Value Driven Representations for Reinforcement Learning

Abstract

In reinforcement learning we typically assume the environment is a Markov decision process. While in theory one can formulate real-world decision processes in a Markovian way, this often requires using observation spaces that are very high-dimensional, and we often do not know what the observation space should be ahead of time. Additionally, in many cases the optimal policy might be representable in a small, non-Markovian observation space. We consider how to automatically learn a small observation space that is sufficient to capture a (near) optimal policy. To do so we introduce a new method to optimistically estimate the value of a policy using offline simulated Monte Carlo rollouts. Our approach avoids modeling the existing observation space as Markov, which allows us to learn the optimal policy for non-Markovian setting. This combined with our adaptive observation space augmentation enables our algorithm to learn a (near) optimal policy significantly faster when the optimal policy can be represented using a small non-Markovian observation space.

1 Introduction

In reinforcement learning (RL) an agent seeks to learn a policy, a mapping from observations to actions, that achieves high cumulative reward. A critical choice then is how to represent the domain observation space, which has a direct influence on the space of policies that can be learned, and the suitable algorithms to apply. The majority of research has focused on using a Markovian observation space, which allows agents to bootstrap and leverage the structure of the Bellman equation (Sutton and Barto, 1998). An alternative, increasingly popular approach is to perform direct policy optimization (Williams, 1992), where the designer specifies a particular policy class as a function that constrains the realizable mappings of inputs to outputs (such as a deep neural network (Schulman et al., 2017)). An open challenge in both cases is how to choose the representation. If a representation is chosen that is insufficient to represent the optimal policy, or which does not satisfy the assumptions of the chosen RL algorithm (such as applying deep Q-learning to a non Markovian state space), then in general performance may suffer substantially. However, if an overly expressive representation is chosen, learning the optimal policy could in principle suffer from a very high sample complexity.

Deep neural networks offer the promise of avoiding the need for the RL domain designer to make this tradeoff: the most general input representation (such as input images) can be employed and the deep Q learning method will learn a reduced representation of "features" that are needed to achieve high reward (Mnih et al., 2015). However, despite their many impressive successes, these approaches are only likely to ensure that the optimal policy is representable in the defined observation space (explicitly or implicitly). If a small subspace is sufficient to represent the optimal policy, the deep neural network will not necessarily identify the optimal policy quickly.

Like in other work on state abstraction (e.g. (Cobo et al., 2014)) we hypothesize that in many cases the observation space needed to represent the optimal policy (also known as a "policy-irrelevance abstraction" in the hierarchy of state abstractions (Li, Walsh, and Littman, 2006)) is substantially smaller than the observation representation needed to provide a Markovian representation of the world's dynamics and rewards. In contrast to much state abstraction work that starts with ground states and then aggregates to coarser levels, in this paper we propose a method for incrementally augmenting the input observation representation (used to define the policy mapping of observations to actions), thereby growing the policy class in order to achieve high reward. Our algorithm assumes discrete observation space which is formed into an extremely coarse state aggregation. Our work is inspired by applications where there may be a very large potential set of features, but computing them all in real time may be prohibitive (e.g. consider certain aggregate features of images), and we wish to start with a small observation space and only augment it if we estimate an immediate benefit to doing so.

Our algorithm (VDR: Value Driven Representation) estimates such a benefit by introducing a new method for off policy policy evaluation using the previously collected data, which does not assume the underlying environment is Markovian. VDR is inspired by Upper Con-

fidence Trees (Kocsis and Szepesvári, 2006), a popular Monte Carlo Tree Search method, but adapted to focus on policy evaluation and, more importantly, does not require a model so that it can be run using old data. Our method is used as part of an off policy policy optimization to estimate the value of the best potential policy with a new proposed observation augmentation. Observation augmentations (splits) are only introduced if an improvement in the expected value is estimated.

We evaluate the performance of our approach on several simulation domains where the true Markovian state space is known and demonstrate significant improvement over prior baselines, including to DQN (Mnih et al., 2015) on the Mountain Car domain. While there remains work to be done to improve the computational scalability of our proposed approach, the results demonstrate the benefit of focusing on representations needed to learn the optimal policy rather than that needed to be Markovian.

2 Related Works

Reinforcement Learning in non-Markovian observation space has been long studied. UTree (McCallum and others, 1996) is a history-based method that uses tree-based representations of the value function and splits observation based on local gain and predictive power. Predictive state representation (PSR) (Littman and Sutton, 2002; James, Singh, and Littman, 2004; Aberdeen, Buffet, and Thomas, 2007) is another history-based method that tries to find the sufficient statistics from history to represent a notion of state. In contrast, our algorithm focuses on utility gain (gain in the value of a policy) rather than the predictive power of the state representation.

