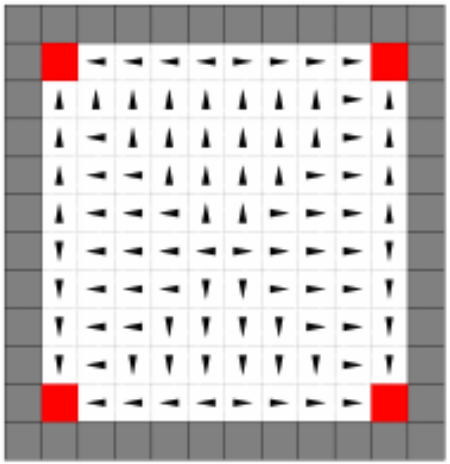


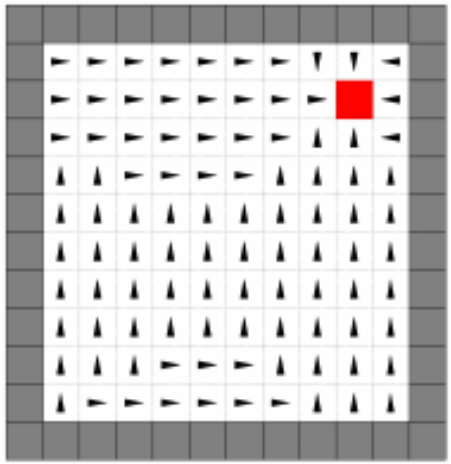


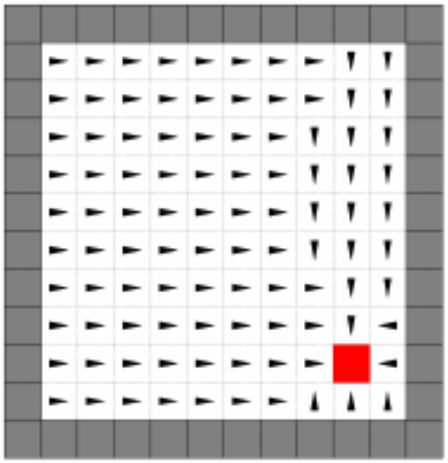


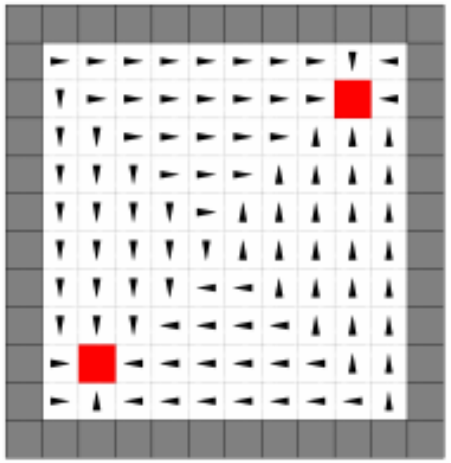


Learned Options

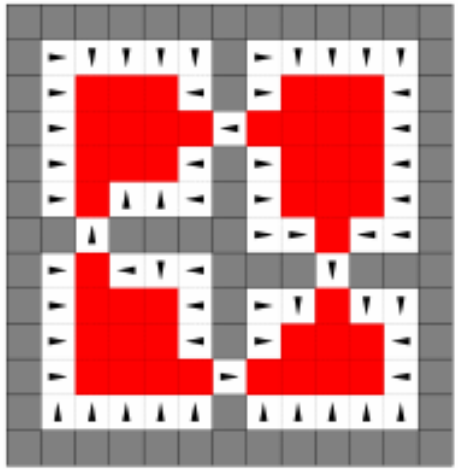


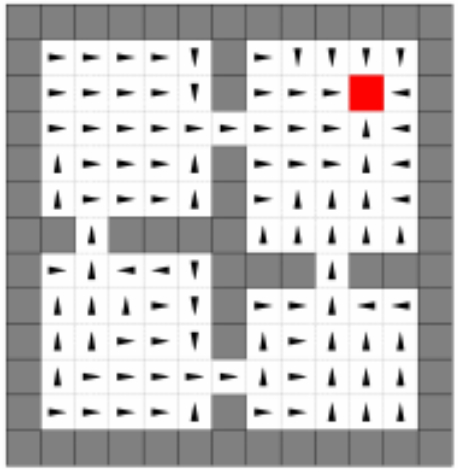


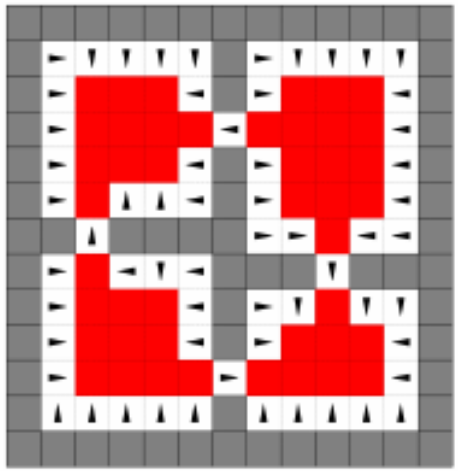


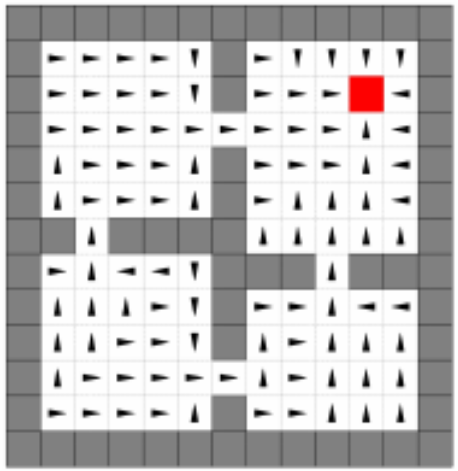


Learned Options









Learned Options

- Meaningful options like running down the corridor, going to the corner of the room and going through doorways were learned
