Mining State-Space Graphs to Build Value-Function Representations

Bryan Silverthorn
bsilvert@cs.utexas.edu

Craig Corcoran
ccor@cs.utexas.edu

7 December 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 work extends those techniques to tackle much larger domains, focusing on the unique challenges of board games. It considers three related problems: *applying* spectral methods to the unusual tree-like state graphs of such games, *adapting* these methods to acquire state information from sampled expert gameplay, and *scaling* them to combinatorial state spaces. The approach is verified on the simple game of Tic-Tac-Toe, before applying it to the vastly more difficult game of Go. In both domains, spectral analysis proves able to automatically generate features relevant to value function prediction.

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 how well the value function is represented. 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 large, an agent using a value function must employ some form of approximation. 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 instead moves toward an approach based on clustering the state graph.

This paper has three parts. First, we replicate existing results on a simple two-room grid world that highlights the desirable properties of the Laplacian eigenvectors for state space representation. Second, we develop and evaluate techniques for applying spectral methods to the unusual state graphs of board games. The classic game of Tic-Tac-Toe (TTT) serves as an easily-understood domain in which to verify these techniques. Third, we extend our methods for TTT to the much larger, much more challenging game of Go, using them to extract meaningful information from recorded expert gameplay.

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

Spectral methods are known to perform well on small, spatial environments like the two-room domain. The central contribution of this paper is their extension to the larger, non-spatial domains of board games. With the goal of scaling up to larger board games, and Go in particular, we first develop our approach on Tic-Tac-Toe (TTT). 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

