Safe RL: Shielding against Dangerous Behaviour while Learning and Deployment

Aniket Shirke 150100012 **Akash Trehan** 150050031 **Huzefa Chasmai** 15D170013 **Samarjeet Sahoo** 150100017
CS 747 Course Project
Computer Science and Engineering Department, IIT Bombay

Abstract

Reinforcement learning algorithms find optimal policies, but they rarely guarantee safety during learning or execution phases. These safety conditions can be in the form of not allowing certain actions to be taken on certain states. While assigning large negative rewards to such actions can be one possible approach in the learning phases, but in cases where the learning takes place in a real world environment, unsafe actions can lead to disastrous consequences even during the learning phase. In this project, we present a summary of having a safety shield for reinforcement learning problems. We implement a shield in the domain of Pacman and present our results. We notice that having a safety shield does not affect the convergence of the learning algorithm. The shield also prevents the agent from taking unsafe actions during both learning and execution.

Introduction

Reinforcement Learning

In reinforcement learning, an agent interacts with the environment, obtaining observations and rewards. Through this, it attempts to learn the optimal actions it needs to take at each step. The environment is assumed to be a **Markov Decision Process (MDP)** with state space S, action space A, reward function $R : S \times A \times S \rightarrow \mathbb{R}$, transition function $T : S \times A \times S \rightarrow \mathbb{R}$ and discount factor γ. The agent is in state s^t at time t, takes an action a^t , gets reward r^t and goes to state s^{t+1}. It's objective is to maximize the expected long term reward

$$\mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + ...]$$

Reinforcement learning algorithms like SARSA and Q-learning maintain a table containing the values of Q(s, a), defined as the maximum expected long term reward obtained by starting in state s, taking action a and following the optimal policy after that. A greedy in the limit with infinite exploration (GLIE) policy based on the Q values is used by the agent. The table of Q values is updated based on the rewards obtained from the environment.

We have open sourced our implementation and experimental data at `https://github.com/CodeMaxx/Safe-RL-Pacman`.

Safe Reinforcement Learning

Rapid advances in the field of reinforcement learning have brought a new paradigm for development of autonomous controllers, which are designed to accomplish complicated tasks in dynamic and uncertain environments. When these learning agents are used in the proximity of humans, one also needs to take into account the safety aspects of the learning and execution processes. Safe Reinforcement Learning can be defined as the process of learning policies that maximize the expectation of the return in problems in which it is important to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment processes.(Garcıa and Fernández 2015)

Our Contribution

The intersection between the use of formal methods and reinforcement learning has been of research interest. Our contribution is to present our understanding of a specific research article 'Safe Reinforcement Learning via Shielding'(Alshiekh et al. 2017). We will apply the concept of shielding to the domain of Pacman and juxtapose our findings with a Pacman agent trained in the absence of a shield.

Related Work

Safe Reinforcement Learning has been an extensively researched topic and can be categorized into two approaches. The first is based on the modification of the optimality criterion, the classic discounted finite/infinite horizon, with a safety factor. The second is based on the modification of the exploration process through the incorporation of external knowledge or the guidance of a risk metric.(Garcıa and Fernández 2015)

(Alshiekh et al. 2017) builds on the latter approach, where policies are learned optimally while enforcing properties expressed in temporal logic. Given the temporal logic specification that is to be obeyed by the learning system, a shield, which is nothing but a finite state reactive system, is synthesized. The shield monitors the actions from the learner and corrects them only if the chosen action causes a violation of the specification.

In the domain of Pacman, multiple methods like search (DFS, BFS, A^*) have been explored for decision making. Reinforcement Learning Methods including Model Based and Model Free approaches have also been used as learning algorithms for training the Pacman agent. (Gnanasekaran, Faba, and An) discusses and compares the performance of Q-learning, approximate Q-learning and Deep Q-learning based on the total rewards and win-rate. Q-learning has been shown to be quite effective to find the optimal policy on small Grid. We will be applying the concept of shielding to Q-learning and approximate Q-learning and compare the results with their corresponding counterparts without shielding.

Problem Statement

- Present our comprehension of (Alshiekh et al. 2017)
- Implement and analyze the effects of the shielding method for Safe Reinforcement Learning and its applicability on the Pacman environment.

