A Gadget Problem for Medical Applications

Subhojyoti Mukherjee[*]
College of Information and Computer Science
University of Massachusetts Amherst
Massachusetts, MA 01003
`http://bio-nlp.org/index.php/people`

1 Introduction

In this report we build a gadget world for the Reinforcement Learning (RL) setup for the medical domain. A gadget problem is a simple environment which captures some of the complexities of the real-world domain which we are trying to model. We can test various RL algorithms in this gadget worlds before transitioning to the real-world higher complexity environments. The hypothesis behind creating such gadget worlds is that if an RL algorithms performs poorly in this small gadget world, it will surely perform poorly in real-world domains.

2 Some complexities of Medical Domain

Some of the difficult features that constitute the medical domain which we try to model in this gadget world are :-

- The medical environment is a partially observed environment. At any instant the physician is only exposed to some of the factors influencing the health of the patient.
- Long horizon problem where you only receive the feedback at the end of the episode or the feedbacks are very sparse in nature.
- Time constrained treatment, which requires that the effective treatment needs to be delivered within a fixed time otherwise it may result in death.
- Time varying adversarial feedbacks, which may be encountered in sudden spikes in responses from the patient to the treatments administered.

3 Notations, Assumptions and Definitions

T denotes the time horizon. We use capitalized calligraphic notations to denote sets while individual elements within the set is denoted by non-capitalized alphabets. $\mathcal{A}$ denotes the finite set of actions with individual action indexed by a such that $a = 1, \ldots, K$. We assume that the total number of actions is discrete and constant throughout the time horizon and $|\mathcal{A}| = K$. In the gridworld setting $K = 4$. We define an MDP M with the tuple $M = (\mathcal{S}, \mathcal{A}, P, d_R, d_0, \gamma)$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, P is the transition function, d_R is the reward distribution over the states indicating how the rewards are generated, d_0 is the initial state distribution and γ is the discounting factor.

4 MDP Formulation

We introduce the 10×10 gridworld Figure 1 which has the following features:-

[*]Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

1. The starting state is shown as S in green color.
2. At any grid, only four discrete actions are possible, left, right, bottom, top.
3. The obstacles are shown in red colors. When an agent hits the obstacles, it stays in the state before attempting to transition.
4. The only difference between Domain 1 and Domain 2 is the position of the death state $S_t = D$.
5. The terminal states are shown in orange. D represents the state "death" with a negative reward of $R(S_t = D, A_t = a) = -60$ while G represents the state "get well" with a time-varying reward of $R(S_t = G, A_t = a) = 60 - t, 1 \leq t \leq 60$. All the other transitions result in a reward of 0.
6. Note, that the time t is part of the representation of state as rewards are changing with time t.
7. The states are featurized by the function $\Phi : S \rightarrow R^{r+c}$, where r is the number of rows and c is the number of columns in the gridworld. So, $\Phi(s)$ is a function that maps states to vectors of features. We define $\Phi(s)$ such that for the state $s_{i,j}$, where i is the row-index and j is the column index in the grid, then $\Phi(s_{i,j})$ is the vector v such that,

   $$\begin{aligned} v_k &= 1, \text{if k = i+j} \\ &= 0 \text{ otherwise,} \end{aligned}$$

   and $k = 1, \ldots, (r + c)$ is the index of the vector v.

Next, we illustrate why these features where included in these gadget worlds and link up with our discussion on the complexities of the medical domain.

1. The state space is discrete and and the action space is also finite and discrete. We wanted to keep the gadget worlds simple.
2. Domain 2 is slightly more difficult than Doamin 1 as the path to "get well" state $S_t = G$ is more restricted in the former.
3. There is only substantial reward (positive/negative) at the end of the long episodes when the agent reaches the states either $S_t = G$ or $S_t = D$. This handles the long horizon problem.
4. Rewards are also adversarial as they are changing with time. Because of this, if the agent reaches the goal state at $t = 60$, it receives a reward of 0. Moreover, the rewards are diminishing with time indicating, that the agent has to reach the terminal "get well" state quickly.
5. The partially observed environment is captured in how we are featurizing the states. Note that $\Phi(s_{2,3})$ will have the same embedding as $\Phi(s_{3,2})$. This follows from the idea that when feature representation of states are *not* rich enough it result in a partially observed environment.
6. The obstacles, (marked in red) forces the q-value function approximator not to generalize too well. These makes the simple environment slightly more difficult to be generalized well enough.