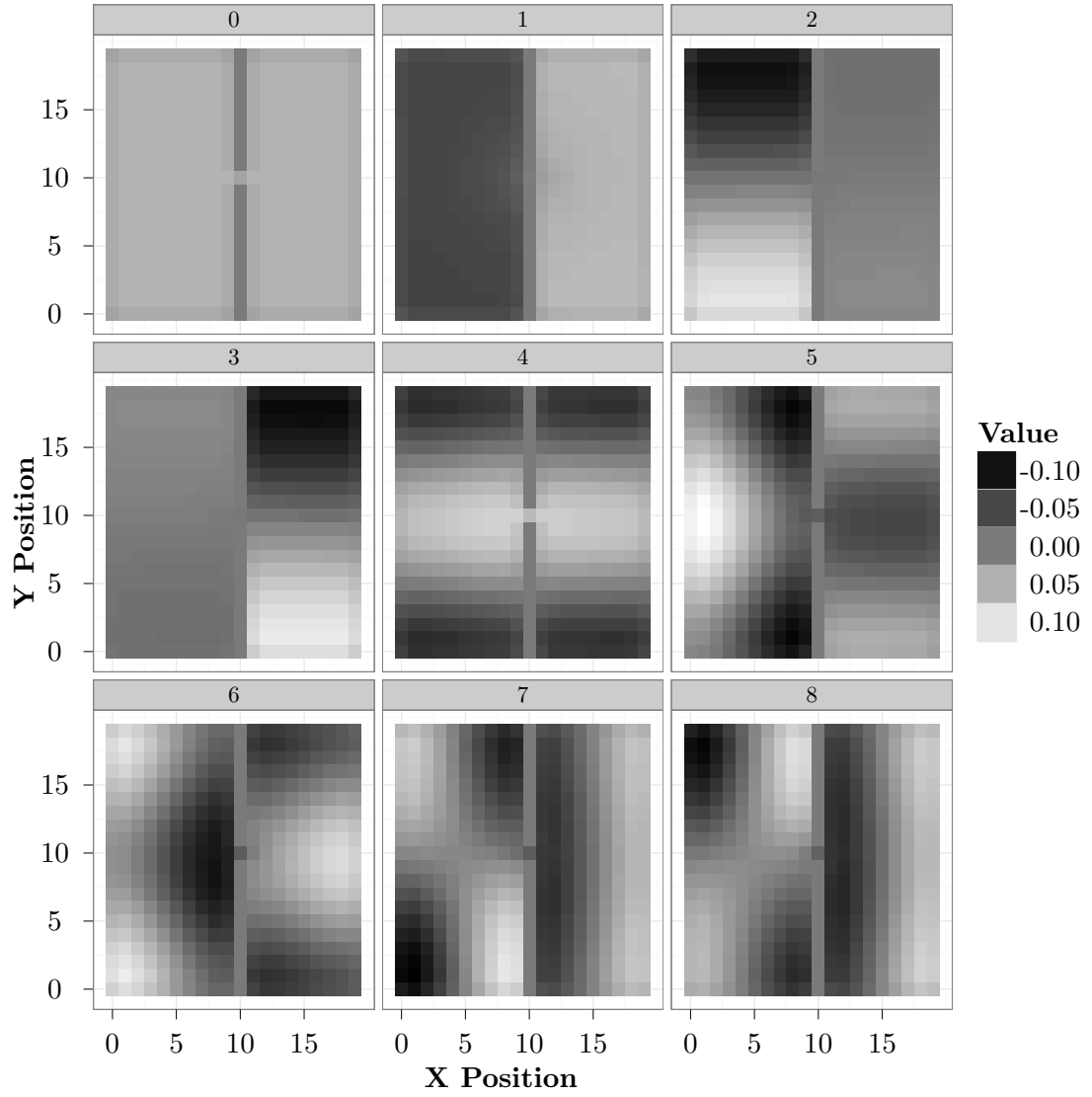

Figure 1: Visualization of the first nine eigenvectors of the graph Laplacian in the two-room domain. A vertical wall separates the two rooms with a single-space door connecting them in the middle. Note that the functions are adapted to the state space connectivity: the second eigenvector separates the 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.

(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.

3.1 Mining the Gameplay Graph

When choosing a graph with which to perform spectral analysis for representation discovery in TTT, the simplest option is the adjacency graph for neighboring board states. We define the *gameplay graph* as having a unit-weight edge between each board and each of the subsequent boards that can be formed by a single move. Every path from the root to a leaf in this gameplay graph represents a complete unique Tic-Tac-Toe game.

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. 2. As in the two-room domain, 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 in forming the adjacency matrix. This simplification is convenient, 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 feature information. The number of game states in most board games, however, increases combinatorially with the size of the board. The set of all possible Go board configurations, for example, becomes intractably large even in small 9-by-9 games. To scale to larger games, we need to construct our graph representation differently.

This work therefore is concerned with the case where we do not have direct access to the full state space, and instead must consider only sample trajectories of games already played. These trajectories helpfully indicate important regions of the game space, but as Fig. 3 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.

Our approach uses a measure of the similarity or "affinity" between arbitrary game states to build an alternative graph representation of a game, one that joins these chains. This representation, which we call 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

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

where δ is the Kronecker delta. An ideal distance measure, however, would be compatible with methods such as k-means clustering, and with the balltree structure [Omohundro, 1989] employed