We hypothesize that the Pacman agent avoids dangerous actions under the guidance of a shield expressed in terms of a Linear Temporal Logic. In the first part, we provide a summary of safe reinforcement learning via shielding and the latter half concerns the deployment of a shield in Pacman environment.

Part I: Safe Reinforcement Learning via Shielding - A summary

In this section, we strongly recommend that the reader has some background in Automata Theory, specifically the knowledge of deterministic finite state automatons.

Technical Preliminaries

Finite-State Reactive System: A *finite-state reactive system* is a tuple $\mathcal{S} = (Q, q_0, \Sigma_I, \Sigma_O, \delta, \lambda)$ with the input alphabet Σ_I, the output alphabet Σ_O, a finite set of states Q, and the initial state $q_0 \in Q$. Function $\delta : Q \times \Sigma_I \to Q$ is a complete transition function, and $\lambda : Q \times \Sigma_I \to \Sigma_O$ is a complete output function. Given the input trace $\overline{\sigma}_I = x_0x_1... \to \Sigma_I^\infty$, the system $\mathcal{S}$ produces the output trace $\overline{\sigma}_O = \mathcal{S}(\overline{\sigma}_I) = \lambda(q_0, x_0)\lambda(q_1, x_1)... \to \Sigma_O^\infty$, where $q_{i+1} = \delta(q_i, x_i) \forall i \geq 0$.
The input and output traces can be merged to the trace of $\mathcal{S}$ over the alphabet $\Sigma_I \times \Sigma_O$, which is defined as $\overline{\sigma} = (x_0, \lambda(q_0, x_0))(x_1, \lambda(q_1, x_1))... \to (\Sigma_I \times \Sigma_O)^\omega$.

Game: A (2-player, alternating) *game* is a tuple $\mathcal{G} = (G, g_0, \Sigma_I, \Sigma_O, \delta, win)$, where G is a finite set of game states, $g_0 \in G$ is the initial state, $\delta : G \times \Sigma_I \times \Sigma_O \to G$ is a complete transition function, and $win : G^\omega \to B$ is a winning condition. The game is played by the system and the environment. In every state $g \in G$ (starting with g_0), the environment chooses an input $\sigma_I \in \Sigma_I$, and then the system chooses some output $\sigma_O \in \Sigma_O$. These choices by the system and the environment define the next state $g' = \delta(g, \sigma_I, \sigma_O)$, and so on. The resulting (infinite) sequence $\overline{g} = g_0g_1. . .$ is called a *play*. A play is *won* by the system *iff* $win(g)$ is true. A (memory less) strategy for the system is a function $\rho : G \times \Sigma_I \to \Sigma_O$. A strategy is winning for the system if all plays $\overline{g}$ that can be constructed when defining the outputs using the strategy (for the respective previous state in the play and the previous environment player move) satisfy $win(\overline{g})$. The *winning region* W is the set of states from which a winning strategy exists.

Safety Game: A *safety game* defines win via a set $F^g \subseteq G$ of safe states: $win(g_0g_1 \, ...)$ is true iff $\forall i \geq 0. \; g_i \in F^g$, i.e., if only safe states are visited.

Inputs :

1. A Safety automaton $\phi^s = (Q, q_0, \Sigma, \delta, F)$, where $\Sigma = L \times \mathcal{A}$, $\delta = Q \times \Sigma \to Q$ and $F \subseteq Q$, set of safe states. This specification is obtained by converting an LTL formula into a safety specification. (Refer to the watertank problem in (Alshiekh et al. 2017))
2. An MDP $\mathcal{M} = (S, s_I, \mathcal{A}, P, R)$ and an MDP observer function $f : S \to L$ for some set L. A deterministic safety word automaton $\phi^\mathcal{M} = (Q_\mathcal{M}, q_0, \mathcal{M}, \Sigma, \delta_\mathcal{M}, F_\mathcal{M})$ an abstraction of $\mathcal{M}$ if $\Sigma = \mathcal{A} \times L$ and for every trace $s_0s_1s_2... \in S^\omega$ with the corresponding action sequence $a_0a_1... \in \mathcal{A}^\omega$ of the MDP, for every automaton run $q = q_0q_1... \in Q_\mathcal{M}^\omega$ of $\phi^\mathcal{M}$ with $q_{i+1} = \delta_\mathcal{M}(q_i, (l_i, a_i))$ for $l_i = L(s_i)$ and all $i \in N$, we have that q always stays in $F_\mathcal{M}$.