5 Experiments

In this section, we run Q-learning and Sarsa with linear function approximation in the two gridworld domain shown in Figure 1 and Figure 2. The results of the experiments are shown in Figure 3(a) and Figure 3(b) for the domain 1 and 2 respectively. All the algorithms were averaged over 50 independent trials and each trial consisted of 6000 episodes.

Experiment 1 (Domain 1): In this experiment we use linear function approximation for both Q-Learning and Sarsa to handle this partially observed environment. From Figure 3(a) we see that Sarsa performs better than Q-Learning in this Domain and stabilizes before Q-Learning.

Experiment 12(Domain 2): In this experiment again we use linear function approximation for both Q-Learning and Sarsa to handle this partially observed environment. From Figure 3(b) we see that

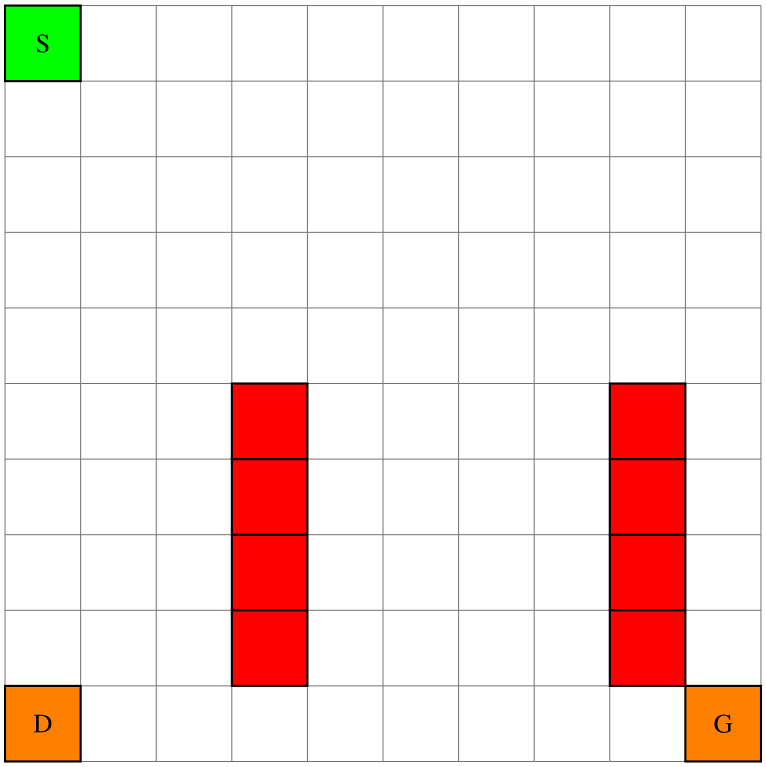


Figure 1: Medical Domain 1 Grid World

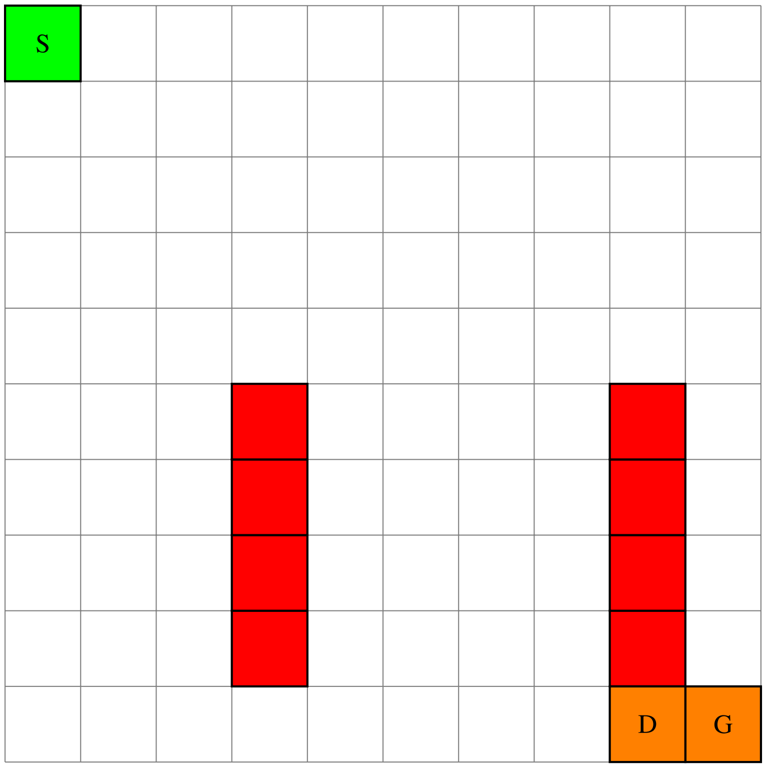