Feature RL (Hutter, 2009) is a framework that defines a mapping between history to states such that state representation becomes Markovian, and then uses general RL algorithms to solve the proposed MDP. A brief summary of FRL can be found in Daswani et al. (2014). The main difference of our work with this line of research is that our agent does not seek a Markovian representation and finds the policy in a possibly non-Markovian observation space. Many other related works are based on the AIXI agent (Hutter, 2004) a formal mathematical solution to the general RL agent, e.g. MC-AIXI-CTW (Nguyen, Sunehag, and Hutter, 2011; Veness et al., 2011); however, in these methods, the policy representation is not explicit and the agent needs to run UCT at every step.

Our work strongly relates to the state aggregation/abstraction literature (Singh, Jaakkola, and Jordan, 1995; Timmer and Riedmiller, 2006; Ravindran, 2003; Anand et al., 2015). However, our work differentiates itself with those in the way that our algorithm starts learning in a small observation space that is often non-Markovian and then trying to augment the observation space to learn the optimal policy, similar to some Bayesian methods like iPOMDP (Doshi-Velez, 2009) which learns a POMDP while growing the state space.

3 Problem Setup

We consider a Markov decision process with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities T, a reward function R and a discount factor γ. We limit our analysis to discrete state and action spaces, and we assume experience comes in episodes with maximum length τ. For simplicity, we assume the agent always starts in a fixed initial state s_0 though our results can easily be extended to a distribution over initial states. The transition model and reward model are initially unknown. We assume the state space is only indirectly observable through the observation space $\mathcal{O}$. We assume there is a many-to-one deterministic mapping from states to observations, and therefore the observation space can be viewed as an aggregated state space. Define π as a stochastic policy $\pi : \mathcal{O} \times \mathcal{A} \rightarrow [0, 1]$. And we will use the standard definitions $Q^\pi(o_t, a_t) = \mathbb{E}_{o_{t+1}, a_{t+1}, \ldots}[\sum_{t'=t}^{\tau} \gamma^{t'} r_{t'}]$ and similarly for $V^\pi(o_t)$. The goal is to maximize the expected discounted sum of rewards.

Before the first episode, the agent is provided with an initial observation space $\mathcal{O}_0$. Note that there always exists some observation space which makes the domain Markovian, e.g. $\mathcal{O} = \mathcal{S}$. In contrast to standard reinforcement learning, we consider the setting where, at regular intervals, the RL agent can consider augmenting the observation space: pursuing a possible split of an existing observation $o_1 \in \mathcal{O}$ into two observations o_1^1 and o_1^2, in order to find an observation representation that enables learning a policy that achieves high reward.

4 Algorithm

Our method interleaves augmenting the observation space with performing strategic reinforcement learning given the current observation space. Algorithm 1 shows a pseudo code of our algorithm, Value Driven Representation (VDR).

At each episode, VDR finds a policy with optimistic off-policy policy optimization using OPTO (Off-Policy Tree Optimization, section 4.2 and 4.4), hypothesizes a latent feature for each observation (section 4.1) and finally decides to augment the observation space if there exists an augmentation that yields a policy with higher value (section 4.3).

Algorithm 1 VDR

$\mathcal{D} \leftarrow []$ // trajectories
for each episode **do**
  $\pi \leftarrow$ optimistic off-policy policy optimization
  $D \leftarrow D \cup$ trajectory of an episode with π
  for all observations $o \in \mathcal{O}$ **do**
    hypothesize a latent feature
    $V_{\pi_o^*}, \pi_o^* \leftarrow$ off-policy policy optimization
  end for
  $V_{\pi^*}, \pi^* \leftarrow$ off-policy policy optimization
  split($(V_{\pi^*}, \pi^*), \forall o : (V_{\pi_o^*}, \pi_o^*)$)
end for

4.1 Hypothesizing Latent Features

Here we consider a myopic approach to propose a set of $\{\mathcal{O}_i\}$ new potential observation spaces, where each observation space $\mathcal{O}_i$ is derived from taking observation o_i in the current $\mathcal{O}_{current}$ observation space, splitting it into two new observations o_i^1 and o_i^2, and adding these two new observations to all of the other non-split observations $o_{k\neq i} \in \mathcal{O}_{current}$ ($\{o_i^1, o_i^2, \forall_{k\neq i} : o_k \in \mathcal{O}_{current}\}$).

Splitting a particular observation o_i into two is performed by executing Expectation Maximization (EM) on the existing collected trajectories to propose 2 potential latent states with different dynamics and/or rewards. Given the EM learned parameters for a given potential observation o_i split (into observations o_i^1 and o_i^2), the Viterbi algorithm can be used to relabel prior trajectories, turning all instances of observation o_i into o_i^1 or o_i^2.