- Bottleneck options were discovered, but they were not the first ones

Evaluation

To make things a little more concrete, we have the following.

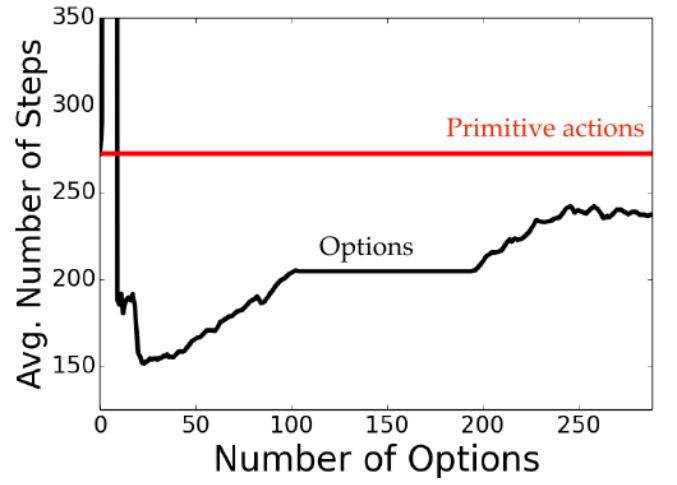
- How Eigenoptions present specific purposes?
- How Eigenoptions improve exploration?
- How Eigenoptions help agents accumulate rewards faster?

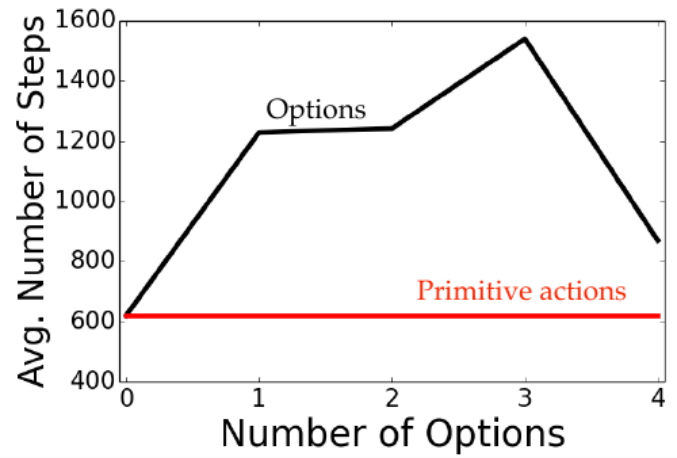
Exploration

Diffusion Time represents the expected number of steps required to navigate between two randomly chosen states when following a random walk.