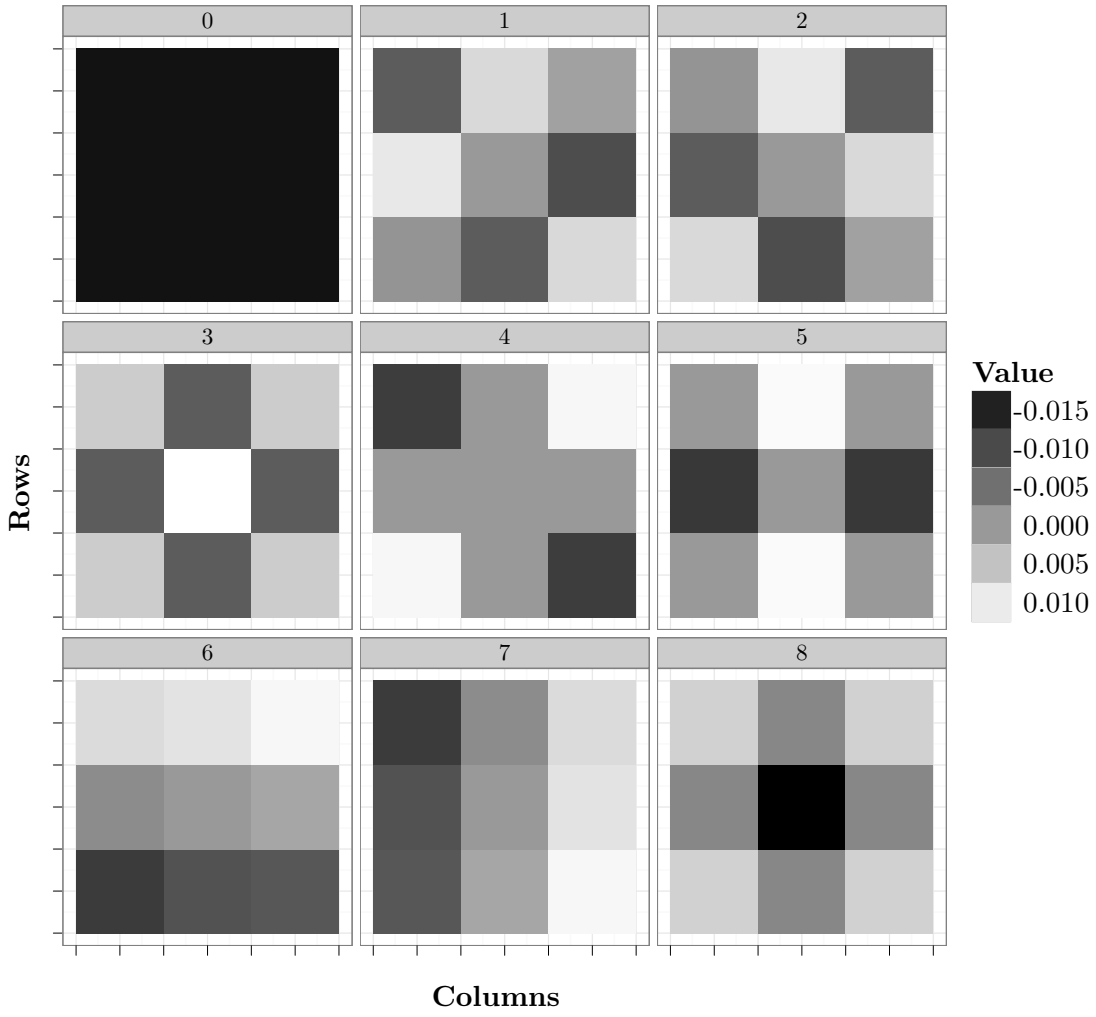

Figure 2: Visualization of the nine smallest eigenvectors of the Laplacian of the TTT gameplay graph. Each grid is associated with an eigenvector, and each cell of the grid is filled according to the value of that eigenvector on the board state reached by the first player placing a mark in that cell 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.