Shield Synthesis Algorithm

S is computed by reactive synthesis from ϕ^s and an MDP abstraction ϕ^M that represents the environment in which the agent shall operate.

1. We translate ϕ^s and $\phi^\mathcal{M}$ to a safety game $G = (G, g_0, \Sigma_I, \Sigma_O, \delta, \mathcal{F}^g)$ between two players. In the game, the environment player chooses the next observations from the MDP state (i.e., elements from $\mathcal{L}$), and the system chooses the next action. Formally, G has the following components: $G = \mathcal{Q} \times \mathcal{Q}_\mathcal{M}$, $g_0 = (q_0, q_{0,\mathcal{M}})$, $\Sigma_I = L$, $\Sigma_O = \mathcal{A}$, $\delta((q, q^\mathcal{M}), l, a) = (\delta(q, (l, a)), \delta^\mathcal{M}(q, (l, a)))$, for all $(q, q^\mathcal{M}) \in G$, $l \in L$, $a \in A$, and $F^g = (F \times Q^\mathcal{M}) \cup (Q \times (Q^\mathcal{M} \, F^\mathcal{M}))$.
2. Next, we compute the winning region $W \subseteq F^g$ of G as described by (Bloem et al. 2015).
3. We translate G and W to a reactive system $S = (\mathcal{Q}_S, q_{0,S}, \Sigma_{I,S}, \Sigma_{O,S}, \delta_S, \lambda_S)$ that constitutes the shield.
   The shield has the following components: $\mathcal{Q}_S = G, q_{0,S} = (q_0, q_{0,M}), \Sigma_{I,S} = L \times \mathcal{A}, \Sigma_{O,S} = A, \delta_S(g, (l, a)) = \delta(g, (l, \lambda_S(g, (l, a))))$
   for all $g \in G, l \in L, a \in mathcalA$, and

$$\lambda_S(g, l, a) = \begin{cases} \text{a} & \text{if} \delta(g, (l, a)) \in W \\ \text{a}' & \text{if} \delta(g, (l, a)) \notin W \text{for some arbitrary} \\ & \text{but fixed a}'\text{with } \delta(g, (l, a)) \in W \end{cases}$$

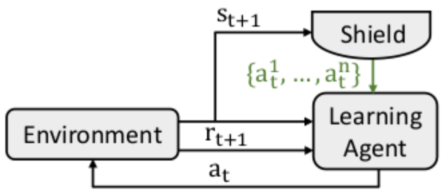

Figure 1: Preemptive Shielding

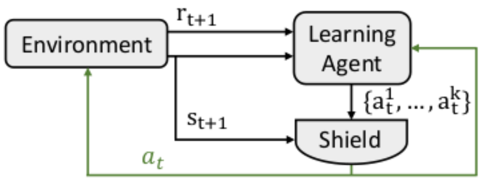

Figure 2: Post-posed Shielding

Types of Shielding

Preemptive Shielding: The interaction between the agent, the environment and the shield is as follows: At every time step t, the shield computes a set of all safe actions $a_t^1, ..., a_t^k$, i.e., it takes the set of all actions available, and removes all unsafe actions that would violate the safety specification ϕ^s . The agent receives this list from the shield, and picks an action $a_t \in \{a_t^1, ..., a_t^k\}$ from it. The environment executes action a_t , moves to a next state s_{t+1} , and provides the reward r_{t+1} . The task of the shield is basically to modify the set of available actions of the agent in every time step such that only safe actions remain.

Post-posed Shielding: The shield monitors the actions of the agent, and substitutes the selected actions by safe actions whenever this is necessary to prevent the violation of ϕ^s . In each step t, the agent selects an action a_t^1 . The shield forwards a_t^1 to the environment, i.e., $a_t = a_t^1$. Only if a_t^1 is unsafe with respect to ϕ^s , the shield selects a different safe action a_t , a_t^1 instead. The environment executes a_t , moves to s_{t+1} and provides r_{t+1} . The agent receives a_t and r_{t+1} , and performs policy updates based on that information. For the executed action a_t , the agent updates its policy using $rt + 1$. We can either assign a punishment r'_{t+1} to a_t^1 or the reward r_{t+1} to a_t^1.

