Mining State-Space Graphs to Build Value-Function Representations

Bryan Silverthorn
bsilvert@cs.utexas.edu

Craig Corcoran
ccor@cs.utexas.edu

15 November 2011

Abstract

The engineering of an appropriate state space representation is one of the essential prerequisites to the use of reinforcement learning (RL) in a new domain. Constructing a successful representation, however, is often difficult and requires an understanding of the peculiarities of a given domain. Recent work of Mahadevan and Maggioni [2006], among others, has applied techniques from spectral graph theory to the problem of automatically discovering or augmenting state representations, but only to small and amenable domains. This first submission works toward extending those techniques to tackle larger, more challenging domains. It considers three related problems: *applying* spectral methods to the unusual tree-like state graphs of board games, *adapting* these methods to acquire state information from sampled human gameplay, and *scaling* them to combinatorial state spaces. The simple game of Tic-Tac-Toe is used here for small-scale exploration, to prepare for a much larger domain, the classic game of Go, in future work.

1 Introduction

The field of reinforcement learning (RL) concerns itself with situations in which an agent learns from interactions with its environment. Value-based approaches to RL estimate the expected utility of each possible state of the agent and its environment—the so-called *value function*—and use that value function to produce a *policy* that maps from states to actions.

Just as feature engineering is often critical to the success of supervised learning methods, the success of value-based RL depends on the quality of the representation used to estimate the value function. This work will explore new methods for automatically generating such a representation, with a focus on approaches that scale to large discrete domains.

1.1 Linear Value Function Approximation

In environments where the number of states is too large to permit a tabular representation of the value function, some form of approximation must be employed. A linear architecture is often chosen for simplicity and ease of analysis. In such a representation, the value $V(s)$ of a state s is estimated as a linear combination of features $\phi(s)$ of that state:

$$V(s) = w^T \phi(s) \tag{1}$$

As in any linear prediction scheme, the quality of the resulting value function and the performance of the resulting policy depends critically on the basis functions or state features $\phi(s)$. These features are typically constructed by hand, a labor-intensive process that requires extensive domain knowledge.

Prior work in automatically generating representations for value function approximation include population-based search methods [Whiteson and Stone, 2006], methods based on Bellman error [Parr et al., 2007], and methods based on augmented Krylov techniques [Petrik, 2007], as well as many others. This work builds upon a particularly promising approach that leverages results from the harmonic analysis of graphs to aid in discovering useful representations [Mahadevan and Maggioni, 2006, Coifman and Maggioni, 2006, Wang and Mahadevan, 2009].

1.2 Spectral Representation Discovery

Spectral graph theory provides many useful tools for analyzing a graph and functions on that graph [Chung, 1997]. The graph Laplacian L and its normalized variants play a central role in this analysis. Given the adjacency matrix of a weighted graph W, the graph Laplacian is defined as

$$L = D - W \tag{2}$$

where D is defined as the row sums of W:

$$D = \sum_i W_{ij} \tag{3}$$

The normalized graph Laplacian is defined to be:

$$\mathcal{L} = D^{-1/2} L D^{-1/2} \tag{4}$$

The graph Laplacian has a number of useful properties; among others, its eigenvectors form a good basis for representing smooth functions on the graph.

The state space of a finite Markov decision process (MDP) can be represented as a graph where each vertex is a state and edges represent state transitions. Using this state adjacency graph W, we can form the (normalized) graph Laplacian and use the eigenvectors as a basis for performing linear value function approximation.

Recent work [Petrik, 2007, Mahadevan et al., 2006, Mahadevan and Maggioni, 2006, 2007] has begun to examine the use of harmonic analysis of graphs in this framework, but have focused on domains small enough for direct application of approaches similar to those described above. As the state space grows, however, representing the full adjacency graph and finding its eigenvectors becomes prohibitively expensive. Some prior research has applied Kronecker matrix factorization to the problem of scaling [Johns et al., 2007]. Our work will instead move toward an approach based on clustering the state graph.

We focus on two domains in this preliminary submission: first, a simple two-room grid world that highlights the desirable properties of the Laplacian eigenvectors in terms of representing the topology of a state space; and second, a classic game, Tic-Tac-Toe (TTT), that serves as an easily-understood domain in which to develop and evaluate techniques for applying spectral methods to the unusual state graphs of board games. Both domains lead us toward future work on the larger, more challenging board game of Go.

