Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence

Scaling Up Reinforcement Learning through Targeted Exploration

Timothy A. Mann and **Yoonsuck Choe**
Department of Computer Science & Engineering
Texas A&M University
3112 TAMU, College Station, TX 77843-3112, USA
{ *mann23, choe* }@*tamu.edu*

Abstract

Recent Reinforcement Learning (RL) algorithms, such as R-MAX, make (with high probability) only a small number of poor decisions. In practice, these algorithms do not scale well as the number of states grows because the algorithms spend too much effort exploring. We introduce an RL algorithm State TArgeted R-MAX (STAR-MAX) that explores a subset of the state space, called the *exploration envelope* ξ. When ξ equals the total state space, STAR-MAX behaves identically to R-MAX. When ξ is a subset of the state space, to keep exploration within ξ, a *recovery rule* β is needed. We compared existing algorithms with our algorithm employing various exploration envelopes. With an appropriate choice of ξ, STAR-MAX scales far better than existing RL algorithms as the number of states increases. A possible drawback of our algorithm is its dependence on a good choice of ξ and β. However, we show that an effective recovery rule β can be learned on-line and ξ can be learned from demonstrations. We also find that even randomly sampled exploration envelopes can improve cumulative rewards compared to R-MAX. We expect these results to lead to more efficient methods for RL in large-scale problems.

Introduction

In the Reinforcement Learning (RL) framework, each time step, the agent must decide whether to act in a way that it can confidently predict the reward it will receive or try an action where the reward received is uncertain but may have higher payoff. This problem is the well-known exploration/exploitation trade-off. If the agent does not explore enough of its environment, it may miss out on undiscovered, high rewards. On the other hand, the agent may waste time looking for a better policy that may not exist.

Recognizing the importance of the exploration/exploitation trade-off, recent research on RL has resulted in algorithms that, with high probability, can act near-optimally after training only on a polynomial (in the # of states and actions) number of time steps (Kearns and Singh, 2002; Brafman and Tennenholtz, 2002; Kakade, 2003; Strehl, Li, and Littman, 2009). The success of these provably efficient algorithms depends critically on showing that the agent will quickly explore all actions in every reachable state. In other words, efficient RL seems to depend on thoroughly exploring all reachable combinations of states and actions. For example, R-MAX (Brafman and Tennenholtz 2002) is a provably efficient RL algorithm that, essentially, exhaustively explores reachable state-action pairs. Unfortunately, when applied to real-world tasks with large numbers of states, R-MAX may spend unreasonable effort exploring.

Simple Exploration Sufficient?: In practice, simple exploration strategies can often quickly find good policies. For example, the ϵ-greedy selection strategy chooses the action that the agent believes will lead to highest long term reward with probability $1 - \epsilon$ and uniformly draws a random action with probability ϵ. (Hester and Stone 2009) introduced a more sophisticated RL algorithm that thoroughly explores at first but stops exploration when a sufficiently high rewarding state is found. However, these exploration strategies can quickly get stuck in trajectories that lead to mediocre total reward (e.g. see the Red Herring domain in (Hester and Stone 2009)).

Additional Assumptions Needed: Without thorough systematic exploration, we cannot prove, in general, that an RL algorithm will converge to an optimal policy because the algorithm may miss critical information for planning. If we want to guarantee that a near-optimal policy will be learned while at the same time avoiding exhaustive exploration, then we need to make additional assumptions concerning either the structure of the problem or the *a priori* information available to the agent.

Alternative 1, Additional Structure: One possibility is to assume that the learning problem has additional structure. For example, (Leffler, Littman, and Edmunds 2007) and (Brunskill et al. 2009) assume that the set of states is partitioned into a set of equivalence classes. Both take advantage of the fact that actions have the same effects for all states in the same class. This assumption is powerful and indeed does result in near-optimal guarantees with less exploration provided that the number of partitions is much smaller than the number of states (see (Leffler, Littman, and Edmunds 2007) for details). The main drawback of this approach is that a useful partition must exist, and the agent must be given *a priori* a mapping from each state to its corresponding equivalence class. In addition, even though the agent does not have to visit every single state-action pair, it does have to visit a representative of every state class at least enough times to try every action (more for the stochastic case). If the number of state classes is huge or one of the state classes is dangerous (e.g. falling off a cliff), then the agent may never survive to learn a near-optimal policy.