Comments

1. In essence, we are generating a shield, which is a finite reactive system, given a safety automaton and an abstraction of the underlying MDP.
2. The shield S enforces two properties: correctness and minimum interference. S enforces correctness against a given safety specification ϕ^s. S restricts the agent as rarely as possible, thereby interfering to the minimum extent.
3. The algorithm involves product construction of the safety and abstraction automaton. Hence in the case of continuous state space, the shield synthesis algorithm has an unreasonable time complexity.
4. According to (Garcıa and Fernández 2015), this approach to Safe Reinforcement Learning can be classified as a *Teacher Advice* method, where the shield assists the agent during the learning phase whenever it feels it is necessary.
5. One of the key feature of this approach is that the shield is necessary even after the learning phase is over. Consider the example of post-posed shielding, if we assign a reward whenever the agent takes an unsafe action a_t (which is rectified by the shield to a'_t), then it is imperative to maintain the shield during the execution phase as the agent has associated a positive reward with a_t. It is risky to remove the shield as the mapping of a_t to some safe action a'_t is implicitly maintained by the shield and the agent is oblivious to the presence of the shield.

Part II: Shielding in the Pacman Domain

In the previous section, (Alshiekh et al. 2017) proposed an algorithm for the automated synthesis of shields for given temporal logic specifications. Even though the inner working of a learning algorithm is often complex, the safety criteria may still be enforced by possibly simpler means. Shielding exploits this possibility.

We now integrate the shielding approach in the common domain of Pacman.

Method

Environment Construction

We employed the Pacman agents environment procured from the open sourced code provided by (Berkeley). The environments tested on are the following :

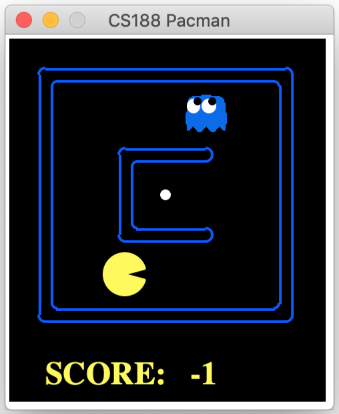

Figure 3: Small Grid

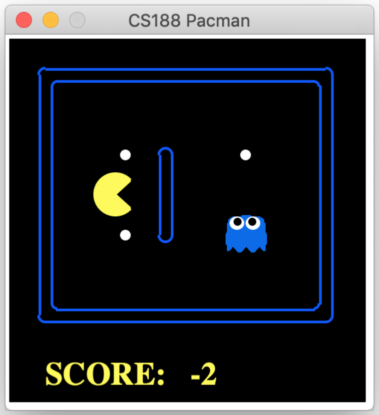

Figure 4: Medium Grid

The following are the parameters of the MDP for Pacman :

- **State** : The environment of the Pacman agent is the grid with the position of the Pacman and the ghost along with the positions of the walls and the positions of the food and energy pellets.
- **Actions** : North, South, East, West
- **Rewards** : -500 for dying, -1 for each passing time step, +500 for winning, +10 for eating a food pellet or an energy pellet
- **Transitions** : for each action the probability of going to a particular state based on action. In our case, the MDP is deterministic with respect to the actions taken.

Learning Agents

We will be using the learning algorithms - Q-learning and approximate Q-learning with and without Shielding and compare the average reward acquired between the two implementations during learning as well as in the deployment stage.
1. The updation step for the Q-learning agent:

```
$q_{max} \leftarrow \text{COMPUTEVALUEFROMQVALUES}(s_{t+1})$
$\text{Q}(s_t, a_t) \leftarrow (1-\alpha) * \text{Q}(s_t, a_t) + \alpha * (r_t + \gamma * q_{max})$
```

2. The updation step for the approximate Q-learning agent:

```
$q_{max} \leftarrow \text{COMPUTEVALUEFROMQVALUES}(s_{t+1})$
$\delta \leftarrow r_t + \gamma * q_{max} - \text{GETQVALUE}(s_t, a_t)$
$w \leftarrow \text{GETWEIGHTS}()$
$f \leftarrow \text{GETFEATURES}(s_t, a_t)$
for $i$ in $len(f)$ do
    $w[i] \leftarrow w[i] + \alpha * \delta * f[i]$
end for
```

Shield Synthesis

We adopted a Post-posed Shielding strategy with a punishment of -500 awarded to an unsafe action. We manually constructed a shield that encodes the Linear Temporal Logic that actions leading the Pacman to reach within a Manhattan distance of 2 from the Ghost is considered to be an "unsafe action". The safety specification followed by the Pacman agent is a linear temporal logic formula which is as follows:

$\neg \Box \Diamond \Diamond \, DeadState$
It is not the case that for all time instances, the next to next state is a dead state

Metrics

A. Average Score vs Episode: The average reward received so far plotted against the number of episodes

B. Average score windowed vs Episode: The average score calculated across a window of 10 episodes vs the number of episodes.