Note that this approach proposes only myopic local changes to the existing observation space, whereas uncovering an observation representation that is needed to express the optimal policy may require more global changes. However, uncovering the number of latent classes for a dynamic process is itself a challenging problem and often data intensive; for computational and data efficiency, here we consider a simpler myopic approach that is also effective in our simulations.

4.2 Off Policy Tree Optimization (OPTO)

A particular observation space $\mathcal{O}$ defines a decision process that is typically non-Markovian. A key challenge is to find the maximum value policy $V_{\mathcal{O}}^{\pi^*}$ for a new proposed observation space.

$$V_{\mathcal{O}}^{\pi^*} = \max_{\pi} V_{\mathcal{O}}^{\pi}$$

To do so we leverage the prior data collected up to the point at which an observation augmentation is considered.

One possibility is to take the relabeled data (with Viterbi algorithm), assume the new observation space is Markovian, compute the maximum likelihood estimate of parameters, and use model based RL methods to compute the optimal value and policy. However, as we observed in preliminary experiments and is shown in McCallum (1995), this approach can fail as we are typically considering non-Markovian observation spaces.

Instead we approach estimating the optimal policy for the new refined representation as an instance of policy optimization using offline policy evaluation and propose a novel method for off-policy policy evaluation. Performing off-policy policy evaluation is a challenging problem which has received attention over the last few years (Munos et al., 2016; Thomas and Brunskill, 2016; Jiang and Li, 2016; Thomas, Theocharous, and Ghavamzadeh, 2015; Dudík, Langford, and Li, 2011). Our case is particularly challenging since both (1) fairly high-accuracy estimates of the value of the estimated best policy for a new observation representation are needed (since these will be compared to the value of the best policy given the current observation representation) and (2) the domains considered may often be long horizon.

Our method is similar to Monte Carlo Tree Search (MCTS) planning and evaluation methods, with the key distinction that *we do not assume access to a domain model of the decision process* which standard MCTS methods rely on to perform simulations. Instead our algorithm first uses prior data to construct a tree representing observed trajectory sequences, storing counts of the number of times each node has been previously visited at each observation and action node. We call this a trajectory tree $\mathcal{T}$.

In order to evaluate a policy we propose OPTE (Off-Policy Tree Evaluation) that combines n-step return by simulated rollouts and bootstrapping Q values of observation action pairs (o, a). Similar to MCTS, OPTE performs simulated rollouts in $\mathcal{T}$, which are operationalized by starting at the root node and sampling transitions using a MLE of transition probabilities and rewards given the counts stored in a node. It is possible the observed outcomes have not included all feasible next transitions. To handle this case, our algorithm maintains a pseudo-count of $C = 1$ over an additional next unseen observation transition, for each action in the tree. If this outcome is sampled, or if an action has not been tried from a particular node previously (e.g. at step d), we terminate the simulated rollout and use a model free $Q(o_d, a_d)$ estimate of the current observation-action pair at the leaf node. Our complete Off Policy Tree Evaluation (OPTE) is shown in Algorithm 2, which takes the average of N simulations.

$$V_{\mathcal{O}}^{\pi} = \frac{1}{N}\sum_{i=1}^{N}(\sum_{t=1}^{d_i} r_t + Q(o_{d_i}, a_{d_i}))$$

Algorithm 2 OPTE($\pi, graph, N, optFlag$)

```
for i = 1 : N do
    node ← graph.initialNode()
    o ← node.OBSERVATION(), R(i) ← []
    done ← False
    while not done do
        node, r, done ← next(node, π(o), optFlag)
        R(i).append(r)
    end while
end for
return avg(discountedReturn(R))
```

Similar to the Q estimates at nodes computed during Monte Carlo Tree Search, we use a model free Q estimate computed by averaging over all returns obtained after observing this (o, a) tuple. If all data used to estimate this Q value asymptotically comes from the same policy, the Q value will converge to the true Q value for that policy for this observation and action. This mild

form of bootstrapping may increase the bias of the resulting policy estimate but can hopefully both reduce the variance and help us to evaluate new potential policies. we will further discuss the benefits and limitations of this choice later in the paper.

Using OPTE as a simulator, any policy optimization method (e.g. REINFORCE (Williams, 1992)) can be applied to achieve a (near) optimal stochastic policy, we call this OPTO (Off-Policy Tree Optimization) that can be used in VDR (Algroithm 1) to perform off-policy policy optimization. Additionally in smaller domains, one can enumerate all the possible $|\mathcal{O}|^{|\mathcal{A}|}$ deterministic policies and use OPTE to find the best deterministic policy.

