Manifold Representations for Value-Function Approximation

Robert Glaubius and **William D. Smart**
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
United States
{rlg1,wds}@cse.wustl.edu

Introduction

Reinforcement learning (RL) has been shown to be an effective paradigm for learning control policies for problems with discrete state spaces. For problems with continuous multi-dimensional state spaces, the results are less compelling. When these state spaces can be effectively discretized, traditional techniques can be applied. However, many interesting problems must be discretized into an infeasibly large number of states. In these cases, other techniques must be used.

Value-function approximation (VFA) addresses some of the problems of traditional RL algorithms, including that of dealing with continuous state spaces. Although it has been successful in a number of specific applications, it has been shown to fail in the general case, and has been known to fail even on simple problems (Boyan & Moore 1995).

We propose a novel approach to value-function approximation based on ideas from topology. We identify a key failing of current techniques, and show how our approach avoids this problem by constructing an explicit model of the state space topology. We begin with a brief description of the problems of current value-function approximation techniques. We then motivate our proposed approach, outline its mathematical foundations, and provide results from initial experiments.

The Problem with VFA

The straightforward application of VFA has been shown to fail in the general case (Boyan & Moore 1995). All commonly used function approximators rely on (weighted) Euclidean distance as a measure of similarity between states. For example, in the space shown in the left of figure 1, this assumption is flawed. Most simple function approximators will tend to generalize across the "hole" in the environment; in most cases this leads to a poor approximation.

Measures of closeness induce a topology on the state space. In particular, distance metrics and a system's dynamics define measures of closeness within the state space. If the measure used by the function approximator results in a significanly different topology than that induced by the system dynamics, we would expect VFA to perform poorly. To address this issue, we explicitly model the state space by constructing a *manifold* representation. A more formal definition of a manifold is given in section . First, we will provide a simple example to illustrate one such conflict.

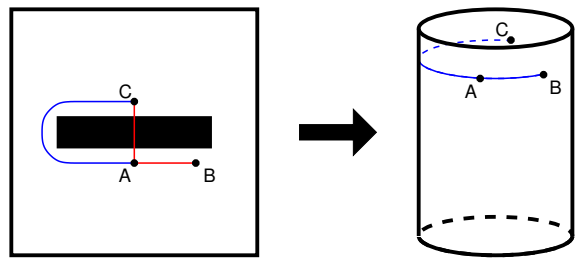

Figure 1: A simple world containing one obstacle (left). The world is topologically cylindrical (right).

Consider the simple world shown on the left of figure 1, a room with a single obstacle in the center. Assuming the normal dynamics associated with a room, i.e. free movement in any direction except through obstacles, point B is intuitively "closer" to A than C is to A. However, the Euclidean distances between A and C and B and C is the same.

One potential model of the state space is the subset of $\mathbb{R}^2$ contained inside the outer wall. Together with the Euclidean distance metric, this world is topologically a disc. However, the manifold that the agent can actually move on – the actual state space of the problem – is a cylinder, as seen in figure 1. The inside wall about the obstacle maps to the top of the cylinder; the outside wall of the world maps to the cylinder's base. Using the Euclidean distance metric on the surface of this cylinder will provide a more reasonable measure of locality. The distance on the cylinder from A to C (blue line) roughly corresponds to the blue line on the left side of the figure.

For this simple example, it is clear what the best model of the state space is. However, in general we only have access to the dynamics of a particular problem through observed transitions. Therefore, we focus on constructing manifold models of the state space by sampling the transition function.

Manifolds

A manifold is a Hausdorff space that is locally Euclidean, i.e., each point in space has a neighborhood where Euclidean distance reflects the true distance between points. A differential manifold M is a manifold together with an *atlas*, which is a set of charts. Each chart ϕ is a homeomorphism from a subset of M to $\mathbb{R}^n$, where n is the dimension of M. Further, every point on M is in the domain of some chart.[1] We use the term "manifold" in place of "differential manifold" throughout this paper.

A global function $F : M \to \mathbb{R}$ can be embedded on a manifold M as follows. Let Λ be an indexing set, then for each chart $\phi_\lambda : U_\lambda \to \mathbb{R}^n$ on M, where U_λ is a subset of M, we define two functions, $f_\lambda : M \to \mathbb{R}$ and $b_\lambda : M \to [0, 1]$. We will delay discussion of f_λ for a bit; the blend function b_λ must be a smooth function with infinite derivatives, all of which are 0 at the boundary of U_λ. b_λ is defined to be 0 everywhere outside of U_λ. The global function F is defined as

$$F(p) = \frac{\sum_{\lambda \in A} b_\lambda(p) f_\lambda(p)}{\sum_{\lambda \in A} b_\lambda(p)} \tag{1}$$

for all points p in M. A more comprehensive description of manifolds can be found in variety of textbooks (Lefschetz 1949).