C. Average Score vs Time: The average score calculated versus the time taken for the execution.

D. Losses vs Episode: Losses are the number of occurrences when the Pacman has been eaten by the ghost. This plot calculates the losses against the number of episodes.

E. Losses windowed vs Episode: This plot is against the losses in a window of 10 learning versus the episode number.

F. Unsafe actions windowed vs Episode: Unsafe actions in the context of shielding is an action that leads to a state which is at a Manhattan distance of less than 2 from the ghost. The reason behind keeping a distance of 2 is that this is an adversarial game in which the ghost takes an action after Pacman has taken an action. We would like to call any action unsafe that can lead to a state where the Pacman can make a move which can kill the Pacman.

Results and Discussion

We conducted experiments for 2 learning agents (Q-learning and Approximate Q-learning), 2 counts of episodes (1000 and 2000), 2 environments (small grid and medium grid), 2 scenarios (with and without shielding) and generated plots for 6 metrics, thereby resulting in 96 graphs. We present some of our results in this section and discuss some of our insights.

Average Score vs Episodes

As we can see in figure 5, the average score for the normal case starts from a hugely negative value and increases gradually with the number of episodes as the agent learns the optimal policy. In the case of shield however, we see that the average score starts off from a positive value, increases quickly and saturates at a high value after which it increases very slightly with time. This is to be expected because the shield protects Pacman by preventing it from taking any action which can cause potential death. Hence, it ends up winning way earlier, losing only in the rare cases where Pacman gets trapped between the walls and has no safe actions at all.

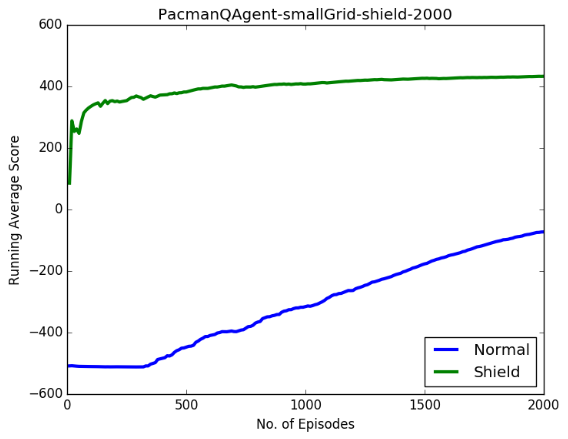

Figure 5: Q Learning: Average Scores vs Episodes

Average Score vs Time

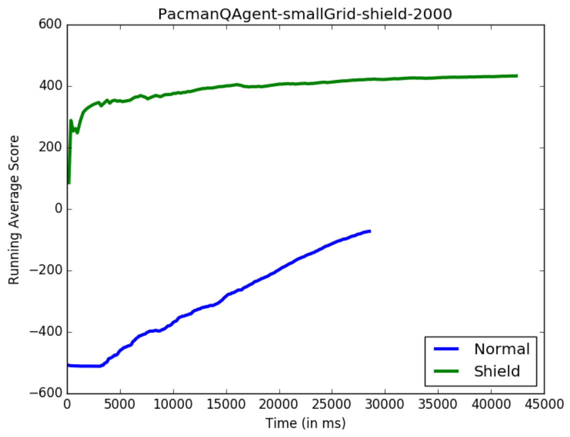

Figure 6: Q Learning: Average Score vs Time

Figure 6 shows the average score against the real time (in milliseconds) for the same instance as in figure 5. As we can see, the time taken for the normal case to complete 2000 episodes is much lesser. This is because in case of shield, the Pacman stays alive for a longer duration due to the protection provided by the shield. Also, initially, when the Pacman hasn't yet learnt the advantage of eating food pellets, it roams around aimlessly while just avoiding ghosts because of the shield, leading to longer episode times. Also, in case of having taken an unsafe action, we have multiple updates of q values for that step, by updating for q values of all unsafe actions. We have adopted the convention of giving negative rewards for the case of unsafe actions. But since that does not guarantee that the optimal policy learnt has only safe actions, we will still need the shield for protection in the deployment phase. This convention is as in the (Alshiekh et al. 2017).