Alternative 2 (Proposed Method): Another possibility is that the agent may have prior knowledge about states that it should avoid, either because (1) they are dangerous or (2) simply irrelevant to the current task. In this work, we assume that the agent is given an *exploration envelope* ξ containing a set of states (or state-action pairs) that are potentially relevant to the current task and a *recovery rule* β that is applied when the agent accidentally leaves ξ. Any state not in the exploration envelope can be safely assumed irrelevant to the current task. On the other hand, states in the exploration envelope should be explored. Although a recovery rule cannot stop the agent from leaving ξ, it can quickly return the agent to a state in ξ. The main advantage of our algorithm, called STAR-MAX, is that it does not need to explore every state, which in some cases should allow it to learn faster than R-MAX. The main potential limitation of STAR-MAX is that the algorithm needs to be given an exploration envelope and a recovery rule. It turns out that both issues can be effectively handled. The recovery rule can be obtained by learning a simple state-independent action model. To address the problem of obtaining an exploration envelope we first show that a useful envelope can be learned from demonstrations and, further, show that even a naïvely constructed envelope (by randomly dropping states) achieves higher cumulative reward than R-MAX.

The main contributions of this paper are (1) the introduction of State TArgeted R-MAX (STAR-MAX), a model-based RL algorithm that thoroughly explores only a subset of state-action pairs called the exploration envelope, and (2) the discovery that a recovery rule can be learned on-line efficiently and a useful exploration envelope can be learned from demonstration or simply by randomly dropping states.

Background

We describe tasks using the Markov decision process (MDP) formulation (Sutton and Barto 1998). An MDP task M is a 5-tuple $\langle S, A, T, R, \gamma\rangle$ where S is a set of states, A is a set of actions, T is a set containing $p(s'|s,a)$ denoting the probability of transitioning to state s' after selecting action a while in state s, $R : S \times A \rightarrow \mathbb{R}$ assigns an expected immediate reward to each state-action pair, and $0 \leq \gamma < 1$ discounts future rewards compared to immediate rewards. The objective of most RL algorithms is to find a policy $\pi : S \rightarrow A$ that maximizes the equation

$$Q^{\pi}_{M}(s,a) = R(s,a) + \gamma E\left[Q^{\pi}_{M}(s', \pi(s'))|M, \pi, s, a\right] \quad (1)$$

for all encountered $s \in S$ and $a \in A$, where E is the expected value taken with respect to the next state s'. Equation 1 is the expected sum of discounted rewards received for taking action a in state s and thereafter following policy π. The equation defines an implicit policy

$$\pi(s) = \arg\max_{a \in A} Q(s,a) \quad (2)$$

for all $s \in S$.

There are many RL algorithms that learn to maximize equation 1 in the limit. Recently algorithms have been introduced that, with high probability, learn a near-optimal policy in a polynomial number of time steps. An example is R-MAX (Brafman and Tennenholtz 2002), a popular model-based RL algorithm, which learns the transition probabilities T and reward function R from observations and then uses a dynamic programming algorithm, such as value iteration, to plan a policy from the model.

Interestingly, while learning T and R, R-MAX uses its incomplete model to decide where to explore. Assuming that insufficiently modeled state-action pairs give the maximum possible reward, the planner generates policies that aggressively explore poorly modeled transition probabilities, until the agent either finds a maximally rewarding trajectory in M or every reachable state-action pair is well-modeled. This strategy for quickly exploring the state-action space is sometimes referred to as "optimism in the face of uncertainty", and allows R-MAX to aggressively and efficiently explore the state space.

Unfortunately, repeatedly exploring every state can be wasteful if an agent has prior knowledge that some states are irrelevant to the task. A good choice of which states to explore, combined with a reasonable recovery rule to help the agent stay within the relevant region of state space, can dramatically reduce the amount of time required to learn a near-optimal policy. However, to our knowledge no algorithm is yet able to exploit this kind of information to reduce sample complexity compared to R-MAX. In the next section, we describe State TArgeted R-MAX (STAR-MAX), an algorithm that is capable of targeting its exploration to specific states while minimizing exploration of states that are known to be irrelevant or dangerous.