2 Grid World Domain

Existing work on spectral methods for feature discovery has focused on domains where the state graph has a simple spatial interpretation, and on grid worlds in particular. The standard two-room domain used by Mahadevan and Maggioni [2006] is therefore a good first test of our implementation, and a good example of the value of spectral methods for uncovering useful representations of the state space.

The domain consists of two (approximately) equally sized rooms connected by a single door. The adjacency graph is formed by connecting each grid point to its neighbors by an edge with weight one; the walls between the rooms are not included in the state space. Using this symmetric adjacency graph as W, we form the normalized graph Laplacian as described in Eq. (4). The first nine eigenvectors, ordered from the smallest to the largest eigenvalues, are shown in Fig. 1.

Because the eigenvectors with the smallest eigenvalues correspond to the smoothest functions over the state space, when building a feature set, the Laplacian eigenvectors are often included in increasing order of their eigenvalues. This scheme starts with a constant vector and includes increasingly higher-frequency components in the representation.

As seen in Fig. 1, the Laplacian eigenvectors form interesting functions across the rooms, capturing domain symmetries and adapting to the spatial connectivity of the states. Linear combinations of these functions are clearly capable of approximating smooth functions in the two-room domain. This visual example motivates our application of the eigenvectors of the graph Laplacian to game state graphs—although the richer topology of such graphs makes it difficult to predict its effectiveness.

3 Tic-Tac-Toe Domain

With the goal of scaling up to larger board games, and Go in particular, we choose Tic-Tac-Toe (TTT) as our second domain. Its properties are similar to Go, including a tree-like state space and a discrete, grid-based board representation. It is also small enough that we can compute optimal play, solve for the optimal value function, and find the Laplacian eigenvalues on the full state-space graph.

In the following sections, state rewards are defined as 1 for all winning states, -1 for all losing states, and 0 for draws. Similarly, the two players, as well as their pieces on the board, are denoted by 1 (for the first player) and -1 (for the second player), with 0 representing an empty space. x_{ij} refers to the configuration of board x in the i^{th} row and the j^{th} column.