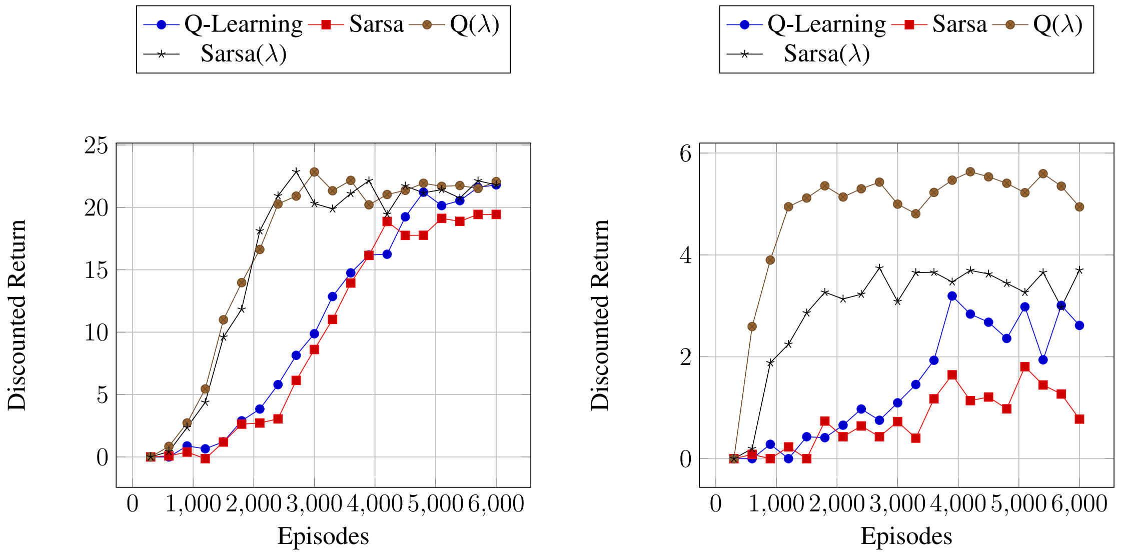


Figure 2: Medical Domain 2 Grid World

Sarsa performs worse than Q-Learning in this Domain. Infact both the algorithms does not stabilize in this experiment. This results from the fact the the entry to the state $S_t = G$ is restricted and both the algorithms spend considerable amount of time in fruitless exploration.



(a) Expt-1: 10×10 Gridworld (Domain 1)

(b) Expt-2: 10×10 Gridworld (Domain 2)

Figure 3: A comparison of the performance of various algorithms.

6 Conclusions and Future Works

In this report we formulated two gadget domains which are easy to handle and yet has sufficient complexities to handle many important and intriguing features of the real-life medical domain. We also showed that both Q-Learning and Sarsa with linear approximation fails to perform well in these domains. Future work includes proposing new algorithm that might perform better in these environments or to test planning algorithms in these domains.

References