Algorithm

Instead of being generally optimistic in the face of uncertainty (which encourages the agent to explore everything), we consider a learning agent that is

1. optimistic about a subset of states $\xi \subseteq S$, called the *exploration envelope*, and
2. given a policy β, called the *recovery rule*, that can return the agent to a state within ξ in a small number of actions.

The exploration envelope is assumed to contain all relevant states and some irrelevant states. Thus the agent must explore ξ to determine which states are relevant and which are irrelevant. The concept of an envelope of states was first introduced by (Dean et al. 1995). However, that work focused on efficient planning given a complete model of the environment. Our approach is different because we are attempting to learn an accurate model for a *subset* of the state space. The main difficulties here are (1) selecting an appropriate exploration envelope and (2) keeping the exploration effort within the envelope. Focusing exploration to a limited region is where the importance of the recovery rule comes in. During exploration the agent may leave the exploration envelope and waste time trying to return to ξ. We do not assume that the recovery rule acts optimally, only that it quickly returns the agent to a state in ξ.

STAR-MAX (outlined in algorithm 1) has parameters:

- S and A : state and action sets, respectively
- γ : the discount factor
- m : # visits before state-action pair is "well modeled"
- ξ : the exploration envelope (a subset of S)
- β : the recovery rule

Algorithm 1 State TArgeted R-MAX (STAR-MAX)

```
Require: $S, A, \gamma, m, \xi, \beta$
1: for all $(s,a) \in S \times A$ do
2:   if $s \in \xi$ then
3:     $Q(s,a) \leftarrow \frac{R_{\mathrm{MAX}}}{1-\gamma}$
4:   else
5:     $Q(s,a) \leftarrow \frac{R_{\mathrm{MIN}}}{1-\gamma}$
6:   end if
7:   $n(s,a) \leftarrow 0$ {# visits to $(s,a)$}
8:   $r(s,a) \leftarrow 0$ {Cumulative reward at $(s,a)$}
9:   for all $s' \in S$ do
10:    $l(s,a,s') \leftarrow 0$ {# transitions $(s,a) \rightarrow s'$}
11:  end for
12: end for
13: for $t = 1, 2, 3, \ldots$ do
14:   Let $s$ denote the state at time $t$.
15:   if $s \in \xi$ then
16:     Choose action $a := \arg\max_{b \in A} Q(s,b)$.
17:   else
18:     Choose action $a := \beta(s)$.
19:   end if
20:   Let $r$ be the immediate reward and $s'$ be the next state
      after executing action $a$ from state $s$.
21:   if $n(s,a) < m$ then
22:     $n(s,a) \leftarrow n(s,a) + 1$
23:     $r(s,a) \leftarrow r(s,a) + r$
24:     $l(s,a,s') \leftarrow l(s,a,s') + 1$
25:     if $s \in \xi$ and $n(s,a) = m$ then
26:       Run Value Iteration on model (Algorithm 2).
27:     end if
28:   end if
29: end for
```

When STAR-MAX is initialized (lines 1 through 12), a data structure Q, used to estimate equation 1, is initialized so that any state action pair with a state in ξ is initialized optimistically; otherwise it is initialized pessimistically. This encourages the agent to explore states in ξ and avoid other states. Data structures are also initialized to count $n(s,a)$ the number of times an agent has visited a particular state-action pair $(s,a) \in S \times A$, $l(s,a,s')$ the number of times an agent visited state-action pair $(s,a) \in S \times A$ and transitioned to $s' \in S$, and the cumulative reward $r(s,a)$ received when the agent visited state-action pair $(s,a) \in S \times A$.

After initialization, the learner enters a loop (line 13) interacting with its environment. It receives the current state s (line 14). Then if the current state is in the exploration envelope the agent will use the estimated Q function to greedily select what it believes to be the best action (line 16). On the other hand, if the state s is not in ξ, then the recovery rule selects the action (line 18). This way, if the agent leaves ξ, then the recovery rule is able to return the agent to a state in ξ. Finally, if the state-action pair (s,a) is unknown, then each of the counters is updated, and if a new state action pair becomes known (line 25), then Q is updated by running value iteration on the estimated model ($\hat{T}$ and $\hat{R}$). The estimated model, outlined in algorithm 2, returns a pair where the first element is the state transition probability and the second element is the immediate reward.