Average score windowed vs Episode

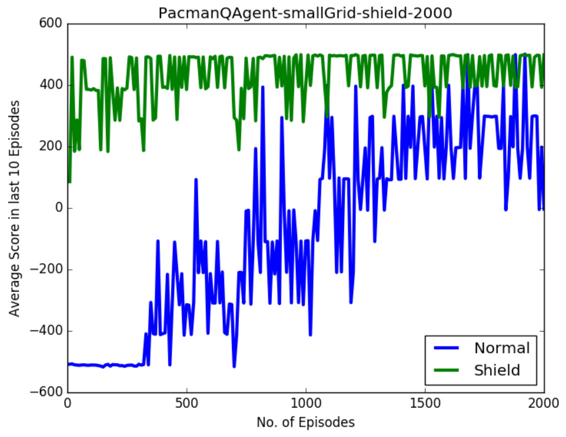

Figure 7: Q Learning: Average score in the last 10 episodes vs Episode

In figure 7, we have plotted the average score of the most recent 10 episodes versus the number of episodes. As we can expect, the score increases gradually from a very low value in the normal case. But, again in our case of the shield, we see that the average score increases rapidly in the first few episodes only and remains mostly constant thereafter. The deviation decreases with the number of episodes because we keep on learning leading to lesser and lesser losses and hence more consistent higher scores.

Total Losses vs Episode

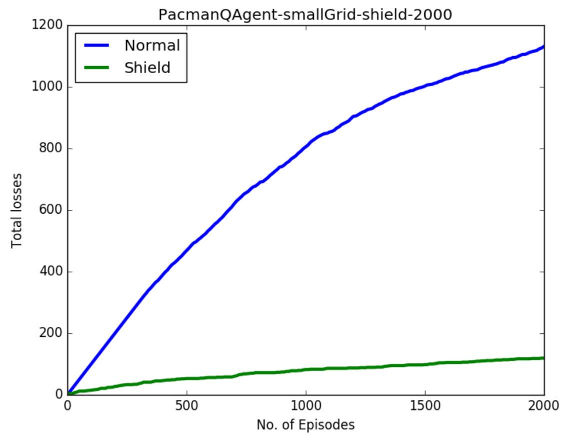

Figure 8: Q Learning: Total Losses vs Episode

As seen from figure 8, which is the plot of the total losses versus the number of episodes, the total losses in the case of shield was very less right from the initial learning phase, which was the main aim of Safe Reinforcement Learning. There are a few losses still because there are still states where the Pacman will be trapped , which is difficult to be avoided. The plot for the Q-learning agent has a decreasing slope since the agents learns policies to avoid these losses.

Losses windowed vs Episode

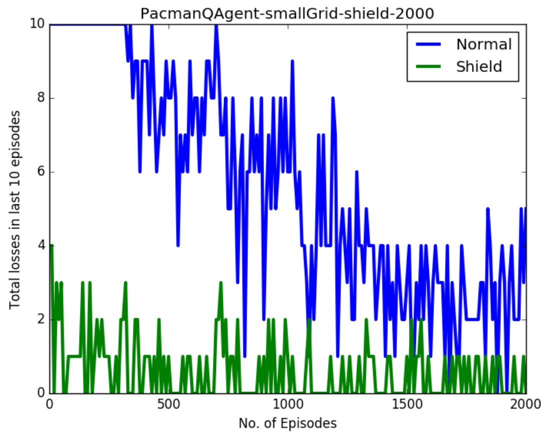

Figure 9: Q Learning: Total Loss in last 10 episodes vs Episode

In figure 9, we have plotted the losses calculated over the 10 most recent windows, and as expected the values decrease with time in the normal case as the agent learns more and more. In the shielded case, however, we see that the losses are quite small from the beginning itself, and decrease slightly with time. The fact that the losses are quite small right from the start indicates the effectiveness of our shield, and the slight decrease with time indicates that the agent keeps learning as well. The windowed graphs are a bit noisy since they are plotting the values averaged over the last 10 episodes as opposed to the total averaged in the earlier cases.