Algorithm 3 next($node, a, optFlag$)

$C \leftarrow 1$
$counts \leftarrow node.getTransitionCounts(a)$
$counts.append(C)$ // Unobserved outcome
$model \sim MLE(counts)$
$nextNode \sim model(a)$
if $nextNode$ is -1 **then**
  // Sample a new potential outcome
  $o \leftarrow node.OBSERVATION()$
  if $optFlag == True$ **then**
    $r \leftarrow graph.Q[o,a] + \frac{r_{max}}{1-\gamma}\sqrt{\frac{log(N)}{n[o,a]}}$
  else
    $r \leftarrow graph.Q[o,a]$
  end if
  $done \leftarrow True$
  return $(nextNode, r, done)$
else
  $r, done \leftarrow nextNode.Info()$
  return $(nextNode, r, done)$
end if

4.3 Value Of Splitting

The objective is to augment the observation space only if there is a value of observation refinement: that splitting an observation yields an optimal policy with a higher value than the optimal policy for the current representation. More precisely, define the potential increase in value as

$$\Delta = \max_i(\max_{\pi'} V^{\pi'}_{\mathcal{O}_i} - \max_{\pi} V^{\pi}_{\mathcal{O}_{current}})$$

where $\mathcal{O}_{current}$ is the current observation space and $\mathcal{O}_i$ is the observation space if observation o_i is split into two observations. The maximum over policies is computed by OPTO. Observation refinement is only defined to be valuable if $\Delta > 0$, indicating that there is a estimated improvement in value by splitting.

Our approach makes two further modifications to the above criteria to increase robustness and adherence to the aim of observation refinement. First, a predicted improvement in value could arise due to more accurate models, even when the resulting optimal policy is the same as before a split. Therefore a split is only done if the best policy for the augmented observation space is different for at least one observation compared to the prior best policy for the current observation space.

Second, observation augmentation will turn one observation into two, and by definition indicates that at least one observation will now have less counts in observed trajectories compared to when that observation was not refined. This reduced data will generally increase the variance of the computed estimated values of the possible refined observation spaces. Therefore we use a bootstrap procedure on the all old data to compute B estimates of the refined observation values. In deciding whether to split, we compare the potential benefit to the estimated standard deviation across bootstrap estimates of the value of the new proposed observation splits as

$$score(i) = \mathbb{1}(\pi_{old} \neq \pi_i)\frac{avg(V^{\pi_i}) - V^{\pi_{old}}}{std(V^{\pi_i})}$$

and split if $score$ exceeds an input threshold. This is a heuristic estimate of a significance test (Z-score) for whether the algorithm is confident that the new split representation will outperform the prior.[1] In the case of stochastic policy we use the $D_{KL}(\pi_{old}||\pi_i)$ instead of the indicator function.

4.4 Strategic Exploration

Most approaches for exploration combined with Monte Carlo evaluation for online reinforcement learning involve stochastic policy gradient methods, where exploration is achieved through stochasticity in the current policy. However, such methods do not typically perform strategic exploration and are only guaranteed to converge to a local optima; thus, while effective, they are not always the best choice for sample-efficient reinforcement learning. Another classic method is to perform exploring starts Monte Carlo learning of the Q function (as used in Thomas, Theocharous, and Ghavamzadeh (2015)); however, this method requires the agent to be able to teleport to any observation, which is infeasible in many domains of interest.

Instead we propose a method that is inspired by optimism under uncertainty approaches. Precisely, we take the OPTE algorithm (Algorithm 2) described above and add in a reward bonus to the $Q(o,a)$ used at the tree leaves. Similar to upper confidence bound RL algorithms (Osband and Van Roy, 2014) we use $Q(o,a) + \frac{r_{max}}{1-\gamma}\sqrt{log(n(o))/n(o,a)}$, where r_{max} is the maximum reward. As this reward is propagated and averaged up the tree, it will create policies that encourage visiting parts of the decision process for which there is little data. Practically this approach is a minor modification of our OPTE algorithm, which can be computed by setting the input *optFlag* to *True*. As shown in VDR (Algorithm 1), *optFlag* is set to *True* to find a policy to run in

[1] We break the tie based on the BIC score of a split.

the environment, and is set to *False* in order to compute the value of a split.

5 Asymptotic Analysis

In this section we analyze the asymptotic behaviour of our proposed algorithm in the limit of infinite data. This allows us to study the asymptotic behaviour, however as shown in section 6, our empirical evaluation shows the desired effect with limited data. Lemma 5.1 states that we can have an accurate estimate of $V^{\pi}_{\mathcal{O}}$ using trajectory tree $\mathcal{T}$.