Algorithm 2 Estimate Model $(\hat{T}, \hat{R})$

```
Require: $s, a, s', m, \xi$
1: if $s \in \xi$ and $n(s,a) \geq m$ then
2:   return $\left(\frac{l(s,a,s')}{n(s,a)}, \frac{r(s,a)}{n(s,a)}\right)$ {Sufficiently explored}
3: else
4:   if $s \in \xi$ and $s = s'$ then
5:     return (1, $R_{\mathrm{MAX}}$) {Under explored, in $\xi$}
6:   else
7:     if $s = s'$ then
8:       return (1, $R_{\mathrm{MIN}}$) {Not in $\xi$}
9:     else
10:      return (0, $R_{\mathrm{MIN}}$)
11:    end if
12:  end if
13: end if
```

The STAR-MAX algorithm is very similar to R-MAX. The key differences are that STAR-MAX initializes only a subset of the state space optimistically and that it uses the recovery rule whenever it leaves the exploration envelope ξ.

Next we show that STAR-MAX scales gracefully to domains with a large number of states and show how the main potential limitations of the algorithm (selecting the right exploration envelope and recovery rule) can be overcome.

Experiments and Results

Our experiments are performed on grid world environments (see (Sutton and Barto 1998)). In a grid world environment the agent's state is a cell on a 2-dimensional grid. The agent has four actions corresponding to the four cardinal directions with probability 0.8 and one of the two corresponding diagonal directions with probability 0.2.

For all experiments we used a discount factor of $\gamma = 0.9$. A value of $m = 5$ was used for all experiments unless otherwise noted. A learning rate of 0.2 was used for Q-learning.

In the first two experiments we consider the scalability of STAR-MAX with a learned recovery rule.

Scalability with Learned Recovery Rule

During exploration the agent does not have a good approximation of the environment and can easily wander into irrelevant regions of the state space. (Hans et al. 2008) introduced a concept similar to our recovery rule, which they

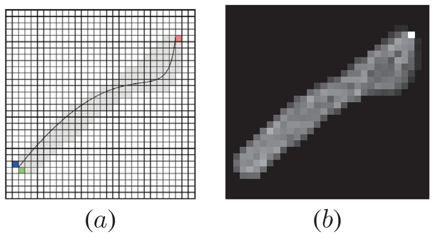
Figure 1: (a) Example grid world instance with exploration envelope (light gray). The goal state is in the top right denoted by a red square, initial location by green, and the agent by blue. (b) Corresponding visitation table. Despite having to learn a recovery rule the agent was able to focus almost all exploration effort on states within the envelope.

call a backup policy, that returns the agent to a safe region of the state space. They suggest that a backup policy can be learned off-line from previously collected data or learned on-line. However, (Hans et al. 2008) did not show how to learn the backup policy on-line, and did not apply it in the context of systematically varied exploration envelopes.

It is generally impossible to estimate an accurate transition model for states that have never been visited. Fortunately, in many problems, avoiding irrelevant or harmful states can be accomplished with a simple and consistent set of rules.

Most states in the grid world environment have similar dynamics. We took advantage of this similarity by learning state-independent transition probabilities, which were used to plan a recovery rule. To learn state-independent transition probabilities, we exploited the notion of action outcomes (Sherstov and Stone 2005). In this case, the outcomes were the relative distances traveled. The function $\kappa : S \times S \rightarrow \mathcal{O}$ maps each pair of states to a small number of outcomes $\mathcal{O}$ (e.g. move left by 1 unit). Unlikely state transitions were mapped to a single dummy outcome. We learned a state-independent action model on-line that could be applied to any state outside of ξ. For all states $s \notin \xi$ and actions $a \in A$ the transition probability to a state $s' \in S$ was estimated by

$$\hat{p}(s'|s,a) = \frac{\sum_{y \in \{x \in S | \kappa(s,x) \equiv \kappa(s,s')\}} l(s,a,y)}{\sum_{z \in S} n(z,a)} \quad (3)$$

where $\{x \in S|\kappa(s,x) \equiv \kappa(s,s')\}$ is the set of states that when paired with s belong to the same outcome class as the pair (s,s') and l and n are counters updated in algorithm 1. A similar approach could also be applied to robotic systems. Figure 1 shows that the learned recovery rule is quite effective in the grid world domain.