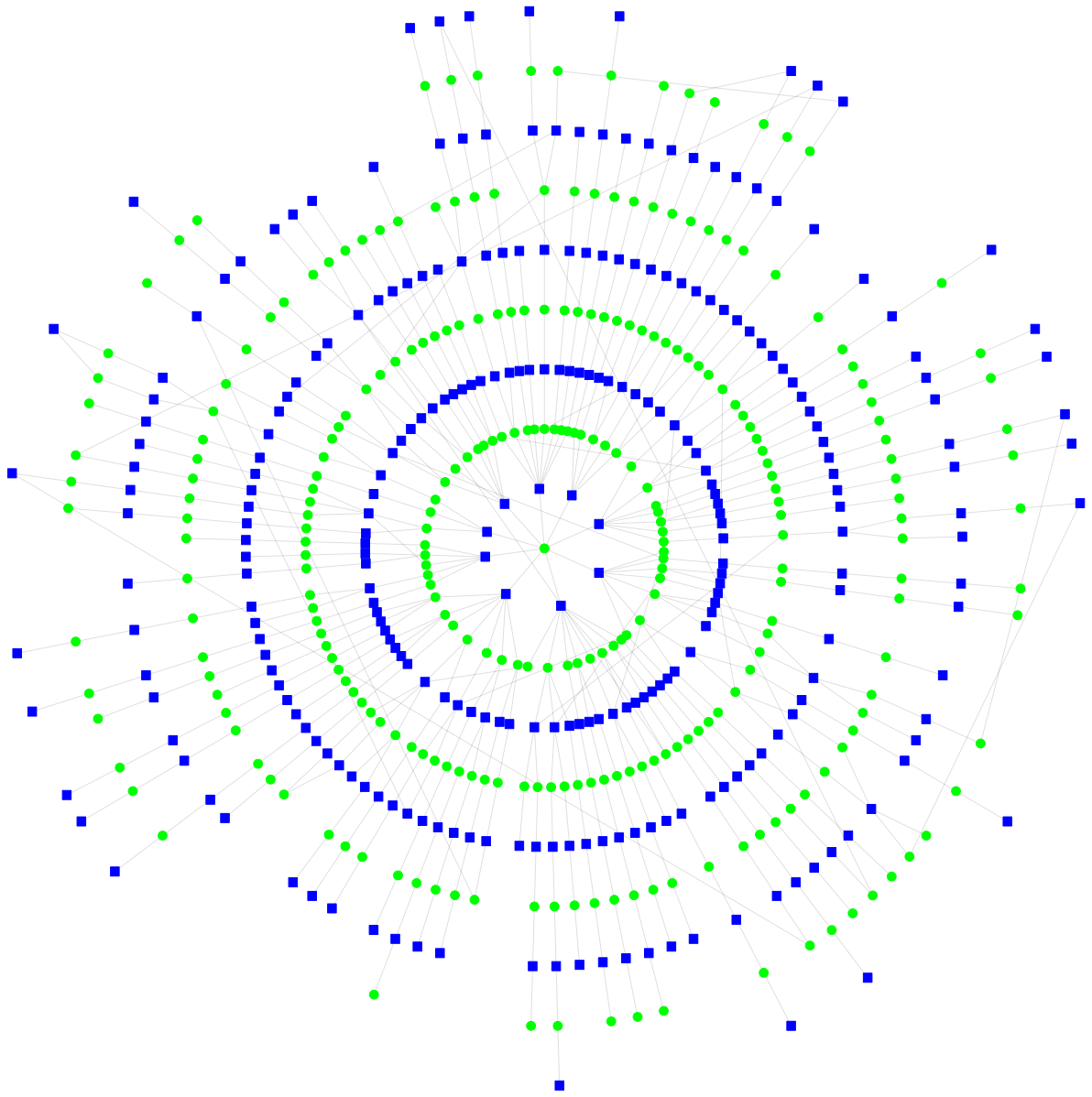

Figure 3: An incomplete TTT gameplay graph constructed from sampled games, i.e., from repeated stochastic walks over the complete gameplay graph discussed in Section 3. Although some edges cross between game trajectories, most are isolated, and the structure of the game is not evident. This problem becomes more pronounced in trajectories sampled from games with higher-dimensional state representations, such as Go. More informed graph construction is required to apply spectral analysis to sampled game trajectories, motivating the use of the affinity graph.

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. 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 Learning to Predict a Value Function

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 uder the optimal policy, $\mathcal{A}_x$ 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 4 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, implemented using minimax search with alpha-beta pruning, using the standard value iteration algorithm of Bellman [1957];
- 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.

