MDPs in Continuous Space

Bharat Chandar

September 12, 2015

Abstract

This paper explores extensions of methods to solve Markov Decision Processes to continuous space. I apply a sample-based approach to estimate the value and policy functions for each point in the space. I consider primarily instances in which the feature vector used in the approximation of the value function is drawn randomly at each point. I briefly compare this to cases where the feature vector is composed of deterministic variables.

1 Introduction

Markov Decision Processes are used in artificial intelligence to solve for the optimal action for the agent to take in contexts where each state is associated with some reward and movements are noisy. In finite discrete space, solving for the value of each state and its associated optimal policy can be done using value or policy iteration. These algorithms are guaranteed to yield the optimal functions in some finite amount of time.

The use of value iteration in continuous space presents several challenges. First, there are an infinite number of possible actions at each state. Second, the noise around an action is also in continuous space. These two conditions makes maximizing over all possible actions at a state potentially challenging if the objective function is some non-smooth function. Finally, and most importantly, the value cannot be evaluated at every point in the space since the number of states is infinite. For the same reason it is impossible to evaluate the current value function at each point in the set of post-action states (the post-action state is the state the agent lies in after choosing some action). This makes use of the Bellman-style iterative optimization methods used in finite discrete space unfeasible without some modification.

I employ a sample-based approach to address such issues. Each point in the space becomes associated with some feature vector. I primarily consider cases where this feature vector is a random variable correlated with the location in the space. The value function is estimated by interpolating the value on the feature

vector over points in a sample of the space. Samples are also used to choose possible actions and post-action states.

I explore difficulties encountered by using such a sample-based approach. Particularly, there can be considerable bias and variance in the value and policy functions. This results from the uncertainty in the samples and from the error in interpolation. Consequently it becomes difficult to evaluate convergence of the value function. There are no guarantees for monotonically decreasing mean squared error in the value function over points in the sample. Thus the optimal value and policy functions cannot be determined exactly as they can in finite discrete space.

Finally, I compare the relative performance when using a deterministic feature vector instead of a random one. Using deterministic features tends to lead to more accurate policy choices. Random feature vectors might be more realistic when deterministic features are not observable at every point. For instance, perhaps the distance or orientation to the closest terminal subspace cannot be found at every point, but there are other landmarks that are correlated with the subspace. As a toy example, the terminal subspace may be located on a body of water, and states near it might have more fertile soil. The fertility of the soil could be a random feature associated with a point in space that is correlated with value.

2 Methodology

I now proceed to describe the methodology in greater mathematical detail.

2.1 Build the state space

I consider a $1x1$ space of points.

2.2 Assign terminal subspaces

I create some number of terminal subspaces within the space, each associated with some reward or penalty. These subspaces are non-overlapping. I apply several simplifying assumptions here, though less restrictive assumptions conceivably would result in little loss in generality. First, I make all of the terminal subspaces circular. This is natural given the method I use for generating random features, which will be discussed subsequently. Second, I make all of the terminal subspaces the same size. Lastly, I assign all "good" subspaces a reward of 10 and all "bad" subspaces a value of -10. Non-terminal states have a value of 0.

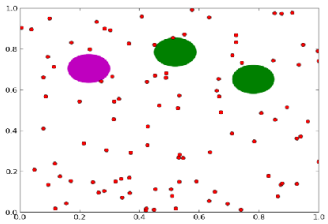


Figure 1: This plot shows the space considered in my continuous MDP model. Green circles are terminal subspaces with positive reward, while purple circles have negative reward. The red dots are sampled points in the space.

2.3 Sample points in the space

I draw N points $\{S^{(1)}, ..., S^{(N)}\}$ uniformly over the non-terminal portions of the space. Figure 1 shows the space after terminal subspaces are chosen and the sample is drawn.

2.4 Associate each point with a terminal subspace

I associate each point with a terminal subspace. This aids in drawing feature vectors that are correlated with the distance to a subspace.