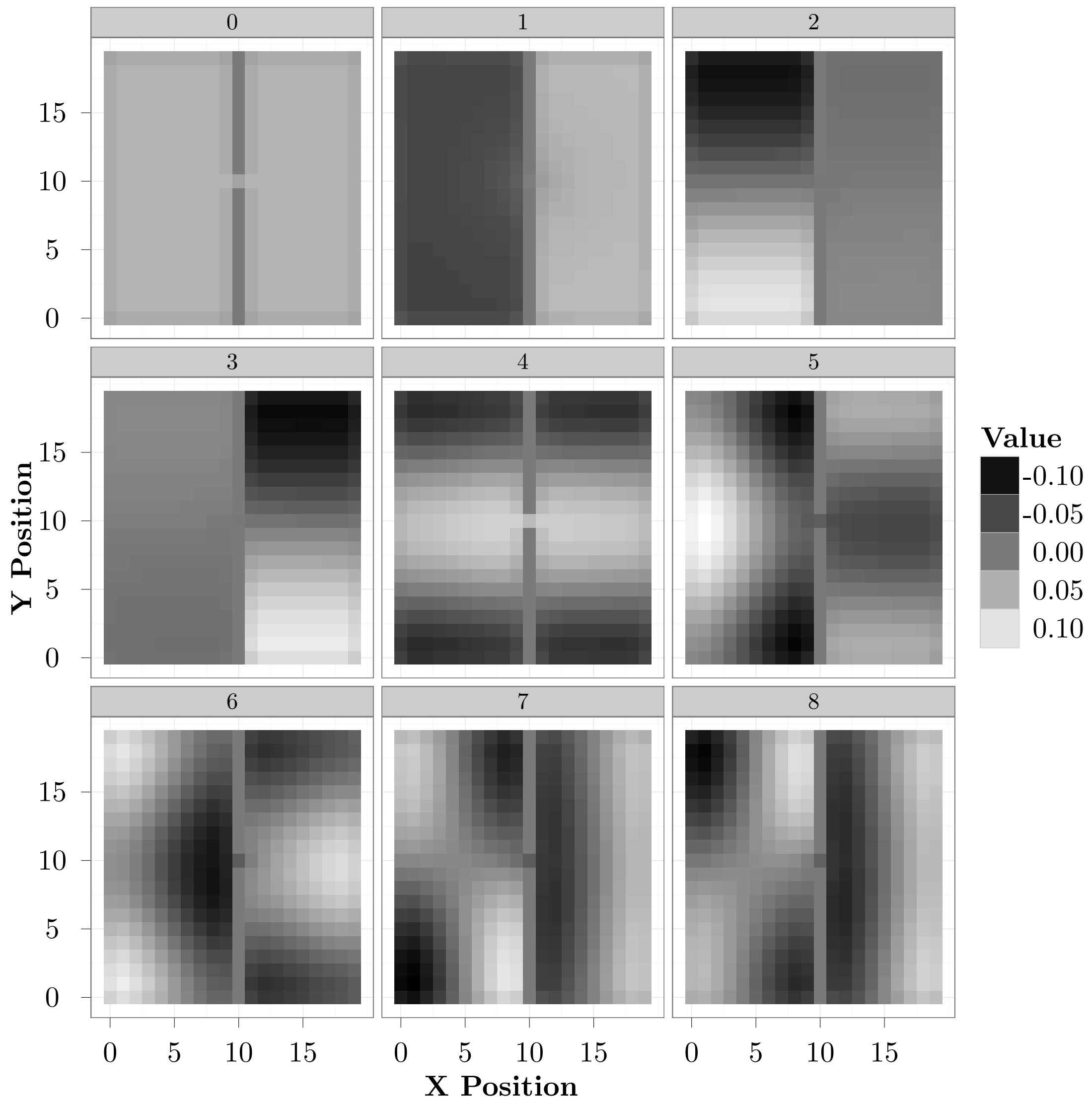

Figure 1: Visualization of the first nine eigenvectors of the graph Laplacian in the two-room domain. A vertical wall separates the two rooms with a single-space door connecting them in the middle. Note that the functions are adapted to the state space connectivity: the second eigenvector separates the two rooms, the third and fourth are near zero in one room while partitioning the other room, and the fifth eigenvector separates the doorway from the corners of the room.

3.1 Mining the Gameplay Graph

When choosing a graph with which to perform spectral analysis for representation discovery in TTT, the simplest and most obvious option is the adjacency graph for neighboring board states. We define the gameplay adjacency graph, displayed in Fig. 2, as having weight-one edges between each board and all previous or subsequent boards.

From the symmetric adjacency matrix representation of the graph, we formed the normalized graph Laplacian as in Eq. (4). A visualization of the first nine eigenvectors is given in Fig. 3. Again, the eigenvectors capture much of domain's symmetry. They also provide useful features for evaluating gameplay, such as separating corner versus center moves.

It should be noted that we are discarding some information: Tic-Tac-Toe is fundamentally a directed game, but we consider only the undirected connections between states. This simplification is convenient, since a symmetric matrix is necessary for computing the eigenvectors, but alternatives such as the directed graph Laplacian [Chung, 2005] exist.

3.2 Mining a State Affinity Graph

A complete graph represention of the rules of a game, a *gameplay graph*, is clearly a rich source of information from which to extract representation information. The graph representation of a nontrivial game, however, is often intractably large. Approaches to larger games will be explored in future work; for now, we will focus on alternative graph representations that may be useful in these contexts.

If we can construct a measure of the similarity or "affinity" between arbitrary game states, that affinity function could be used to build an alternative graph representation of a game. This representation, an *affinity graph*, represents information differently than the gameplay graph, but may include enough information to allow useful features to be extracted.

In Tic-Tac-Toe, one obvious distance function is the the Hamming distance between two board configurations

$$d_H(x, z) = \sum_{i=1}^{3} \sum_{j=1}^{3} (1 - \delta_{x_{ij} z_{ij}}) \tag{5}$$