Lemma 5.1. *Let M be a Markov decision process, and $\mathcal{T}$ be a trajectory tree generated by infinite data gathered using a policy π s.t. $\forall o, a : \pi(a|o) > 0$. Then $\forall \pi : \mathcal{O} \times \mathcal{A} \to [0, 1]$, $V^{\pi}_{\mathcal{T}} \xrightarrow{i.p.} V^{\pi}_{M}$. Where $V^{\pi}_{\mathcal{T}}$ and V^{π}_{M} are values of the policies evaluated in trajectory tree $\mathcal{T}$ and MDP M, respectively.*

Proof. (sketch) Given infinite data, evaluating a policy using $\mathcal{T}$ invovles no bootstrapping with Q, and all the history based transition probabilities and rewards converges in probability to their real values by the law of large numbers. As a result value of the policy, $\sum_{t\sim\pi} p(t)G(t)$ converges in probability to its real value, where $t \sim \pi$ are all the trajectories generated by policy π, $p(t)$ is the probability of trajectory under policy π, and $G(t)$ is the return of the trajectory. □

However, in the case of limited data the accuracy of $V^{\pi}_{\mathcal{T}}$ depends on the accuracy of the model free $Q(o, a)$ estimates. Using Lemma 5.1 we can find the best deterministic policy, since one can simply enumerate all the possible $|\mathcal{O}|^{|\mathcal{A}|}$ policies and find the best policy, where Lemma 5.1 ensures the accuracy of value estimation. Empirically, we find that in most cases the optimal (stochastic) policy can be learned in almost linear time in the size of observation space $|\mathcal{O}|$ using policy optimization methods.

Theorem 5.2 states that, by Lemma 5.1, our algorithm will not split a Markovian representation further.

Theorem 5.2. *Let M and $\tilde{M}$ be Markov decision processes over the same action space $\mathcal{A}$ and state spaces $\mathcal{S}$, $\tilde{\mathcal{S}}$, respectively. Where $\tilde{\mathcal{S}} = \mathcal{S} \setminus s_0 \cup \{s^1_0, s^2_0\}$ such that s^1_0, s^2_0 are the split of state s_0. Let $\pi^* : \mathcal{S} \to \mathcal{A}$ be the optimal policy in M. Then, $\exists \tilde{\pi} : \tilde{\mathcal{S}} \to \mathcal{A}$ such that $V^{\tilde{\pi}}_{\tilde{M}} = V^{\pi}_{M}$ where $\tilde{\pi}(s^1_0) = \tilde{\pi}(s^2_0) = \pi(s_0)$, and $\forall \pi' \neq \tilde{\pi} : V^{\pi'}_{\tilde{M}} \leq V^{\tilde{\pi}}_{\tilde{M}}$.*

Proof. (sketch) In Markovian representation, any state s_0 and its split s^1_0, s^2_0 are indistinguishable, so for any policy with different actions in s^1_0 and s^2_0 there exists a policy with higher or equal value with the same action in s^1_0 and s^2_0. □

Detailed proof can be found in supplementary materials. However, In the case of limited data and using policy optimization to find the optimal policy, we might not get an accurate estimate of the value, or find the deterministic policy that is optimal in MDP.

The π^*-irrelevance abstraction (Li, Walsh, and Littman, 2006) is a state abstraction of an MDP that, in every cluster, the optimal action is the same. The following lemma states that our algorithm will not split a observation space that is a π^*-irrelevance abstraction.

Lemma 5.3. *If the observation space $\mathcal{O}$ is π^*-irrelevance abstraction of an MDP M, our algorithm will not split further.*

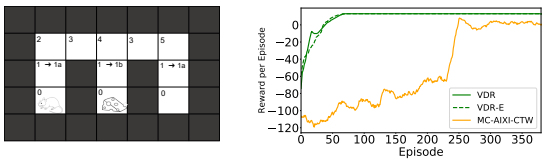
Proof. (sketch) The optimal policy in π^*-irrelevance abstraction has the same value as the optimal policy for M. Thus, there does not exist a split that yields higher return and our algorithm will not split further. □

A remaining question is if our algorithm continues splitting until finding the optimal policy. Unfortunately there is no guarantee on the amount of data needed for EM to discover the true latent feature. Spectral methods (Anandkumar et al., 2014) can yield a finite sample guarantee but these methods often require separability and/or fast mixing. However, with a stochastic policy, all the possible trajectories will happen *almost surely* in infinite time and we will have the necessary transitions to discover the latent features, and the sufficiency of these data depends on the choice of algorithm. Empirically our algorithm will continue splitting until finding the optimal policy.

6 Simulation Results

In this section we compare the performance of our algorithm on some standard reinforcement learning environments with PSRL (Osband and Van Roy, 2014), TCRL (Mandel et al., 2016) and MC-AIXI-CTW (Nguyen, Sunehag, and Hutter, 2011). PSRL uses a Markovian state representation to find the optimal policy. TCRL is a clustering algorithm, which starts with full Markovian representation and clusters outcomes together to generalize between states and showed a superior performance in environments where state aggregation can boost learning. We additionally compare to MC-AIXI-CTW, which outperformed other feature RL and history based methods (Veness et al., 2011).