I consider a framework where each sampled point $S^{(i)}$ is viewed as being drawn from a mixture of multivariate normal distributions centered at the midpoints of the terminal subspaces Y_k. The probability that $S^{(i)}$ becomes associated with a given terminal subspace is computed as follows:

$$P(Y_k|S^{(i)}) = \frac{P(S^{(i)}|Y_k)P(Y_k)}{\sum_{k'} P(S^{(i)}|Y_{k'})P(Y_{k'})}$$

Since each of the terminal subspaces were assumed to be the same size, this reduces to

$$P(Y_k|S^{(i)}) = \frac{P(S^{(i)}|Y_k)}{\sum_{k'} P(S^{(i)}|Y_{k'})}$$

$P(S^{(i)}|Y_k)$ is assumed to be bivariate Gaussian with mean μ_k and covariance matrix $\sigma_s^2 I_{2x2}$. Modeling the conditional distribution of the state given Y_k as an isotropic Gaussian makes it natural to construct each terminal subspace as circular.

Note that the chance a state $S^{(i)}$ will get associated with some terminal subspace is proportional to its distance from the center of that subspace.

2.5 Draw Feature Vectors

I next draw some d-dimensional feature vector $\theta^{(i)}$ for each sample state $S^{(i)}$. Let $E[\theta|Y_k] = \theta_k$ and $E[\S^{(i)}|Y_k] = \mu_k$. The distribution of $\theta|S^{(i)}, Y^{(i)}$ can be found by augmenting $P(S^{(i)}|Y^{(i)})$ and computing the conditional distribution. Let $Cov((S, \theta))$ be some covariance matrix common across each Y_k,

$$Cov((S,\theta)) = \begin{pmatrix} V_s & A' \\ A & V_\theta \end{pmatrix}$$

where V_s is the covariance matrix for S, V_θ is the covariance matrix for θ, and A is $Cov(S, \theta)$. The conditional distribution of $\theta|S = s, Y = k$ is

$$N(\theta_k + AV_s^{-1}(s - \mu_k), V_\theta - AV_s^{-1}A')$$

Modeling the feature vector this way makes it correlated with proximity to a terminal subspace. Note that there is considerable randomness in $\theta^{(i)}$, arising both from the randomness in $Y^{(i)}$ and in the variance of $\theta|S, Y$.

2.6 Initialize $V_0(S, \theta) = \vec{0}$

V is the value function.

2.7 Run Value Iteration

Run value iteration according to the following algorithm:

Algorithm 1 Value iteration in continuous space

for t = 1...T **do**
  Set $V_t(s, \theta)$ = Regress V on (s, θ)
  for i = 1...N **do**
    $$V_t^{(i)} = \max_a \frac{\sum_{s'} P(s'|s^{(i)},a)[R(s')+\gamma V_t(s',\theta')]}{\sum_{s'} P(s'|s^{(i)},a)}$$
  end for
end for
Set $\pi_{V_t}(s) = \arg\max_a \frac{\sum_{s'} P(s'|s,a)[R(s')+\gamma V_t(s',\theta')]}{\sum_{s'} P(s'|s,a)}$
return V_T, π_{V_T}

Here $V_t(s, \theta)$ is the interpolated value function from $\mathbb{R}^{d+2}$ to $\mathbb{R}$, while $V_t^{(i)}$ is the value at state $S^{(i)}$ in the sample. $1 - \gamma$ is the discount rate.

This algorithm resembles the finite discrete space algorithm, but there are some important distinctions. The set of actions over which the expected value is maximized is not the full set of possible actions. I consider a setting in which for any action the mean distance traveled, δ, is the same. However, the agent can proceed in any direction from their current state. Then the set of possible mean post-action states is the edge of the circle of radius δ around the current state. Since the value cannot be maximized over this entire continuous set, I instead sample uniformly from the circle of possible mean post-action states.