Unsafe actions windowed vs Episode

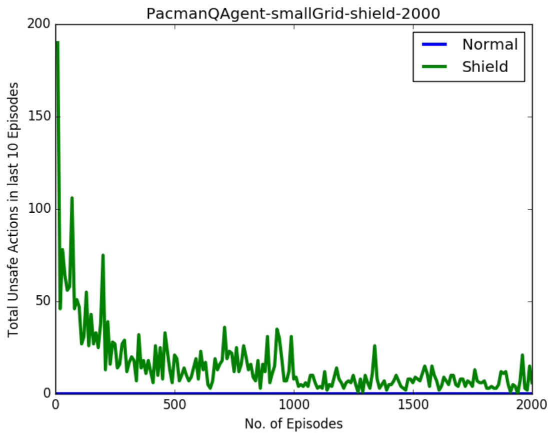

Figure 10: Q Learning: Unsafe actions in the last 10 episodes vs episodes

Figure 10 shows that the total unsafe actions taken by the shield algorithm goes on decreasing drastically during the initial learning stages. This is because the agent learns not to take these unsafe discarded actions because of the negative rewards associated with them while simultaneously learning to take more rewarding actions at a particular state. Furthermore as we can see from the values of the unsafe actions at episodes exceeding the 1000^{th} episode, the average unsafe actions taken is around 1 per episode. This reinforces the fact that the shield agent is learning not to take the actions which have been discarded by our shield but still explores certain unsafe actions even after considerable learning, i.e. in the deployment phase. This proves that there is a need of the shield even in the deployment stage.

Results for Approximate Q-learning

We are providing the corresponding graphs for the experiments done with the approximate Q-learning agents for both the shielded and unshielded agents. As we can see, the general pattern trend is similar to the Q-Learning case. The only major difference is that the learning is faster in case of approximate Q learning. This can be confirmed by observing that the average scores and losses for the normal case catch up with the shielded case.

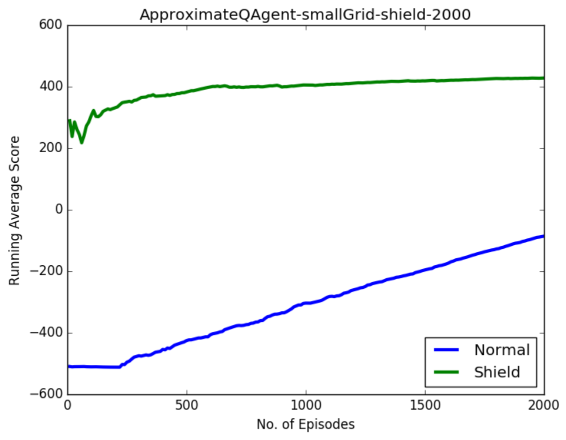

Figure 11: Approximate Q Learning: Average Scores vs Episodes

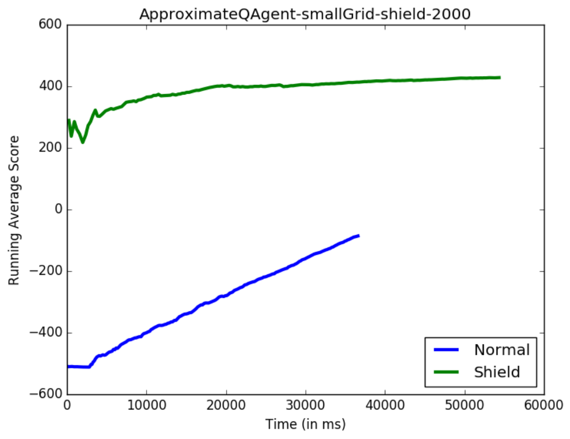

Figure 12: Approximate Q Learning: Average Score vs Time

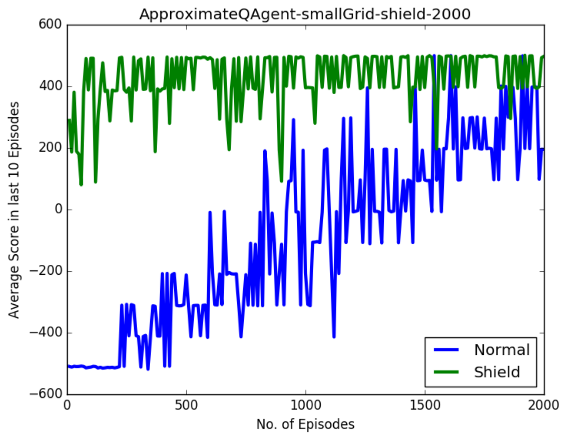

Figure 13: Approximate Q Learning: Average score in the last 10 episodes vs Episode