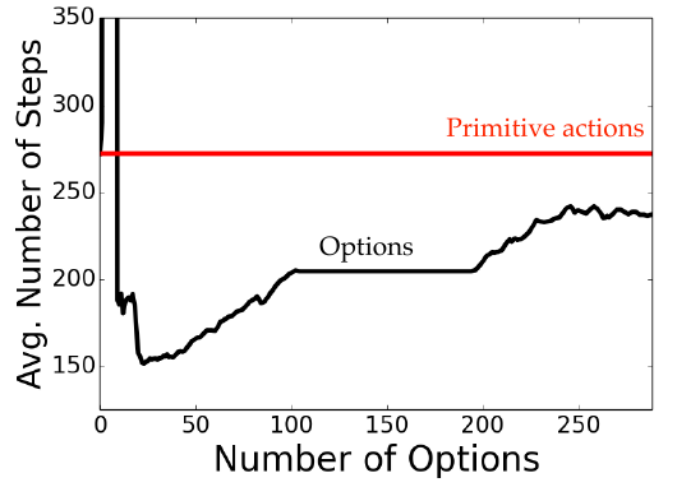
This is easy to calculate for small gridworlds because we can choose a random goal and run a uniformly random policy which encodes how many steps it takes to reach the goal. *Policy Evaluation.*