As a proof of concept, for well chosen exploration envelopes and our learned recovery rule, we tested how average total reward achieved by STAR-MAX is affected as the number of states in simple grid world MDPs is increased. The results, shown in figure 2, compare STAR-MAX's performance against that of R-MAX and Q-learning initialized to be generally optimistic and Q-learning constrained to ξ (Q-learning with β). STAR-MAX scales significantly better than R-MAX and Q-learning even when Q-learning is artificially restricted to the exploration envelope. This suggests that model-based exploration is important for efficient learning even when the state space is restricted. We use the same technique for learning a recovery rule in all of the following experiments.

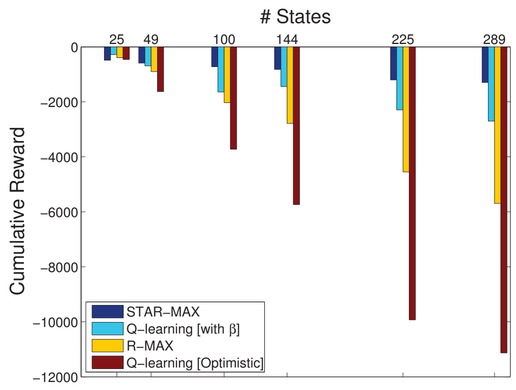

Figure 2: Average final cumulative reward achieved as the number of states in the grid world domain increase. For a good choice of exploration envelope STAR-MAX outperforms both R-MAX and Q-learning (even when Q-learning is given an oracle recovery rule β). (Note that the total reward is negative because a reward of -1 was given until the goal state was reached.)

Sample Complexity vs. Envelope Size

We considered how sample complexity grows with envelope size. Unfortunately we cannot directly measure the theoretical sample complexity, but the last time a state-action pair transitioned from insufficiently modeled to "well-modeled" (last exploration event) provides a good indication of when the agent stopped exploring.

Figure 3 shows the cumulative reward for STAR-MAX with exploration envelopes containing different numbers of states. Notice that the cumulative reward tends to decrease linearly as the number of states increases. On the other hand, the last exploration event occurs later as the number of states increases. Smaller exploration envelopes require less exploration and achieve higher cumulative reward.

In the next two experiments we consider how to learn exploration envelopes and whether STAR-MAX is sensitive to degenerate exploration envelopes.

Envelope from Demonstrations

In many problems exploration envelopes can be determined by a knowledgeable engineer. We demonstrate an important alternative here: learning an envelope from demonstrations.

We considered a figure-8 tracing task (figure 4a). This task has 1,000 states because the location of the arm's end-effector was discretized into 100 locations and time was discretized into 10 phases. Trajectories were generated for a figure-8 tracing task (figure 4a) by running Q-learning on the task for 4,000 episodes. This resulted in a visitation table (figure 4b) containing counts on the number of times each

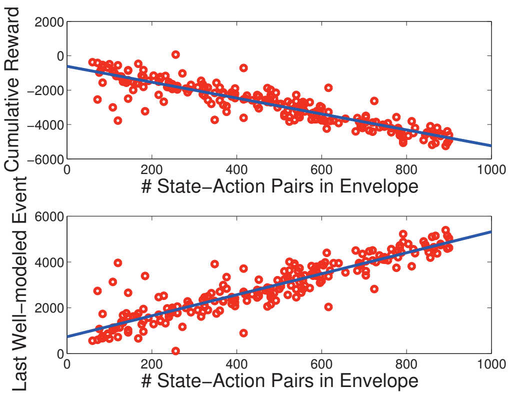

Figure 3: (Top) Cumulative reward for STAR-MAX decreases linearly as the envelope size increases for a grid world instance with 225 states and 4 actions. (Bottom) Last time a state-action pair changed to "well-modeled" increases linearly with the # states in the exploration envelope.