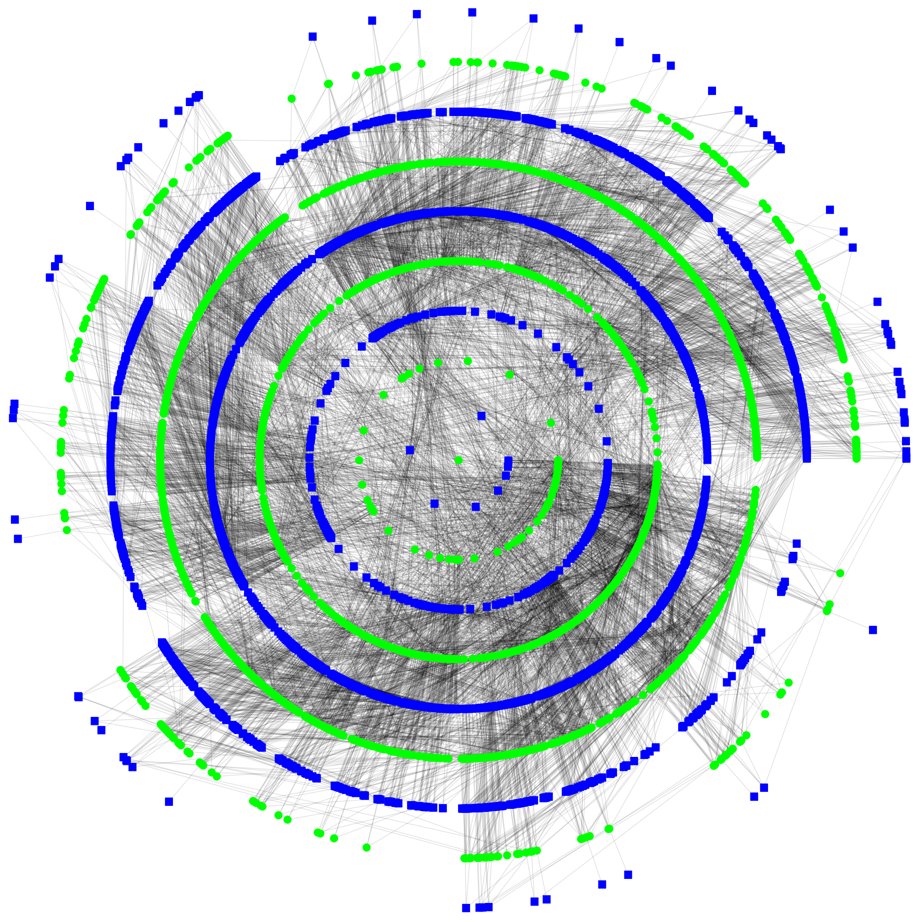

Figure 2: The complete TTT gameplay graph. Each vertex represents a board configuration, with the empty starting board at the center of the graph, and each edge represents a move. Green circles denote positions in which the first player is to move, as do blue squares for the second player. Every path from the root to a leaf represents a complete possible Tic-Tac-Toe game. While the tree-like structure is clear, the complete gameplay graph also includes many nodes in which multiple game paths intersect, i.e., it remains a DAG. This structural complexity will largely disappear in future work, when only sampled paths are available.