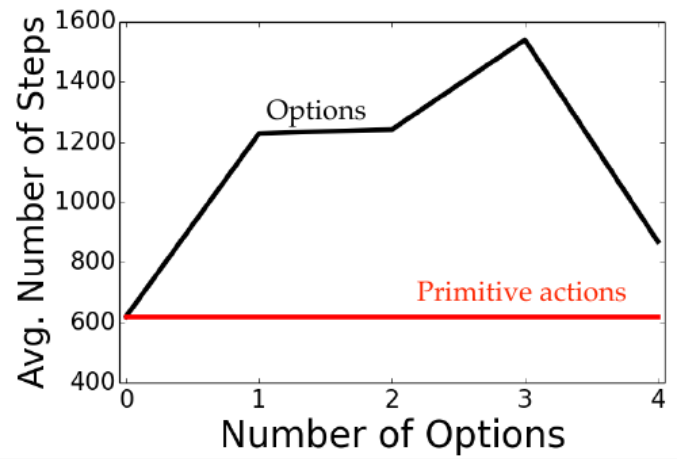
Diffusion Time Comparison





Diffusion Time Comparison

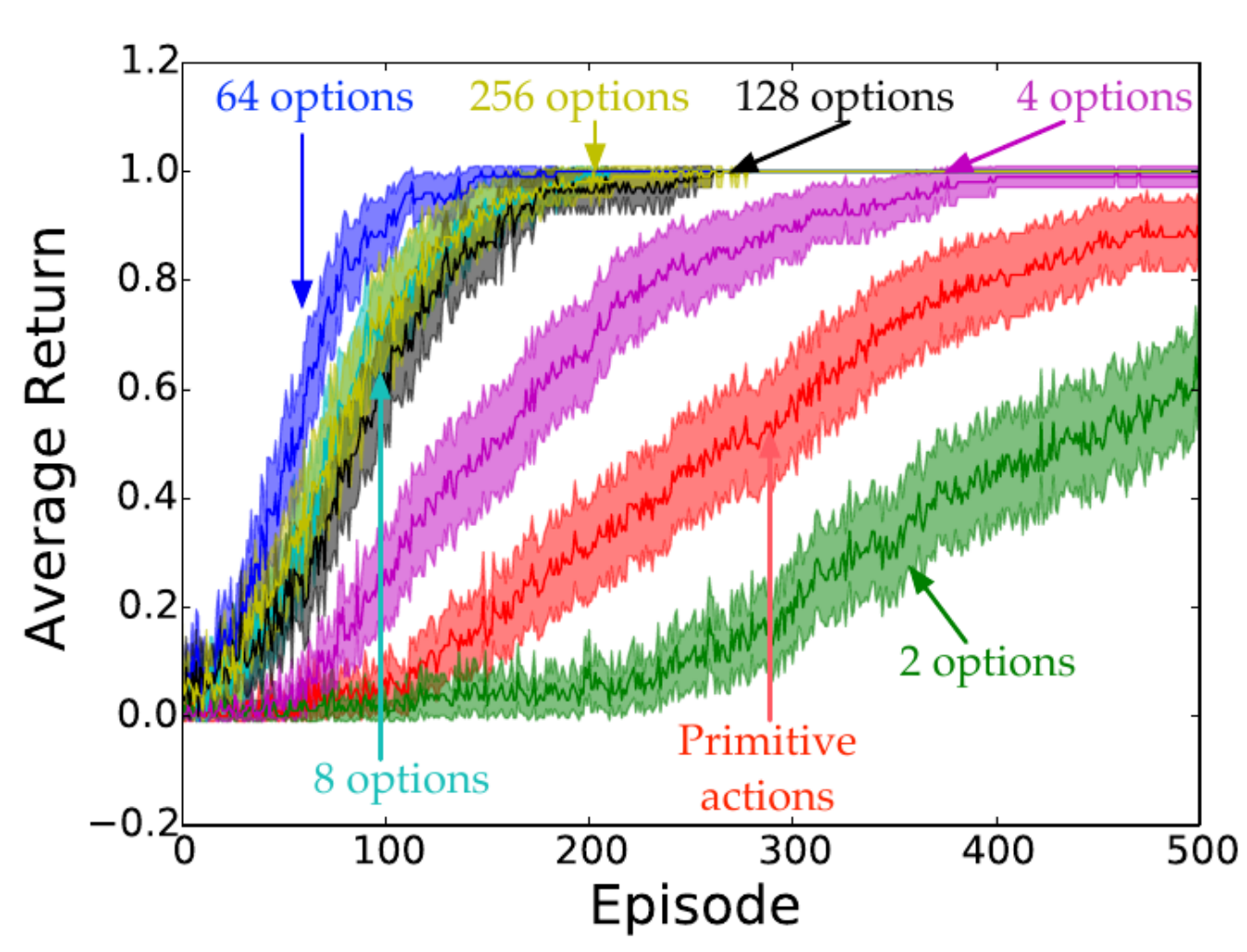




Possible short-coming in their analysis? Very few options considered.

Dithering.

Accumulating Rewards



64 options gave the best result for them. *The larger the number of options, harder it is to learn.*

Approximate Option Discovery

In real-world problems, $|\mathcal{S}|$ might be so big that the same state may not even be visited more than once. We need to resort to sample-based approaches if we cannot construct the adjacency matrix. And we might also want to use function approximation in such cases.

Sample-based Option Discovery

Sampling to construct the Adjacency Matrix does not extend well to linear approximation

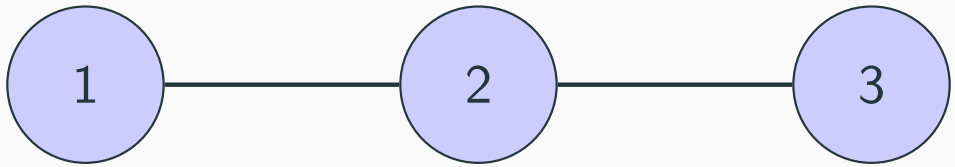
- Consider a matrix T, which is called the *incidence matrix* and is initially empty
- Add the entry $\phi(s') - \phi(s)$ to the matrix if it does not already exist
- Perform a SVD on T to get $T = U\Sigma V^T$ and use the columns of V as Eigenpurposes
- These are also the right Eigen vectors of the matrix T

Sample-based Option Discovery

Why does this work?

$$T = U\Sigma V^T$$

We can show that $T^T T = 2L$. So the Right Eigen Vectors are the Eigenvectors of the Laplacian



$$L = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad T = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

Function Approximation

The method followed is the same

- Sample transitions and get s, s'
- Use the *set* datastructure to avoid duplicates and store $\phi(s') - \phi(s)$ in T
- Perform a Singular Value Decomposition and use the right Eigen Vectors

But now because we are using function approximation is being used, it no longer makes sense to choose the smallest Eigen Values.

This can be understood by thinking of an adversarial state representation which is a permutation of the tabular case.