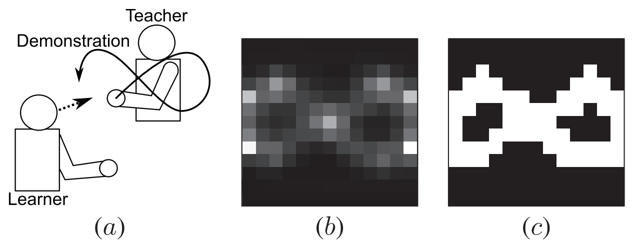

Figure 4: (a) Figure-eight demonstration. (b) Visit table of Q-learning agent. (c) Envelope derived from visit table by extracting states with number of visits greater than the 95 percentile. (Bright = High, Dark = Low Value)

state was visited. The final envelope was generated by selecting all states for inclusion that were visited greater than the 95$^{\text{th}}$ percentile (figure 4c). The results (figure 5) demonstrate that STAR-MAX is able to learn far more efficiently with the demonstration envelope than either Q-learning or R-MAX. It is also worth noting that STAR-MAX is computationally faster than R-MAX because it runs value iteration fewer times. This kind of demonstration-based exploration envelope can be naturally applied to robot training.

Degenerate Envelopes and Red Herring States

The Red Herring domain (see figure 6) is a grid world instance introduced by (Hester and Stone 2009) to demonstrate a potential weakness of the RL-DT algorithm. The space is partitioned into four rooms. The initial state is selected randomly from one of the cells in the top left room. All states produce an immediate reward of -1 except for the two "red herring" states marked by "R", which provides a reward of 0 and terminates, and the goal state marked by "G", which gives a reward of $+25$ and terminates. For the Red Herring domain we used a value of $m = 10$ to match with experimental results from other work.

Provided that ξ contains the goal state "G", STAR-MAX and R-MAX will both eventually explore the goal state in

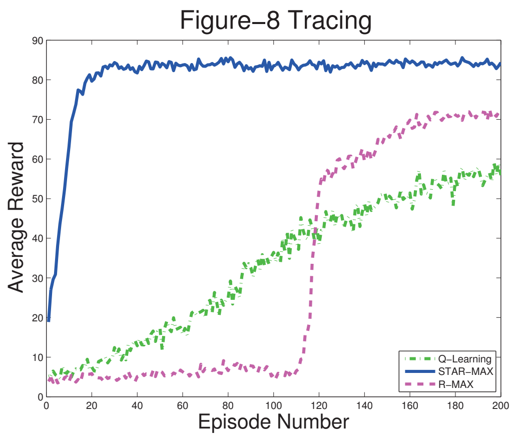

Figure 5: STAR-MAX learns a high reward policy almost immediately with the exploration envelope described in figure 4c.

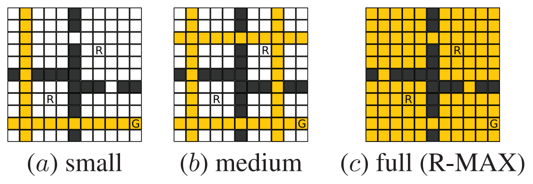

Figure 6: Red Herring domain with various exploration envelopes (yellow states). (a) The small envelope has few states and leads directly to the goal along a single path, (b) the medium envelope has two paths to the goal state, and (c) the full envelope behaves identically to R-MAX.

favor of the "red herring" states. The main difference is that STAR-MAX will have fewer states to explore.

Figure 7 shows that envelopes with more states receive less cumulative reward, which is expected. What is surprising is that envelopes constructed by randomly dropping a certain percentage of states improves cumulative rewards (statistically significant for 10% - 60% with p-values less than 0.02, see figure 8). This provides evidence that in practice STAR-MAX can learn more efficiently than R-MAX even when little is known about the environment.

Discussion

STAR-MAX aggressively and efficiently explores its environment. However, STAR-MAX does not need to explore as many states as R-MAX before STAR-MAX begins exploiting. This allows STAR-MAX to scale to large state MDPs.

The main limitation of STAR-MAX is that it requires an exploration envelope and recovery rule. However, we have shown here that a recovery rule can be learned online. There is a caveat that this solution is only plausible for MDPs where the states outside of the exploration envelope are likely to share similar dynamics. Specifying the exploration envelope is also a critical design decision. We have shown here that one can be learned from demonstration and that even naïvely generated exploration envelopes

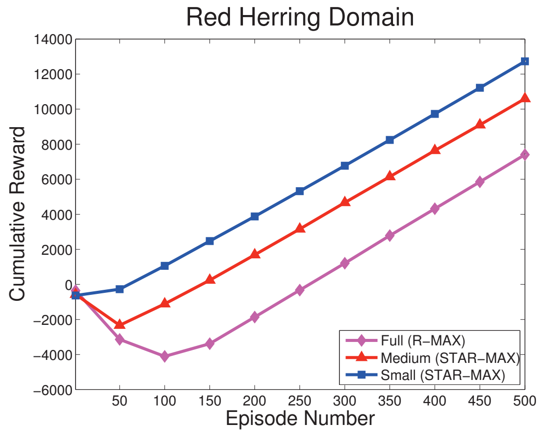

Figure 7: Cumulative reward achieved by STAR-MAX comparing exploration envelopes from figure 6.