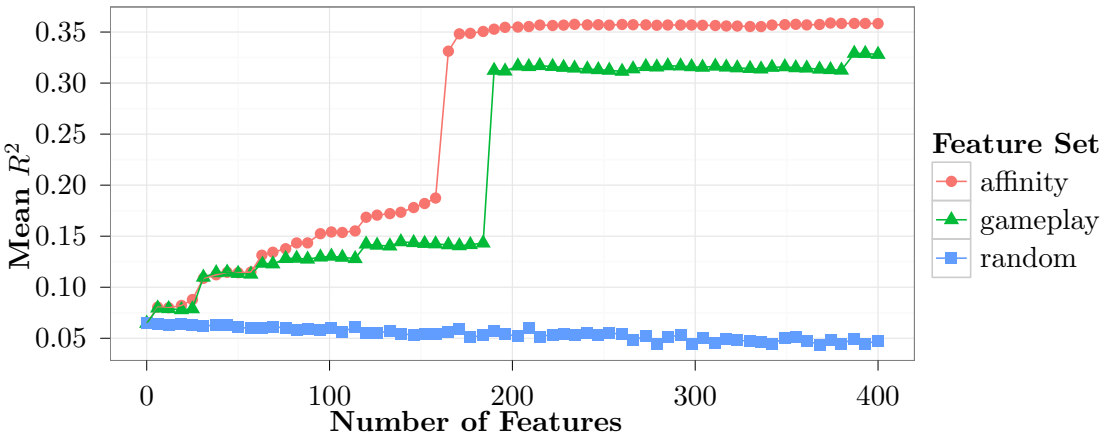

Figure 4: The mean R^2 score of value function prediction versus the number of feature vectors added from the specified feature set, under 10-fold cross validation using ridge regression ($\alpha = 1.0$) in the TTT domain. These features are added to the set of raw, flattened-grid features used to construct 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.

3.4 Learning to Play Tic-Tac-Toe

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 5 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 Scaling to 9x9 Go

Go has long been a challenge domain for AI [Cai and Wunsch, 2007, Bouzy and Cazenave, 2001], with the best computer Go players only recently approaching the professional level. While grandmaster play by computers has been achieved in the challenging games of Chess and Backgammon, the approaches used to address these domains have proven ineffective for Go. The high branching factor of the Go game tree yields classical planning techniques like minimax search intractable. Furthermore, it has proven difficult to hand-craft a good evaluation heuristic for the value of a board as has be done with other board games.

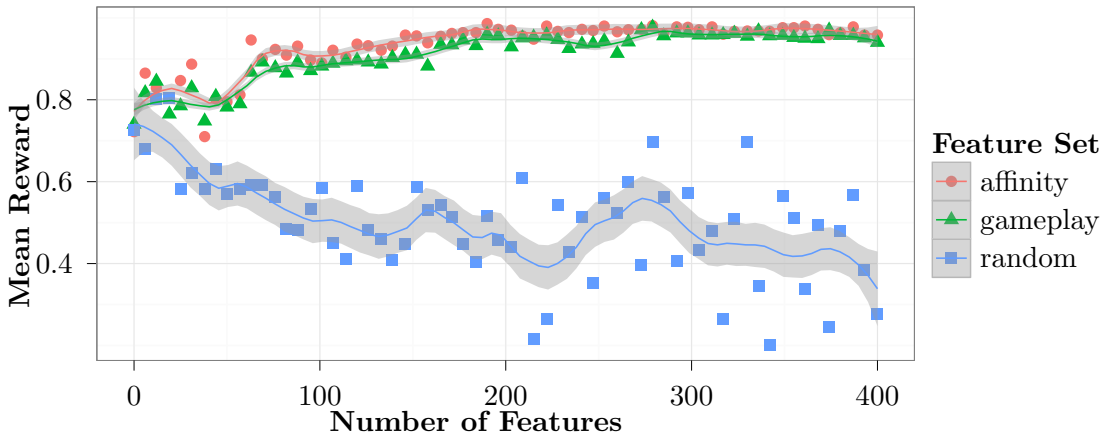

Figure 5: The mean reward obtained by a TTT policy over 1,000 games, averaged under 10-fold cross validation, using linear function approximation with weights tuned via ridge regression as in Section 3.3. Again, the eigenvector features are clearly informative.

Despite its challenges, Go has several appealing properties for studying representation discovery in a large domain. It is a discrete game with simple, deterministic rules, making simulation cheap and effective. Also, the best handcrafted programs have only proven moderately successful; even to domain experts it is not clear what the best features to use are. This makes automatic methods for generating a representation in this domain appealing.