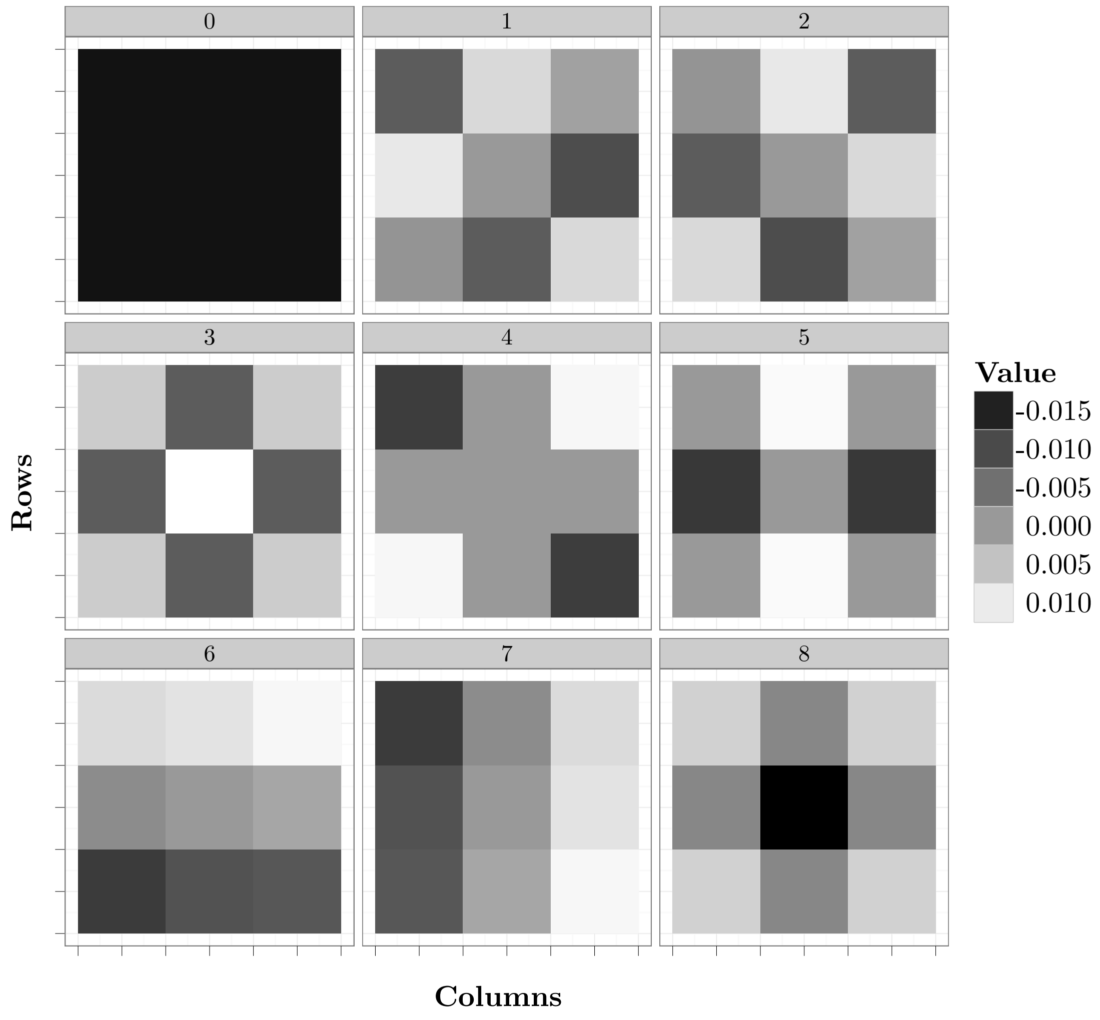

Figure 3: Visualization of the nine smallest eigenvectors of the Laplacian of the TTT gameplay graph. Each grid is associated with an eigenvector, and each cell of the grid is filled according to the value of that eigenvector on the board state reached by the first player placing a mark in that cell on an empty board. The smallest eigenvector is a constant function, as expected, while the others capture useful aspects of the game rules; eigenvectors 1 and 2, and 6 and 7, for example, show that these features have recovered the game's symmetry over rotated board states.