I allow for noise in the motion model. While the post-action state has mean on the boundary of the circle, the final state is drawn from an isotropic Gaussian centered around the mean, that is, the post-action state is distributed $N(s + \vec{a}, \sigma_\delta^2 I_{2x2})$ where s is the current state and a is the action with magnitude δ in the direction of motion. The parameter σ_δ^2 controls the amount of noise in the model. This is demonstrated in figure 2. The set of possible post-action states corresponding to a given action is then sampled from this Gaussian distribution. The value corresponding to an action is taken to be the sample mean value over the post-action states. The policy and value for the starting state corresponds to the maximum expected value over the sampled actions.

The feature vector θ' at the post-action state is drawn the same way as was done for the original sample of states. First the post-action state gets associated with a component in the mixture model. The feature vector is then drawn by augmenting the mixture. The value at each possible post-action state is calculated using the interpolated value function from the previous iteration. I examine both linear regression and random forest regression in interpolating over the value function.

An important distinction between finite discrete space framework and this continuous space analog is convergence criteria in the value function. In finite discrete space, the mean squared error in the value at each state is monotonically decreasing across iterations. This is not the case in continuous space. Figure 3 shows a plot of the sum of squared error in the value of points between time t and time $t+1$. Note the spikes in squared error at certain time steps. This makes it difficult to determine appropriate convergence criteria. The spikes are because of variance in the interpolation, the feature vector, the sample of actions, and the sample of post-action states. For the moment I fix T to be some constant. Choice of proper convergence criteria could be an area for further research.

The algorithm returns the value at each sample point along with the policy vector (the action that maximizes the expected value) for each point.

Section 4 discusses results of value iteration for various parameter settings.

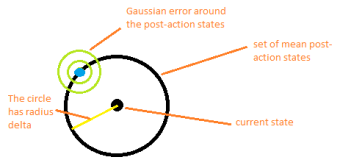


Figure 2: This plot shows the plot of possible actions around the current state, along with the noise in motion.

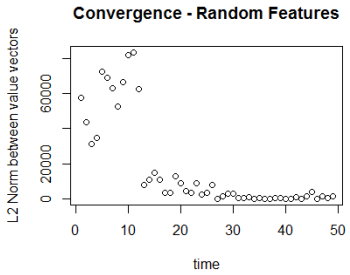


Figure 3: This shows the rate of convergence in the value function. Here the parameters are the same as in section 4.5.

3 Sample-based path planning

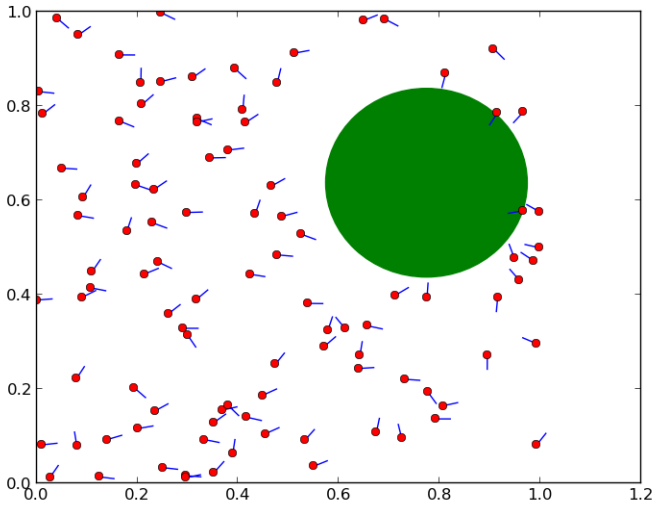
I consider using the policy function to solve for the optimal path from any point in the space to a terminal subspace. This would be inappropriate when γ is small since if the future is heavily discounted the goal may often not be to reach a good terminal subspace. However, it could be a reasonable method for motion planning if the future is not discounted very much.

I consider two methods for solving for the path. The first is to simply follow the action of the closest sampled point. If the size of the sample is sufficiently large, this should be a reasonable approximation. The second method is to find the best action at each point using the same optimization used in value iteration as described above. At each step in the path, a sample of actions are drawn from a circle of radius δ, and the action the leads to the maximum empirical expected value is chosen.