Go programs have become much stronger recently with the introduction of Monte Carlo Tree Search (MCTS) and upper confidence bound (UCB) algorithms to Go [Brügmann, 1993, Gelly and Wang, 2006]. However, these approaches start each game with no domain knowledge and do not learn across games at all.

Attempts to learn a global value function using reinforcement learning have had some success, but the resulting policies do not perform well compared to UCT-based players. In Silver et al. [2007] they enumerate all nxm templates up to size 3x3 and then use Temporal Difference learning and self-play to learn a policy that can beat the Average Liberty Player, a tactical benchmark program. This approach results in around 1.5 million weights and suffers from enumerating too many irrelevant features without being able to represent the global aspects of the board.

This work aims to adress these shortcomings by using spectral techniques for representation discovery, rather than enumeration, to generate useful features for a global value function. To make the strongest Go player, such a value function may be best utilized alongside a UCT-style program as a heuristic for search or to initialize the values of leaves of the MC search tree. Such combinations of online and offline learning for Go are discussed in Gelly and Silver [2007]. In this work, we focus on the feature generation in large domains using 9x9 Go as a test-bed. Improvements for optimizing Go performance are left for future work.

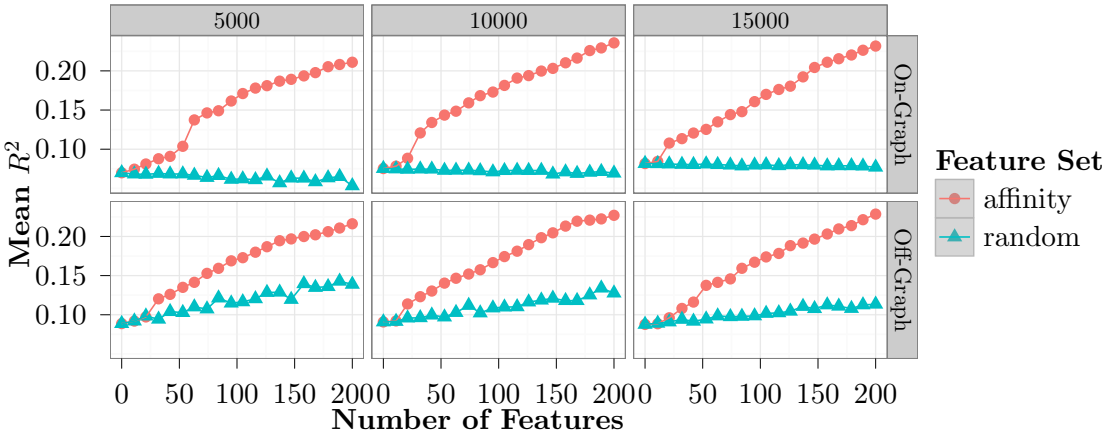

Figure 6: The mean R^2 score of value function prediction versus the number of feature vectors added from the specified feature set, under 10-fold cross validation using ridge regression ($\alpha = 1.0$) in the Go domain. As in TTT, these features are added to the set of raw, flattened-grid features used to construct the affinity graph. Graphs of three different sizes are evaluated, and the test states are drawn either from the set of the graph vertices ("On-Graph") or from a held-out set ("Off-Graph") whose features are then computed by linearly interpolating those of their 8 nearest neighbors. Interpolation has the effect of smoothing the graph, giving random features some minimal ability to generalize, but spectral features are clearly much more useful. The limiting factor appears to be the computational cost of computing additional eigenvectors.

4.1 Experiments

To evaluate the performance of our feature generation approach in the large state space of Go, we ran the same procedure applied to Tic-Tac-Toe to fit the generated features to true board values using ridge regression. As it is intractable to evalutate the true value function, we performed monte-carlo estimates using Fuego [Enzenberger et al., 2010] for the target values. To analyze the effects of sampling on performance, we varied the sample set size used to learn the features. The results for both the on-graph values and the held out off-graph values are shown in Fig. 6.