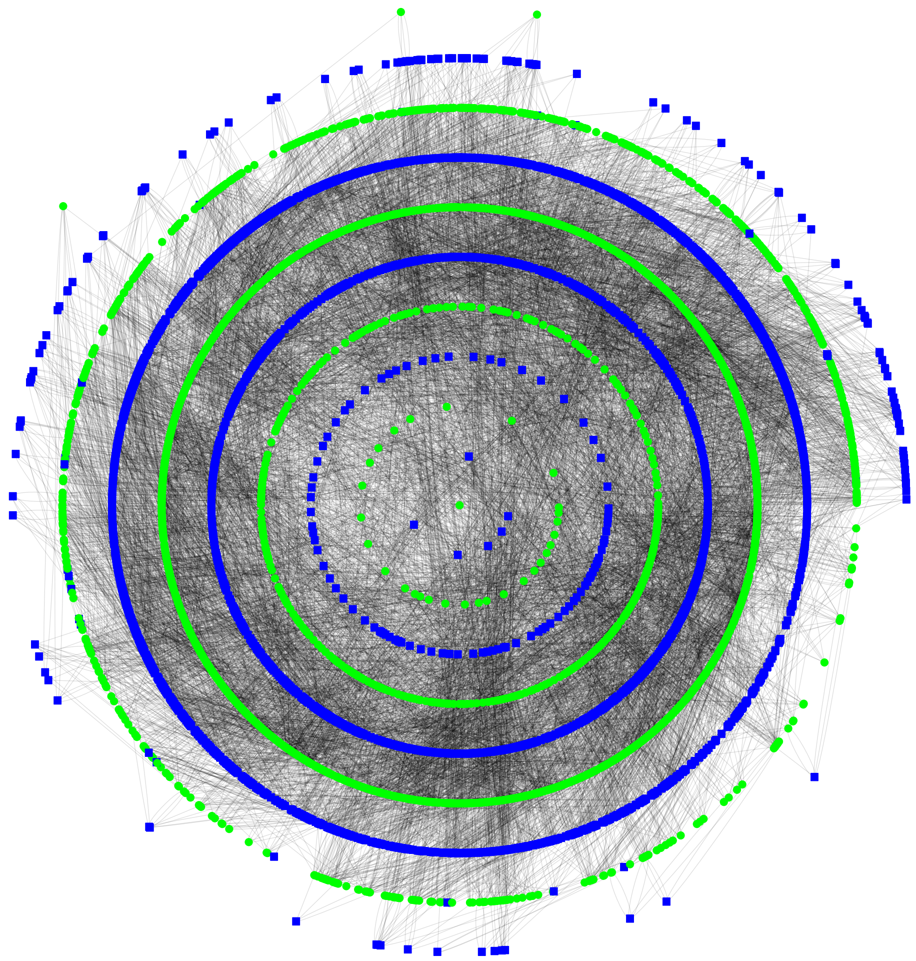

Figure 4: The complete TTT “affinity” graph: TTT board states connected by edges according to the Euclidean distance between flattened grid representations. The comparison to Fig. 2 (formatting is identical) shows that the affinity graph captures much of the same structure, but is significantly messier; for example, many more edges cross between distant nodes. In later experiments, we will see that spectral features acquired from this graph nonetheless form an effective state representation.

where δ is the Kronecker delta. An ideal distance measure, however, would be compatible with methods such as k-means clustering, and with the ball-tree structure [Omohundro, 1989] employed for efficient graph construction. We therefore first map each board state to a vector of hand-selected state features, then use a vector norm to measure distance; we can approximate the Hamming distance by simply flattening our board representation into a vector, then using, e.g., Euclidean distance. The Gaussian kernel

$$g(x) = \exp(-\frac{x^2}{2\sigma^2}) \tag{6}$$

is used to convert distances into affinities when necessary.

Given an affinity function $a(x, z)$, a graph can be constructed by placing edges from each game state to its k most similar game states, weighting each edge by the affinity between the two states. Figure 4 shows such a graph of TTT game states. While less clean than the gameplay graph, it nonetheless appears to encode some of the same information.

While constructing an affinity graph requires defining a simple state feature set by hand, the following sections show that the features mined from this graph are much more useful than the features used to construct the graph.

3.3 Evaluation in Value Function Regression

The goal of this work is to construct features that are useful for linear approximations of the *state value function*; that is, a linear combination of our features should approximate the utilities of our game states, with utility given by the fixed point of the deterministic Bellman equation [Bellman, 1957]

$$V(x) = \max_{a \in \mathcal{A}_x} [R(x) + \gamma V(T(x, a))] \tag{7}$$

where $V(x)$ is the value of state x, $\mathcal{A}_x$ is the set of possible actions in x, $R(x)$ is the reward given by transitioning to x, and $T(x, a)$ is the state that results from taking action a in x.

Figure 5 presents the outcome of using spectral and baseline random features to predict the values of TTT states. This experiment proceeds by

- computing the true value function against the optimal opponent using value iteration;

- mapping each state to a vector of real-valued features, always including at least the raw grid features used for affinity graph construction;
- splitting the state space for cross validation;
- training a linear regression method on the feature vectors of the training states, labeled with their true value;
- and comparing the linear prediction of the test states' value to their computed true value.

The results show that prediction accuracy improves with the number of basis vectors: spectral features computed from the either gameplay or affinity graphs do capture relevant aspects of the game.

3.4 Evaluation in a Tic-Tac-Toe Policy

The important test of a value-function representation, however, comes when the value function is used inside a policy—a mapping from states to actions—that operates in a domain. Figure 6 plots the performance of a policy that is greedy with respect to its value function using a range of numbers of state features drawn from various feature sets. This experiment proceeds similarly to that of Section 3.3, except that its value function is computed against a uniform-random opponent, and that it measures the average reward, playing against that opponent, of a policy whose internal value function is represented by the learned weight vector and associated feature set.

As in our measurement of value prediction error, both the affinity and state graph features outperform the random baseline, and their performance improves with additional eigenvector features—to the point where the associated policy consistently wins the vast majority of its games versus a random opponent.

4 Future Work

The number of game states in most board games increases combinatorially with the size of the board. The set of all possible Go board configurations, for example, becomes intractably large even on only a small 9-by-9 board.