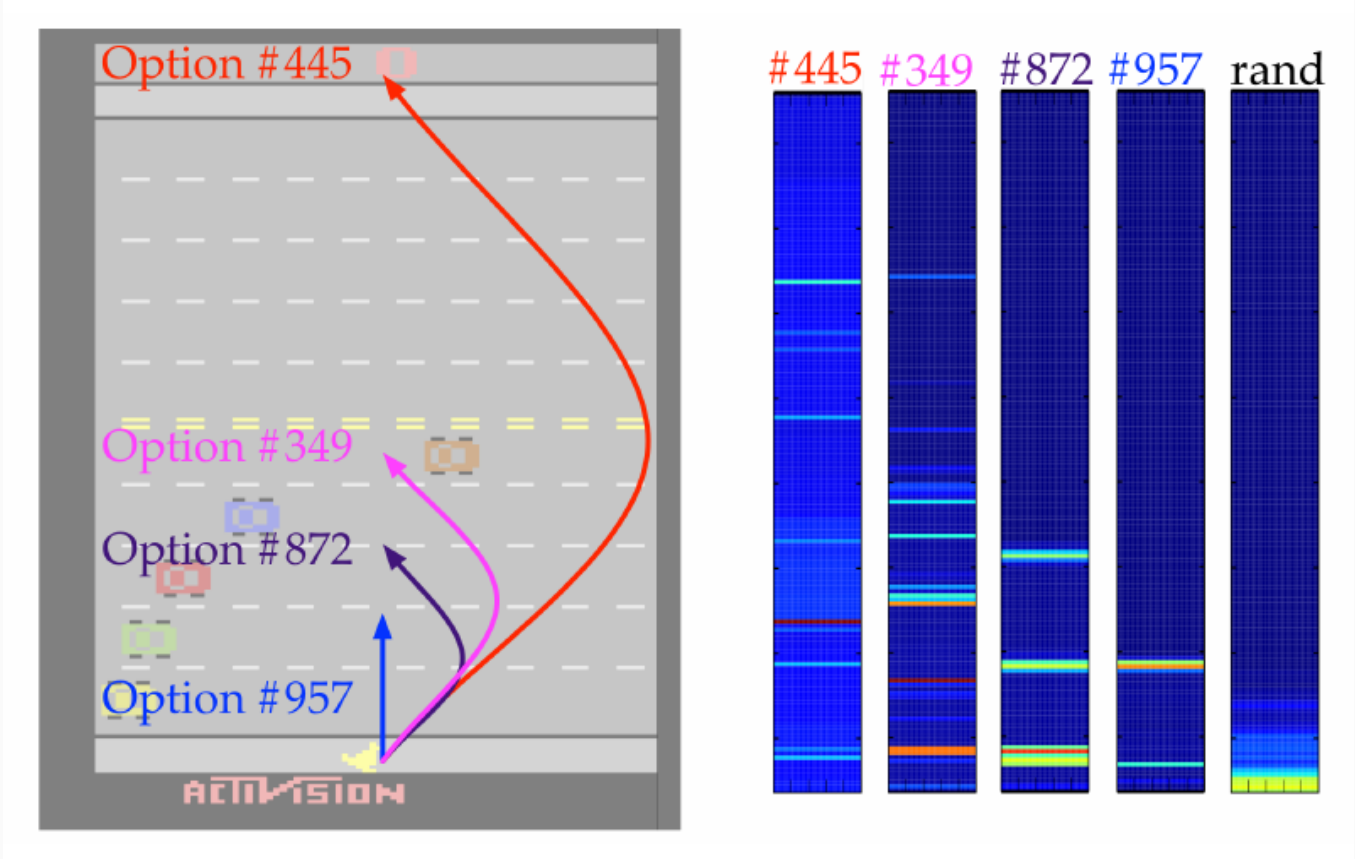
Experiments in Atari

- The RAM State of the game was used as the state representation
- All the Eigen Vectors need to be evaluated, no other choice
- The next best option is chosen using