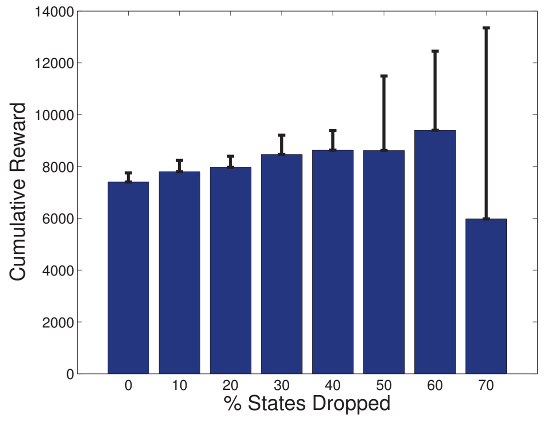

Figure 8: Average cumulative reward of STAR-MAX (n=30 trials, errorbar=std, 500 episodes/trial) increases under the Red Herring domain when a percentage of states are randomly dropped from the full exploration envelope. Performance degrades if too many states are dropped ($> 60\%$).

can improve performance.

Although we leave a formal analysis to future work, a major motivation for the development of STAR-MAX is to exploit prior knowledge (about where NOT to explore) to reduce theoretical upper bounds on sample complexity compared to R-MAX. We hypothesize that, with few additional assumptions about the exploration envelope and recovery rule, a polynomial upper bound on the sample complexity of STAR-MAX can be established that is dependent on the size of ξ (and independent of total number of states) because all transitions from states in ξ to states not in ξ can be modeled as transitions to a single fictitious state. Any proof will also require a slight modification to the notion of optimality (compared to (Kakade 2003)) since it is not reasonable to expect STAR-MAX to act optimally outside of ξ.

There are several directions for future work. For example, we can (1) apply an exploration envelope strategy to Delayed Q-learning (Strehl, Li, and Littman 2009), which is a model-free RL algorithm with polynomial sample complexity guarantees, (2) extend STAR-MAX to tasks with continuous state spaces, or (3) investigate learning ξ and β in a transfer learning setting to reduce sample complexity.

Conclusion

We have introduced a model-based RL algorithm STAR-MAX that is an extension of the popular R-MAX algorithm. Intuitively, narrowing exploration to a subset of the total space should result in faster learning. We have shown that there are multiple productive ways to select a useful exploration envelope, and that STAR-MAX is robust to degenerate choices for ξ. We expect the results from this work to lead to new insights about how RL can be scaled to handle large-scale real world problems.

References

Brafman, R. I., and Tennenholtz, M. 2002. R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3:213–231.

Brunskill, E.; Leffler, B. R.; Li, L.; Littman, M. L.; and Roy, N. 2009. Provably efficient learning with typed parametric models. *Journal of Machine Learning Research* 10:1955–1988.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76(1-2):35 – 74. Planning and Scheduling.

Hans, A.; Schneegaß, D.; Schäfer, A. M.; and Udluft, S. 2008. Safe exploration for reinforcement learning. In *European Symposium on Artificial Neural Networks*, 143–148.

Hester, T., and Stone, P. 2009. Generalized model learning for reinforcement learning in factored domains. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2369–2374.

Kakade, S. M. 2003. *On the Sample Complexity of Reinforcement Learning*. Ph.D. Dissertation, University College London.

Kearns, M., and Singh, S. 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49:209–232. 10.1023/A:1017984413808.

Leffler, B. R.; Littman, M. L.; and Edmunds, T. 2007. Efficient reinforcement learning with relocatable action models. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, 572–577. AAAI Press.

Sherstov, A. A., and Stone, P. 2005. Improving action selection in MDP's via knowledge transfer. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*.

Strehl, A. L.; Li, L.; and Littman, M. 2009. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research* 10:2413–2444.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. MIT Press.