This work therefore moves toward the case where we do not have direct access to the full state space, and instead must consider only sample trajec-

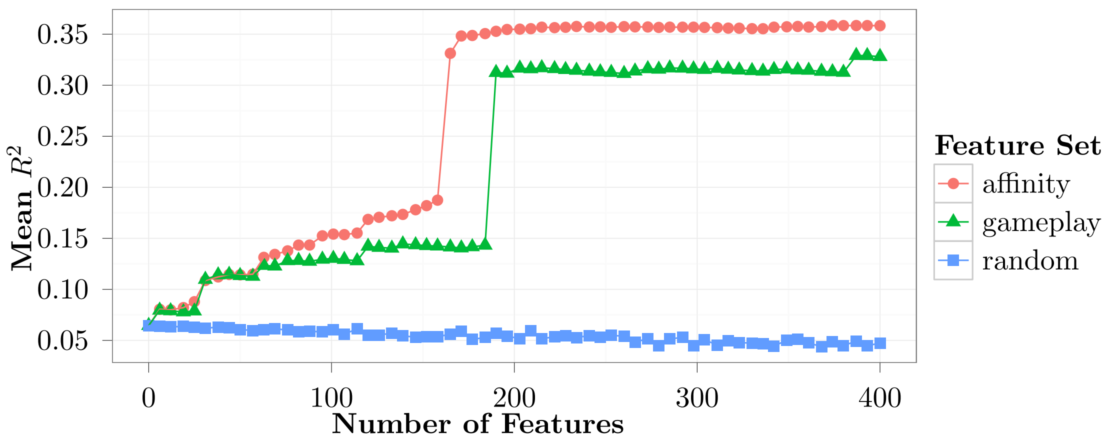

Figure 5: The mean R^2 score of value function prediction versus the number of feature vectors added from the specified feature set, under 10-fold cross validation using ridge regression ($\alpha = 1.0$) in the TTT domain. These features are added to the set of raw, flattened- grid features used to construct the affinity graph. As expected, random features do not generalize, while additional spectral features computed from both the affinity and gameplay graphs improve prediction accuracy.

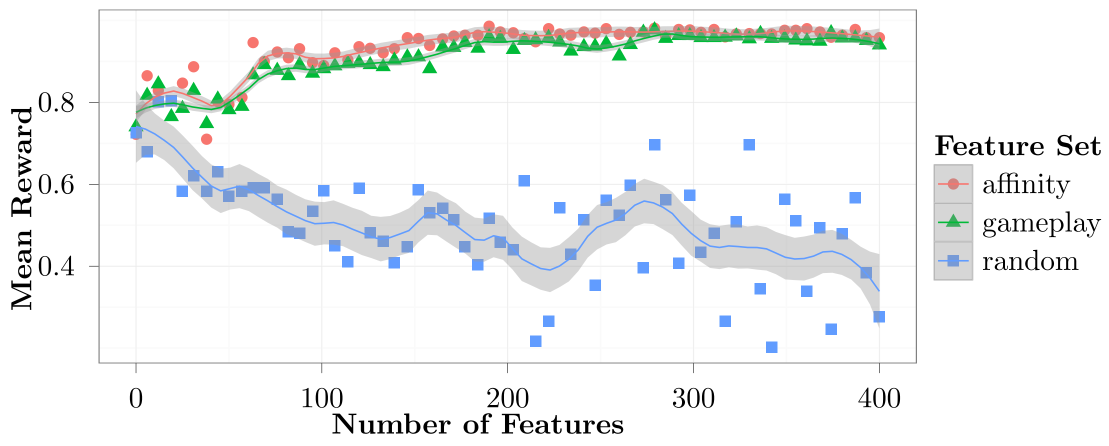

Figure 6: The mean reward obtained by a TTT policy over 1,000 games, averaged under 10-fold cross validation, using linear function approximation with weights tuned via ridge regression as in Section 3.3. Again, the eigenvector features are clearly informative.

tories of games already played. These trajectories are informative because they indicate important regions of the game space, but as Fig. 7 illustrates, they are often also disjoint; they form long, isolated chains that may not provide a sufficiently rich topology to apply spectral representation-discovery techniques directly. State affinity information may allow us to join these chains—perhaps by aliasing states with an existing clustering method.

We must also work toward approaches to apply when finding eigenvalues is computationally infeasible even on the sampled Laplacian matrix. Clustering as a preprocessing step to reduce the size of the graph may be a viable remedy to this problem as well.

For both statistical and computational reasons, then, the development and use of effective state aliasing could be an important ingredient in scaling to more challenging domains. It will be our primary focus in future work.