Manifolds for Value-Function Approximation

The basic strategy employed in our work is to model the state space as a set of simpler regions; these are our charts. Consider the world shown in figure 2. There are an infinite number of atlases for this example; any set of charts that completely covers the reachable part of the state space is acceptable. One possible atlas is shown in figure 2. Notice that charts on opposite sides of the obstacle do not overlap. Each chart is convex and contains no obstacles, so Euclidean distance should be a reasonable measure of locality within the chart. This means that straightforward VFA techniques ought to work well within each chart.

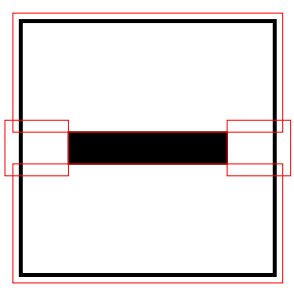

Figure 2: One possible chart allocation for a simple world with one obstacle.

Once we have constructed a model of the state space, we embed a global function approximator on this model. To achieve this, we embed a local function on each chart; these correspond to the functions f_λ in equation 1. Following Grimm and Hughes (Grimm & Hughes 1995), we use a one-dimensional piecewise polynomial spline as the blend function b_λ. The value of the function approximator at a point is computed as in 1. The use of the blend function allows us to make some guarantees about the resulting global function. Building a global function from this manifold, under certain easily-satisfiable conditions (Grimm & Hughes 1995), results in a C^k (smooth and continuous up to the kth derivative) manifold surface. The value of the global function will be locally bounded by the values of the embedding functions as well, since the blend functions are a partition of unity.

In this work we approximate the state-action value function using one embedding function for each available action. Updating the value of a state corresponds to presenting a training experience to the function approximator embedding on each chart containing that state, using the standard Q-learning iterative update (Watkins & Dayan 1992). In all other ways, we can treat the set of embedding functions as a drop-in replacement for the tabular representation used by standard RL algorithms.

First, however, we show how manifold representations of the value function relate to currently successful VFA techniques.

Connections to Other VFA Techniques

Manifold representations provide a unifying framework for a number of other empirically successful VFA techniques. In this section, we identify these techniques and briefly describe how they can be represented by a manifold.

A common VFA approach is to break the state space into a set of discrete states, represent the values of these states in a table, and apply traditional RL techniques (Moore & Atkeson 1995; Uther & Veloso 1998). Others use more advanced function approximation techniques within the cells (Munos & Moore 1999). These can be modeled by allocating one chart per aggregated state, and embedding an appropriate function approximator in it. These charts will have no overlap, which breaks the strict definition of a manifold. However, this can be easily overcome by growing all charts by a small amount, so that each overlaps its neighbors by an infinitesimal amount.

One of the most empirically successful function approximators for VFA is Albus' CMAC (Albus 1975). Interestingly, a CMAC is exactly a manifold. The CMAC tiles the state space with overlapping tiles, each of which has a constant value assigned to it. The prediction for a point, p, is the sum of the values of all of the tiles in which p falls. This sum is often weighted by the distance of p from the center of the tile. Using the language of manifolds, the tiles are charts, the value is the embedding function, and the weights are the blend functions.

[1] In this paper we sometimes refer to the domain of a chart as a chart; the intent should be clear from context.

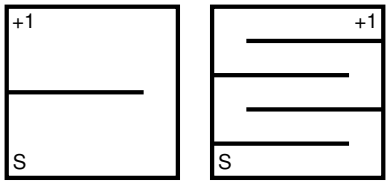

Figure 3: The WallWorld test domain.

Chart Allocation

There are many possible chart allocations for any given domain; however, some of these will serve our purposes better than others. Computation time scales with the number of charts, so we prefer allocations with fewer charts. However, we expect that larger charts will require more complex embedding functions in general, so there is a tradeoff between chart size and approximator complexity.

In this section, we present three chart allocation schemes, and demonstrate their behavior on two sample problems. We simplify matters by considering only approximations of the state-action value function, which can be calculated using an off-policy algorithm. Furthermore, we assume that we can arbitrarily sample states and actions, rather than being constrained to trajectories through the space. Finally, we consider only problems with a single non-zero reward.