Both methods seem to yield reasonable paths, though they are somewhat jagged.

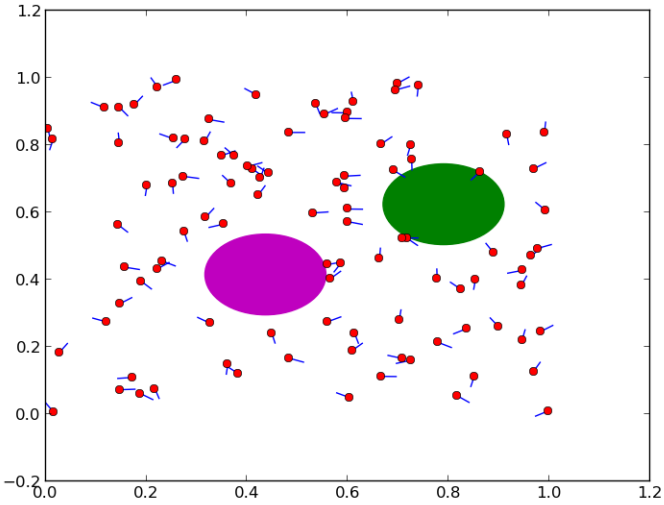
4 Results

4.1 One good subspace



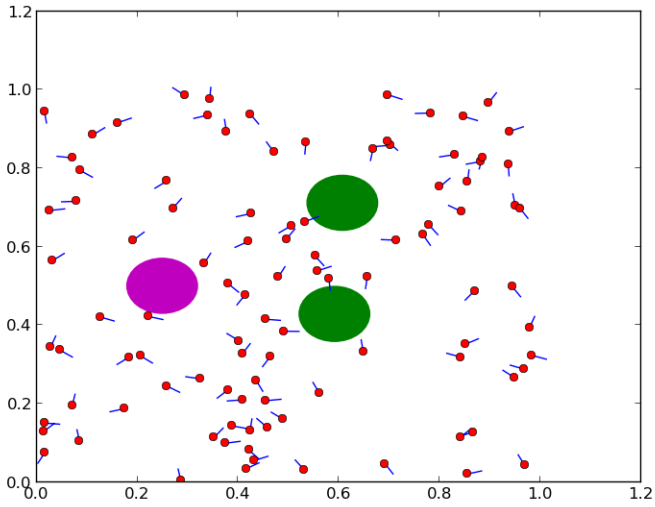
Above is a plot of the policy at each sample point when there is only one good terminal subspace. The blue lines represent the optimal action vector for each sample point. Note that in this simple example γ is 0.10, δ is .03, σ_δ^2 is .001, and the interpolation method is linear regression. Though there is quite a bit of noise, the action vectors generally point in the direction of the terminal subspace.

4.2 One good and one bad subspace



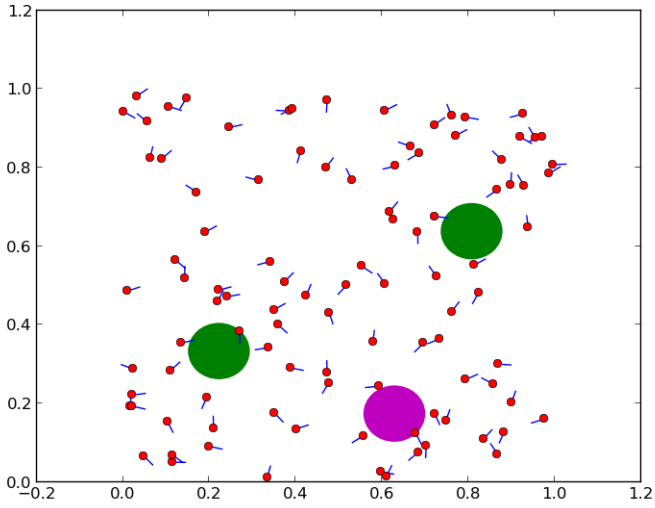
In this example, there is both a terminal subspace with positive reward and a terminal subspace with negative reward. The parameters are otherwise identical to those in section 4.1. Though it appears that states near the good terminal subspace generally point towards its center and states near the bad subspace seem to point away from its center, the action vectors appear much noisier. Introducing another terminal subspace to the model seems to complicate value iteration.