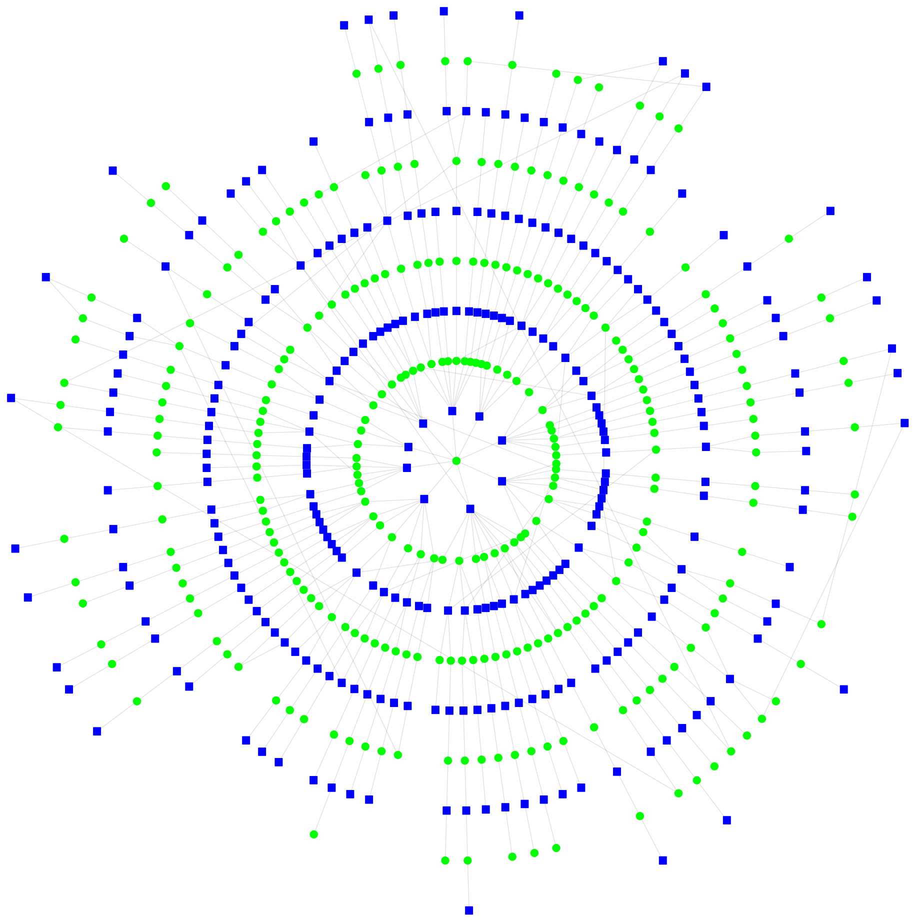

Figure 7: An incomplete TTT gameplay graph constructed from sampled games, i.e., from repeated stochastic walks over the complete gameplay graph of Fig. 2. Although some edges cross between game trajectories, most are isolated, and the rich topology of the game is not evident. This problem is likely to be even more pronounced in samples from a more complex game. If future work is to apply spectral analysis to sampled game trajectories, more informed graph construction must be performed.

5 Conclusion

This first submission has shown spectral analysis to be a promising method for automatically discovering linear representations of decision-process value functions, even in a board game domain less clearly amenable to spectral analysis than the spatial domains evaluated in existing work. The features learned in these experiments, for example, are sufficiently informative to represent an effective policy for Tic-Tac-Toe. This submission has established some initial techniques for representation discovery in a broad class of game-like domains, identified relevant challenges of scale, and proposed avenues for overcoming those obstacles. Future work will continue to explore these paths.

References

R. Bellman. *Dynamic programming.* 1957.

F. Chung. *Spectral graph theory.* 1997.

F. Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, 2005.

R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 2006.

J. Johns, S. Mahadevan, and C. Wang. Compact spectral bases for value function approximation using Kronecker factorization. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007.

S. Mahadevan and M. Maggioni. Value function approximation with diffusion wavelets and Laplacian eigenfunctions. *Advances in Neural Information Processing Systems (NIPS)*, 2006.

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*, 2007.

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

S. M. Omohundro. Five balltree construction algorithms. Technical report, 1989.

R. Parr, C. Painter-Wakefield, L. Li, and M. Littman. Analyzing feature generation for value-function approximation. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007.

M. Petrik. An analysis of Laplacian methods for value function approximation in MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

C. Wang and S. Mahadevan. Multiscale analysis of document corpora based on diffusion models. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *The Journal of Machine Learning Research*, 2006.