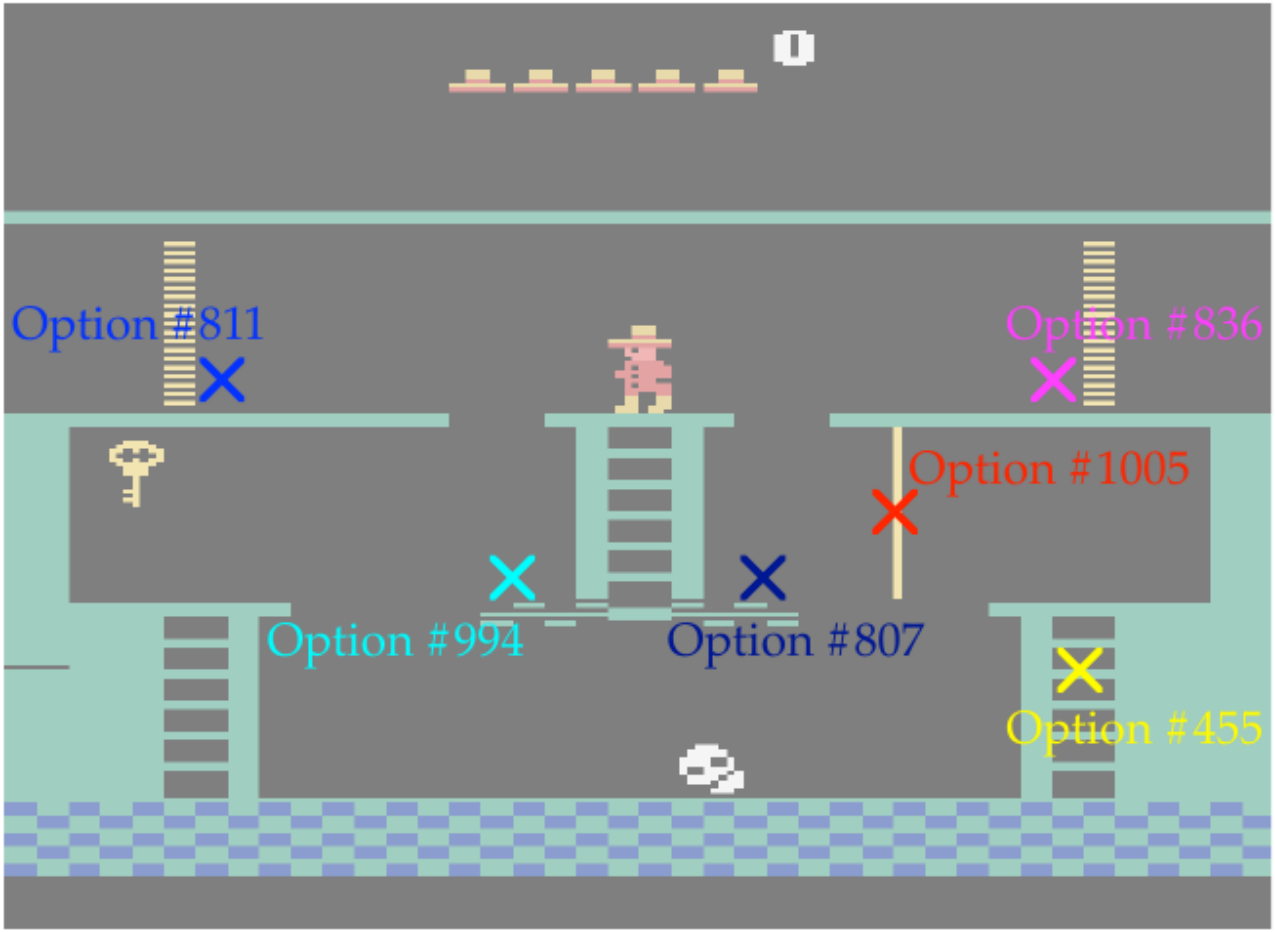
$$arg \max_{b \in \mathcal{A}} \int_{s'} p(s'|s, b) r_i^e(s, s')$$

- This is essentially a one-step look-ahead. Is this a reasonable assumption?

Learned Options



Even though the agent did not learn options to maximize the rewards, the rewards received (28) is close to state-of-the-art



A few options were similar to the handcrafted ones in [1]

Conclusion

Summary

- *Always read old papers carefully*
- PVFs give representations which capture the geometry of the environment
- The geometry is often directly related to what options are useful
- *These options are expected to have good transfer properties*

Questions?

References i

T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum.
Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation.
In *Advances in neural information processing systems*, pages 3675–3683, 2016.

M. C. Machado, M. G. Bellemare, and M. Bowling.
A laplacian framework for option discovery in reinforcement learning.
arXiv preprint arXiv:1703.00956, 2017.

S. Mahadevan.
Proto-value functions: Developmental reinforcement learning.
In *Proceedings of the 22nd international conference on Machine learning*, pages 553–560. ACM, 2005.

References ii


S. Mahadevan and M. Maggioni.
Proto-value functions: A laplacian framework for learning representation and control in markov decision processes.
Journal of Machine Learning Research, 8(Oct):2169–2231, 2007.