4.3 Three subspaces



This plot shows one negative terminal subspace and two positive subspaces. Many of the action vectors seem to point in directions that are roughly random.

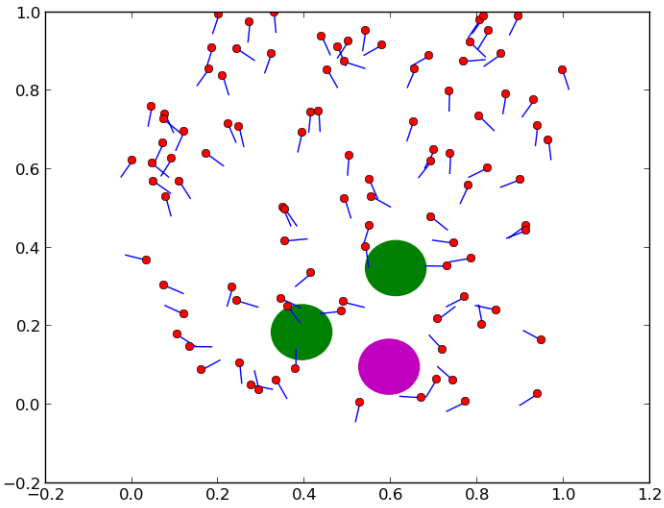
4.4 Random Forest Interpolation



In this plot, the interpolation method is random regression where the maximum depth is product of d

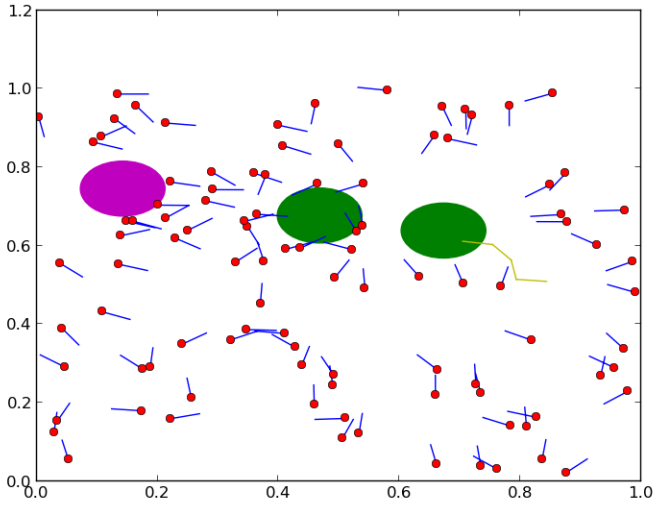
and the number of terminal subspaces. The parameters are otherwise the same. In this case there seems to be little improvement in the performance of value iteration compared to linear regression. However, it is worth noting that random forests are of course much more robust to nonlinearities in the relationship between the value and the features. In this case, the feature vectors associated with each terminal subspace are nondecreasing in the reward for each subspace, so there is little improvement in switching to a random forest regression.