The Test Domains

We use two test domains in this work, WallWorld and the standard mountain car domain. WallWorld is shown in figure 3. The agent starts in the lower left of the world, and can move a fixed step size in the four cardinal directions. The goal is to get to the upper left or right, depending on whether there are an odd or even number of walls, respectively. A reward of 1 is given at the goal, with 0 reward everywhere else.

The mountain car domain (Moore & Atkeson 1995) simulates an under-powered car learning to drive up a hill. The reinforcement learning agent may drive forward, backward, or may choose to coast. A reward of 1 is given when at the top of the hill; 0 reward is given everywhere else.

Random Allocation

The simplest allocation scheme we experiment with is random allocation. Random uncovered points are covered with a fixed-size axis-aligned rectangular chart until no point in the state space is uncovered. It seems reasonable to expect that such a simple strategy would not perform well, but as we will show in the next section, this scheme can work quite well in some domains.

Random Walk

Our second allocation scheme uses random walks. From a random state, several n-step random walks are performed with actions chosen uniformly at random. The smallest bounding box enclosing these walks becomes the domain of a chart. We repeat this process until the domain is covered. Figure 4 shows some sample allocations for our two example domains.

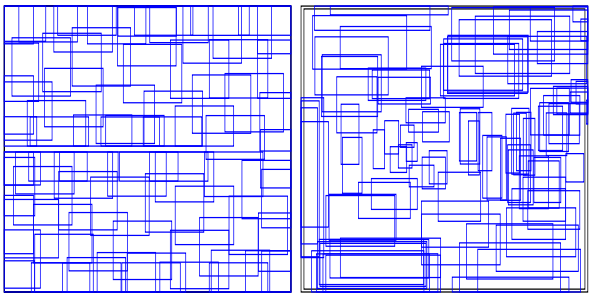

Figure 4: Random walk chart allocations for WallWorld (left) and mountain car (right). Only some of the allocated charts are shown for mountain car.

Growing Charts

Our final strategy is based on the likelihood of an action carrying the agent a relatively long distance. Let r be a positive real number and s be a state, then P_{out} is the probability that an arbitrary action will have result in a new state with distance at least r from s in the state space.

We apply this statistic in a general framework for "growing" charts. Basically, we choose a point that is not yet covered and cover it with a small initial chart. Regions of the state space adjacent to the initial chart are merged with it if the distribution of some random variable across each is not significantly different. This is repeated until no new region can be added. New points and charts are allocated until the entirety of the state space is covered.

We use the P_{out} statistic to implement this strategy with axis-aligned rectangular charts. An uncovered point is selected at random and covered with a small chart. Sides of this chart are selected uniformly at random and extended by an increment of δ if the distribution of sampled P_{out} values does not significantly differ from the distribution within the chart itself. This process is repeated until every side has failed to grow once. We continue to allocate new charts until the state space is covered

This measure is well-suited to detecting changes in system dynamics that alter the relative magnitude of state transitions. For instance, we would expect a significant difference when sampling a region containing a wall versus a region with no obstacles. figure 5

Experimental Results

Our initial experiments focus on the quality of the policy learned by Q-learning using a manifold value-function approximator. Three chart allocation schemes (random, random walk, and growth-based) were used, in combination with two simple function approximators

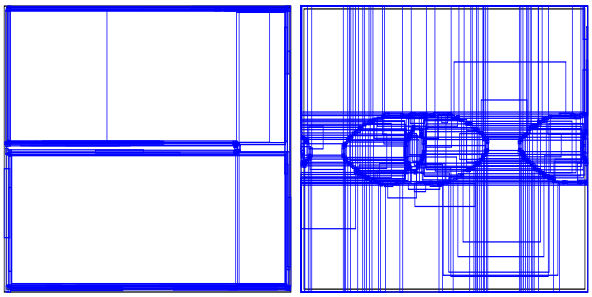

Figure 5: Growth-based allocations for Wallworld (left) and mountain car (right).

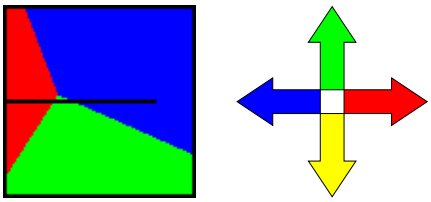

Figure 6: WallWorld learned policy with a global linear function approximator. Colors correspond to movements in the four cardinal directions.

(constant and linear) on WallWorld and the mountain car worlds. The results of these experiments are presented in this section.

In all experiments, the VFA was trained with one million state action pairs selected at random. The learning rate α was set at 1, and a discount factor of $\gamma = 0.99$ was used. The quality of the learned policy was evaluated by executing the greedy policy from 1,000 different starting points.