Our algorithm (VDR) that uses REINFORCE as a policy optmization method, starts with a small observation representation and augments the representation as it progresses, if the augmentation yields a better policy. The goal is to see how much better we can do if we don't assume full state Markovian representation for environments from the beginning. Additionally, we have evaluated our algorithm using policy enumeration where rather than using REINFORCE we exhaustively searched over all $|\mathcal{O}|^{|\mathcal{A}|}$ deterministic policies, it is shown as VDR-E (VDR-Enumeration) in plots. Also, in order to speed up the simulations we changed the frequency in which VDR considers an observation space augmentation. (More experimental results can be found in supplementary materials)



(a) (b)

Figure 1: Cheese Maze, (a) Environment: agent has 4 actions: move up, down, left, right and receives a reward of -10 if it hits the wall, -1 for moving to an empty state and 10 for getting to the cheese state which also teleports the agent to the initial state (bottom left). Numbers in each cell indicate the initial observation space, and split of observation 1 into (1a, 1b). (b) Comparison to MC-AIXI-CTW

6.1 Cheese Maze

Cheese Maze was used as a benchmark environment in Veness et al. (2011); for details refer to McCallum and others (1996). Figure 1a shows the environment, and numbers in each cell indicate the initial observation space. We compared our algorithm to MC-AIXI-CTW in this environment with a small modification that we start each episodes at the bottom left cell. We set the maximum length of each episode to 20, and consider a split every 5 episodes.

As shown in Figure 1b VDR outperforms MC-AIXI-CTW and finds the optimal policy in fewer number of episodes. Additionally VDR is computationally an order of magnitude faster mainly because MC-AIXI-CTW uses UCT algorithm at every step to decide what action to take, but VDR computes a fix policy for each episode. VDR-E and VDR find the optimal policy by splitting observation 1 into two (1a, 1b) that requires different action for representing the optimal policy in center and left side of the maze (see Figure 1a). Using a stochastic policy is slightly outperforming the deterministic policy since before the split, a stochastic policy can reach to a positive reward by randomizing action in observation 1.

6.2 Sixarms

Sixarms is a 7 state MDP (Strehl and Littman, 2008) shown in Figure 2a. We used a horizon of 10 steps, evaluate a split every 20 steps and start with one initial observation that aliases all the states in Markovian state representation. The optimal policy in this environment is taking action 5 from state 0 and the same action from state 6, this policy can be represented by only using one observation.

Due to small transition probabilities, a large amount of data is required to get a good empirical estimate of transition probabilities and reward, this introduces a challenge for finding the optimal policy. In the ideal case, we don't need any extra split to learn the optimal policy but around 10 percent of the time VDR does at least one split. The reason for the erroneous splits is that we might need more trails to get a decent empirical estimate of transition probabilities. A poor estimate can make a sub-optimal policy yield a high reward at the time of a split. However, after collecting enough data our algorithm is able to learn the optimal policy. Figure 2b shows the performance of our algorithm with a major improvement over baselines.

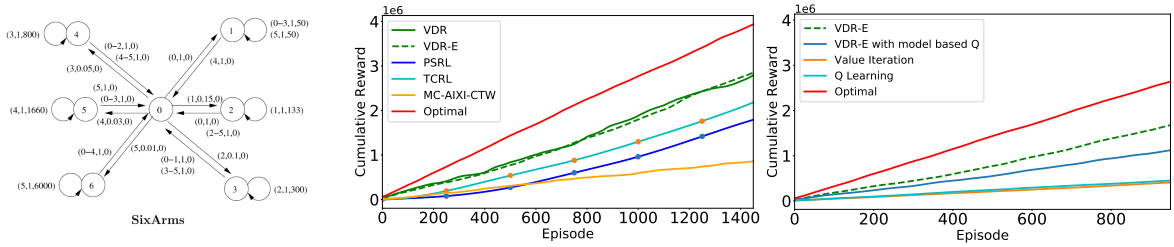
In section 4.2 (algorithm 2) we mentioned the importance of not erroneously making the Markov assumption in evaluating the policy offline. Figure 2c compares 4 different methods for off-policy policy evaluation: (1) using VDR-E with MC estimate of $Q(o, a)$ at leaf nodes, (2) using VDR-E with $V(o)$ at leaf nodes obtained by value iteration, (3) using model-based value iteration, (4) using Q-learning. Methods (3) and (4) completely fail to find the optimal policy as our observation space is non-Markovian. Method (2) works well in an extreme case with (a) a fairly small observation-action space and (b) a short horizon. In these cases OPTE eliminates the use of value at leaf nodes since we can collect sufficient data. Sixarms satisfies both requirements and as shown in Figure 2c method (1) and (2) has comparable results.