4.5 Changing δ and γ



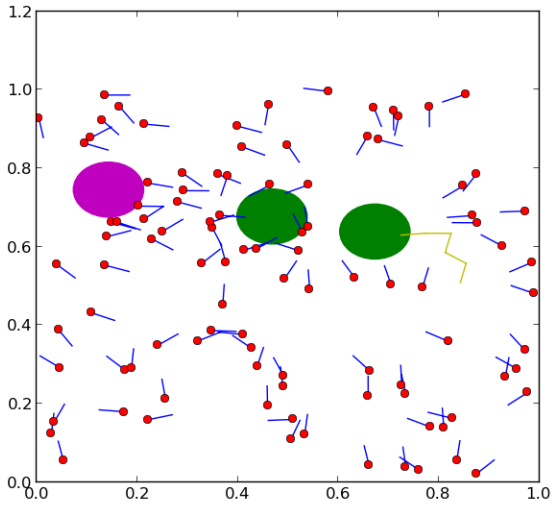
Here the noise is greatly reduced and the the future discounted less: $\sigma_\delta^2 = 10^{-8}$ and $\gamma = .9$. The action vectors seem to point more cohesively towards the positive states and away from the negative states. Since γ is small actions may be more likely to aim towards a positive terminal subspace. There should also be lower variance in the direction the arrows point since the noise is smaller.

4.6 Finding an optimal path



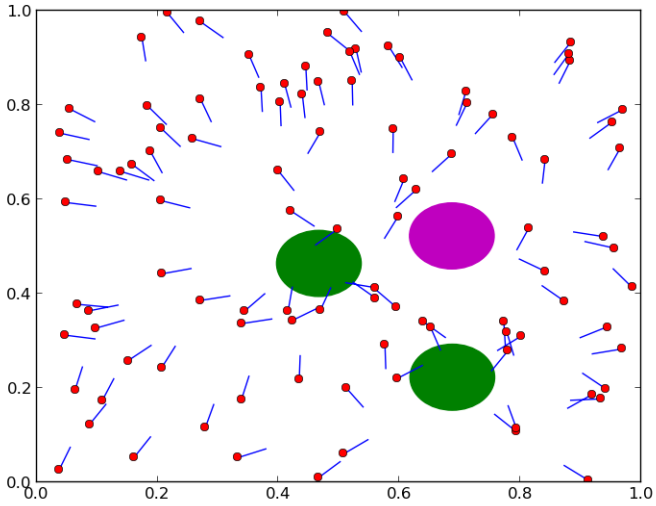
In this example a random point is drawn from the space and a path to a terminal subspace is constructed by finding the optimal action. Note that the parameters are identical to the preceding example except value iteration is run for a longer time.

4.7 Following the closest sample point



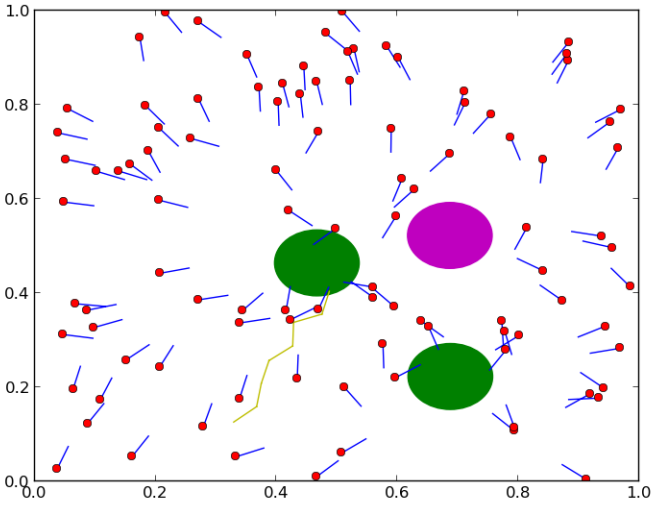
In this case the path is found by following the action of the closest node. The path is noticeably more jagged.

4.8 Deterministic feature vectors



The following two examples examine cases with deterministic feature vectors instead of random vectors. Each point is associated with four features: distance to the closest good subspace, distance to the closest bad subspace, orientation to the closest good subspace, and orientation to the closest bad subspace. Distance is measured using the $l2$ norm and orientation is the angle between the center of the subspace and the state relative to the positive horizontal axis. Nearly all of the actions clearly point towards a positive terminal subspace.

4.9 Optimal path using deterministic vectors



Finally, this plot shows the optimal path when the vectors are deterministic. Though the path is slightly jagged, it reaches the terminal subspace in few iterations.