As with Tic-Tac-Toe, we see that the generated features provide good generalization as compared to the baseline random features. An increase in sample size seems to increase performance up to a point, but the primary constraint appears to be number of eigenvalues that can be computed.

4.2 Scaling Challenges

The most significant of challenge to scaling is the cost of computing the desired number of eigenvectors of the graph Laplacian. As our approach depends on interpolating from known boards to new states, a good deal of coverage (a large sample set) is necessary to span the relevant state space. Simply putting all samples in one large graph would result in an intractable eigenvalue computation for large state spaces. Using an approach similar to Savas and Dhillon [2011], we have

run preliminary experiments in which we cluster the full graph using `graclus` [Dhillon et al., 2007], then find the eigenvectors of the graph Laplacian on each cluster's subgraph.

This approach controls the cost of eigenvector computation on a large graph, but construction of the graph itself becomes the computational bottleneck at large sample sizes. Our balltree implementation, for example, scales poorly to sample sets larger than 10^5. This may be mitigated by clustering in the affinity space directly, using an iterative algorithm like k-means.

5 Future Work

This paper has laid down the outline of a general approach to representation discovery in board games. Many areas of this outline, however, deserve to be further refined and explored. Broadly, three areas of possible future work seem most promising.

First, this paper has evaluated only the most straightforward vector-space mapping for constructing the affinity graph. Using existing, domain-specific heuristic features to measure state similarity could improve the quality of the resulting graph. Similarly, there is an opportunity to use learning to optimize the affinity space mapping.

Second, alternative approaches to scaling could be explored. Whether partitioning should occur on a graph representation or in the initial vector space, for example, is an open question. An efficient approximate method for constructing the nearest-neighbor graph would also be useful. Additionally, the computational and performance tradeoffs between number of clusters, number of eigenvectors solved for per cluster, and sample size should be further explored.

Third, value-function approximation may be most useful working alongside existing effective Monte Carlo methods for Go: the value function can provide high-level strategic insight gleaned from expert examples, as a complement to the precise tactical judgments of MCTS. An interesting avenue for future work is the development of such a hybrid.

Overall, the direction of future work is toward refining the feature discovery pipeline outlined in this paper, and toward maximizing the value of applying it.

6 Conclusion

This work has extended spectral representation discovery to board-game domains, which pose unique challenges, both of kind and of scale, not present in the small spatial domains employed in existing work. Three ideas are proposed: approximating the full gameplay graph using coarse state similarity; operating on partitions of the graph obtained from clustering; and extracting information about the space of relevant games from expert examples. Experiments cover the games of Tic-Tac-Toe and Go. The features learned in these experiments are, for example, sufficiently informative to represent an effective policy for Tic-Tac-Toe, and to substantially improve value function prediction over baseline features in Go. Future work will further develop these methods.

References

R. Bellman. *Dynamic programming.* 1957.

B. Bouzy and T. Cazenave. Computer go: an ai oriented survey. *Artificial Intelligence*, 132(1): 39–103, 2001.

B. Brügmann. Monte carlo go. *Physics Department, Syracuse University, Tech. Rep*, 1993.

X. Cai and D. Wunsch. Computer go: A grand challenge to ai. *Challenges for Computational Intelligence*, pages 443–465, 2007.

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.

I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors : a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

M. Enzenberger, M. Müller, B. Arneson, and R. Segal. Fuego—an open-source framework for board games and Go engine based on Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games (CIG)*, 2010.

S. Gelly and D. Silver. Combining online and offline knowledge in uct. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM, 2007.

S. Gelly and Y. Wang. Exploration exploitation in go: Uct for monte-carlo go. 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.

B. Savas and I. Dhillon. Clustered low rank approximation of graphs in information science applications. In *Proceedings of the SIAM Data Mining Conference. To appear*, 2011.

D. Silver, R. Sutton, and M. Müller. Reinforcement learning of local shape in the game of Go. In *Proceedings of the 20th International Joint Conference on Artifical 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.