VDR can do better than the state-of-the-art Markovian based algorithms when the optimal policy can be represented in a smaller observation space; however, when this assumption is not true our algorithm can still find the optimal policy. We changed the sixarms environments by swapping all actions that yields a reward in each state (except state 0) to obtain the worst case scenario for VDR, in this case the optimal policy can only be learned if all states are distinct (i.e. Markovian state representation is required). In this situation VDR can still learn the optimal policy but slower than algorithms like PSRL that assumes Markovian representation from the beginning (refer to supplementary materials for more details).

6.3 Mountain Car

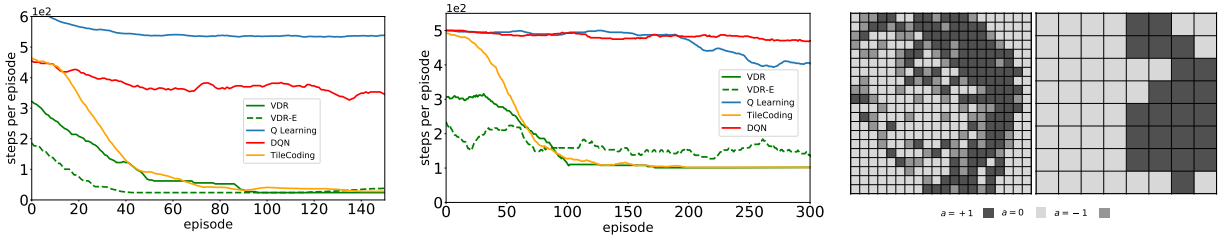
Mountain car is continuous state space environment introduced in Sutton and Barto (1998) where a car on a hill tries to reach the goal position up the right side of the hill. Using Tile coding (Sherstov and Stone, 2005) and Gaussian Processes (Wilson and Adams, 2013; Deisenroth, Fox, and Rasmussen, 2015) are effective in this envinronement. Additionally continuous action space algorithms can solve this environment very efficiently (Lillicrap et al., 2015; Degris, White, and Sutton, 2012).

We considered mountain car where the agent always starts at the same location $x_0 = -0.5$ but with two different initial velocities ($v_0 = 0.03, v_0 = 0.05$). In all simulations the initial starting position and velocity is fixed for that entire process. We set the maximum episode length to 500 and consider a split every 20 episodes. In our algorithm we treat the underlying true state space as a discrete 8 by 8 grid, though the true space is best modeled continuously. We compared our



(a) (b) (c)

Figure 2: (a) Six Arms domain, each node in the graph is a state in Markovian state representation and each edge is labeled by one or more transitions of form (action, probability, reward) and agent starts in state 0 (Strehl and Littman, 2008), (b) comparison with TCRL, PSRL and MC-AIXI-CTW (Results of PSRL and TCRL are from (Mandel et al., 2016)), (c) Comparison of 4 different methods for off-policy policy evaluation



(a) (b) (c)

Figure 3: Comparison of Q-learning on 20 by 20 grid, DQN, tile coding and VDR on mountain care with initial velocity (a) $v_0 = 0.05$ and (b) $v_0 = 0.03$. Split analysis: (c) $v_0 = 0.05$ left: policy found by Q-learning on 20 by 20 gird, right:policy found by VDR on underlying 8 by 8 grid. Each color represent a distinct action.

algorithm with Q-learning[2] with ϵ-greedy exploration, DQN and tile coding.

Figure 3a shows the result for $v_0 = 0.05$, where VDR can learn the optimal policy in less than 100 episodes. Figure 3c shows the policy found by Q-learning on 20 by 20 grid and VDR-E (on underlying 8 by 8 grid), where VDR-E found a coarse version of Q-learning algorithm that is just enough for this initial velocity.

For $v_0 = 0.03$ if we start VDR-E with only one initial observation, there is no policy which reaches the goal. In that case our algorithm receives no reward signal so it will not do any split and cannot find the optimal policy. However, this is not the case for VDR with stochastic policy. To remedy this issue, we initialize the initial observation space with two observation by cutting the location axis into half. With this initialization, VDR converges to the optimal policy faster than other methods. (Figure 3b).

7 Future Work And Conclusion

In this work we have presented an algorithm for adaptively augmenting the observation space of a decision process and a novel off-policy policy evaluation method (OPTE) to obtain a decision policy with high performance. By relying on strategic policy optimization, we have shown that we can quickly learn a good observation space that supports representing the optimal policy in a few simple but classic reinforcement learning tasks, even when that observation space is non-Markovian.

We have found significant improvements over baselines in our simulated experiments. Going forward, one important challenge is reducing the computation required to hypothesize a split with EM. As one of the key motivations for this work is when there may be observation features or distinctions that are not initially included in the state specification, we are also curious to try this method with a human-in-the-loop to provide new proposed observation refinements.