We present here the learned policies, rather than a performace metric, such as percentage of successful runs. We have found that a poor policy close to the goal state will cause many evaluation runs to fail, even though the policy is correct in the majority of the state space. Empirically, we have found that the policy is often poor very close to the goal state, but good everywhere else. We have no concrete explanation of this phenomenon at present.

WallWorld

Figure 6 shows the policy learned when a global linear function approximator is used. One plane is learned for each of the four actions. In this example, only three of the planes intersect, while the fourth (corresponding to the down action) consistently has lower value than the other three. This policy performs poorly, and generalizes across the obstacle in the domain.

Figure 7 shows the policies learned with random chart allocation and constant function approximation. These policies are quite poor in most parts of the state space. Failure to account for system dynamics results in contiguous areas of similar action across the wall obstacle.

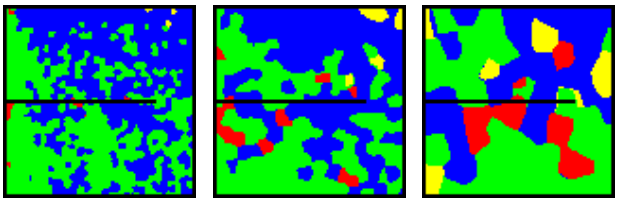

Figure 7: WallWorld learned policies with random chart allocation and constant function approximation. Chart sizes are 1% (left), 10% (middle), and 20% (right) of domain width.

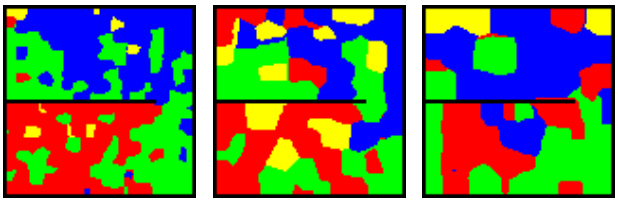

Figure 8: WallWorld learned policies with random walk chart allocation and constant function approximation. Random walk lengths are 1 (left), 2 (middle), and 3 (right) steps.

As expected, as the chart size increases, the amount of generalization across the wall does as well.

Figure 8 shows the results of learning with the random walk allocation scheme using walks of lengths of one, two, and three steps. 1-step charts result in the best policy, which is actually optimal. This is expected, since charts allocated using this strategy can only overlap the ends of walls. Since constant function approximation commits to a single optimal action in a chart (excluding overlap regions), large charts commit to large areas of the same action in the policy. As a result, the learned policy degrades as the walk length increases.

Switching to linear function approximation (figure 9) alleviates this problem. All policies shown are near optimal. This illustrates the trade-off between chart size and function approximator complexity. Larger charts generally need more complex embedding functions, while smaller charts can utilize simpler schemes.

Our third chart allocation scheme is shown in fig-

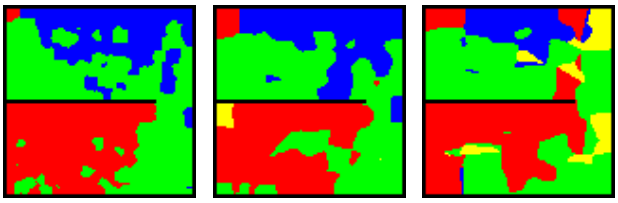

Figure 9: WallWorld learned policies with random walk chart allocation and linear function approximation for walks of length 1 (left), 2 (middle), and 3 (right) steps.

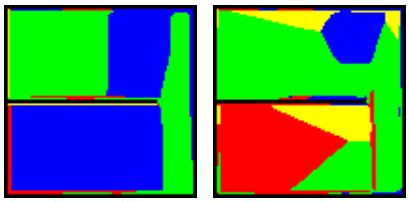

Figure 10: WallWorld learned policies with growth-based chart allocation, with constant (left) and linear (right) function approximation.

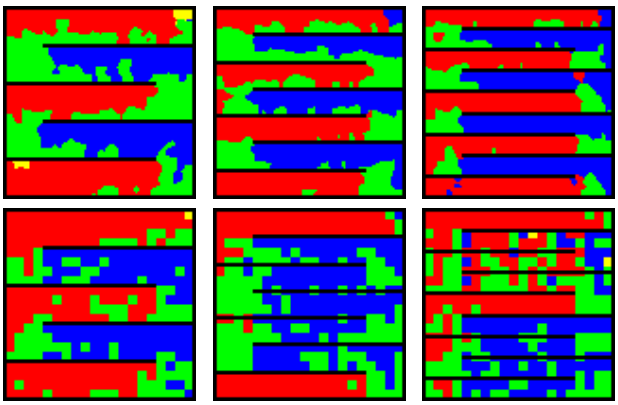

Figure 11: WallWorld learned policies. Top: 1-step random walk chart allocation, and linear function approximation, with 4 (left), 6 (middle), and 8 (right) walls. Bottom: Tabular Q-learning with 400 states.

ure 10. The scheme favors large charts, and performs poorly in WallWorld. Linear function approximation is somewhat better, but a more complex function approximator is probably necessary.

It is reasonable to ask these approximation schemes compare to standard techniques, such as tabular Q-learning with a uniform state space discretization. Figure 11 compares the performance of the 1-step random walk allocator with linear VFA to a tabular approach for three WallWorld instances. Although they perform similarly for four and six wall environments, the manifold representation is clearly better in the 8-wall case. This illustrates one strength of our approach, that the system is self-tuning, rather than requiring discretization to be tuned by hand.

Mountain Car

We also performed experiments with the well-known mountain car domain. Three actions (left, coast, right) are available, and a single reward is given at the top of the hill regardless of velocity. We report the percentage of 1000 evaluation runs that reach the goal in less than 500 steps when started from random positions. A finely discretized tabular representation achieves a performance of 84.7% successful runs after one million training points.

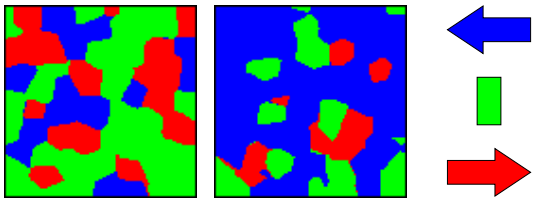

Figure 12: Mountain car learned policies with random chart allocation, and constant (left) or linear (right) function approximation.

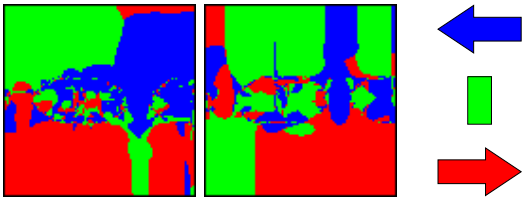

Figure 13: Mountain car learned policies with growth-based chart allocation, and constant (left) or linear (right) function approximation.

Our initial results, although preliminary, are promising. For growth-based allocations, constant function approximation resulted in a success rate of 17.9%, while linear approximation led to a rate of 92.4%.

Figure 12 shows the results of random chart allocation. For constant function approximation, 21.1% of runs were successful, while linear function approximation resulted in 94.3% success. The policy learned with linear embedding functions gets to the goal nearly every time. However, runs from a particular state have a lower discounted total reward than those generated by the tabular policy, as the linear policy is not quite as good.

Figure 13 shows the results for growth-based chart allocation. We do not report results for random walk allocations, as this method generated an infeasibly large number of charts even for relatively long walks.

Current Work

The problem of how best to allocate charts to cover teh domain is the focus of our ongoing work on this problem. In particular, we are working on methods that are suited to more complex domains than those presented in this paper. When the restrictions of a sparse reward function and insistence on state-action value approximation are relaxed, we expect the chart allocation problem to become more difficult. We are also currently investigating the effectiveness of other function approximators for embedding on charts.

References

Albus, J. S. 1975. A new approach to manipulator control: The cerebellar model articulation controller

(CMAC). *Journal of Dynamic Systems, Measurement and Control* 220–227.

Boyan, J. A., and Moore, A. W. 1995. Generalization in reinforcement learning: Safely approximating the value function. In Tesauro, G.; Touretzky, D. S.; and Leen, T., eds., *Advances in Neural Information Processing Systems*, volume 7, 369–376. MIT Press.

Grimm, C. M., and Hughes, J. F. 1995. Modeling surfaces of arbitrary topology using manifolds. *Computer Graphics* 29(2). Proceedings of SIGGRAPH '95.

Lefschetz, S. 1949. *Introduction to Topology*. Princeton University Press.

Moore, A. W., and Atkeson, C. G. 1995. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning* 21:199–234.

Munos, R., and Moore, A. W. 1999. Variable resolution discretization in optimal control. *Machine Learning*.

Uther, W. T. B., and Veloso, M. M. 1998. Tree based discretization for continuous state space reinforcement learning. In *AAAI/IAAI*, 769–774.

Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine Learning* 8:279–292.