Additionally, extending this approach to continuous state space is an interesting line of research, that can be seen as iteratively projecting the state space in lower dimensional spaces and find the minimum rank subspace that is sufficient to learn the optimal policy.

[2]We experimented with multiple grid sizes, including 8x8, 20x20 and 30x30. Empirically we found that a 20 by 20 grid often showed the best performance at converging to a good quality policy.

References

Aberdeen, D.; Buffet, O.; and Thomas, O. 2007. Policy-gradients for psrs and pomdps. In *Artificial Intelligence and Statistics*, 3–10.

Anand, A.; Grover, A.; Singla, P.; et al. 2015. Asap-uct: Abstraction of state-action pairs in uct. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Anandkumar, A.; Ge, R.; Hsu, D.; Kakade, S. M.; and Telgarsky, M. 2014. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research* 15(1):2773–2832.

Cobo, L. C.; Subramanian, K.; Isbell, C. L.; Lanterman, A. D.; and Thomaz, A. L. 2014. Abstraction from demonstration for efficient reinforcement learning in high-dimensional domains. *Artificial Intelligence* 216:103–128.

Daswani, M.; Sunehag, P.; Hutter, M.; et al. 2014. Feature reinforcement learning: state of the art. In *Proc. 28th AAAI Conf. Artif. Intell.: Sequential Decision Making with Big Data*, 2–5.

Degris, T.; White, M.; and Sutton, R. S. 2012. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*.

Deisenroth, M. P.; Fox, D.; and Rasmussen, C. E. 2015. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence* 37(2):408–423.

Doshi-Velez, F. 2009. The infinite partially observable markov decision process. In *Advances in neural information processing systems*, 477–485.

Dudík, M.; Langford, J.; and Li, L. 2011. Doubly robust policy evaluation and learning. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, 1097–1104.

Hutter, M. 2004. *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media.

Hutter, M. 2009. Feature reinforcement learning: Part i. unstructured mdps. *Journal of Artificial General Intelligence* 1(1):3–24.

James, M. R.; Singh, S.; and Littman, M. L. 2004. Planning with predictive state representations. In *Machine Learning and Applications, 2004. Proceedings. 2004 International Conference on*, 304–311. IEEE.

Jiang, N., and Li, L. 2016. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, 652–661.

Kocsis, L., and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, 282–293. Springer.

Li, L.; Walsh, T. J.; and Littman, M. L. 2006. Towards a unified theory of state abstraction for mdps. In *ISAIM*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Littman, M. L., and Sutton, R. S. 2002. Predictive representations of state. In *Advances in neural information processing systems*, 1555–1561.

Mandel, T.; Liu, Y.-E.; Brunskill, E.; and Popovic, Z. 2016. Efficient bayesian clustering for reinforcement learning. In *IJCAI*, 1830–1838.

McCallum, A. K., et al. 1996. Learning to use selective attention and short-term memory in sequential tasks. In *From animals to animats 4: proceedings of the fourth international conference on simulation of adaptive behavior*, volume 4, 315. MIT Press.

McCallum, A. 1995. *Reinforcement Learning with Selective Perception and Hidden State*. Ph.D. Dissertation.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Munos, R.; Stepleton, T.; Harutyunyan, A.; and Bellemare, M. 2016. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 1046–1054.

Nguyen, P.; Sunehag, P.; and Hutter, M. 2011. Feature reinforcement learning in practice. In *European Workshop on Reinforcement Learning*, 66–77. Springer.

Osband, I., and Van Roy, B. 2014. Near-optimal reinforcement learning in factored mdps. In *Advances in Neural Information Processing Systems*, 604–612.

Ravindran, B. 2003. Smdp homomorphisms: An algebraic approach to abstraction in semi markov decision processes.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sherstov, A. A., and Stone, P. 2005. Function approximation via tile coding: Automating parameter choice. In *International Symposium on Abstraction, Reformulation, and Approximation*, 194–205. Springer.

Singh, S. P.; Jaakkola, T.; and Jordan, M. I. 1995. Reinforcement learning with soft state aggregation. In *Advances in neural information processing systems*, 361–368.

Strehl, A. L., and Littman, M. L. 2008. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences* 74(8):1309–1331.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Thomas, P. S., and Brunskill, E. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*.

Thomas, P. S.; Theocharous, G.; and Ghavamzadeh, M. 2015. High confidence off-policy evaluation. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence*.

Timmer, S., and Riedmiller, M. 2006. Abstract state spaces with history. In *Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American*, 661–666. IEEE.

Veness, J.; Ng, K. S.; Hutter, M.; Uther, W.; and Silver, D. 2011. A monte-carlo aixi approximation. *Journal of Artificial Intelligence Research* 40(1):95–142.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wilson, A., and Adams, R. 2013. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning*, 1067–1075.