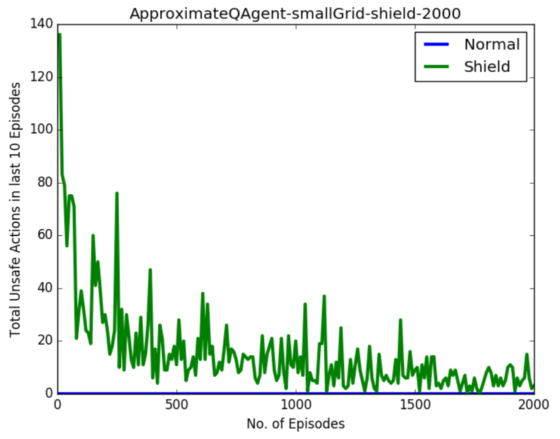

Figure 16: Approximate Q Learning: Unsafe actions in the last 10 episodes vs episodes

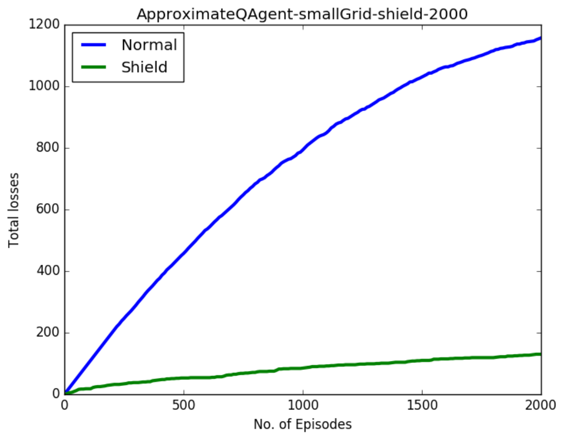

Figure 14: Approximate Q Learning: Total Losses vs Episode

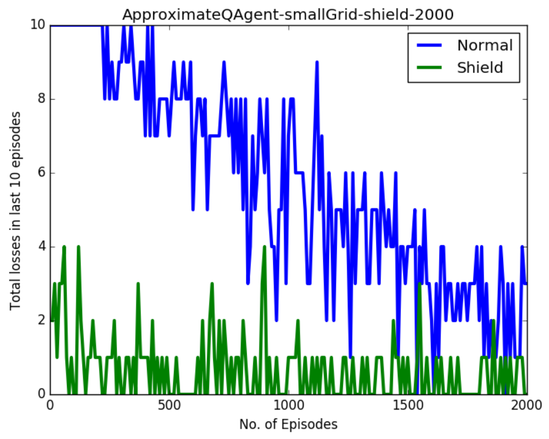

Figure 15: Approximate Q Learning: Total Loss in last 10 episodes vs Episode

Conclusion

Based on the results of our experiments, we can conclude the following: Learning with a shield affects neither the correctness nor the final convergence of the algorithm. The shield we have implemented satisfies the requirement of minimum interference too, by changing actions only if absolutely necessary. The shield is highly effective in preventing the choice of unsafe actions, the only case of failure is when all the available actions are unsafe corresponding to the agent being in a trapped state. But just like many other methods, it has its drawbacks. The time taken to learn a good policy is more compared to the unshielded case. Also, it increases the complexity of the learning algorithm significantly. Finally, the biggest advantage of having a shield is preventing potential real life agents from reaching dangerous states in both the learning and deployment phase.

References

[Alshiekh et al. 2017] Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2017. Safe reinforcement learning via shielding. *arXiv preprint arXiv:1708.08611*.

[Berkeley] Berkeley, U. Uc berkeley, cs 188 intro to ai – course materials , pacman. `http://ai.berkeley.edu/reinforcement.html`. Accessed: 2018-10-30.

[Bloem et al. 2015] Bloem, R.; Könighofer, B.; Könighofer, R.; and Wang, C. 2015. Shield synthesis. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 533–548. Springer.

[Garcıa and Fernández 2015] Garcıa, J., and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16(1):1437–1480.

[Gnanasekaran, Faba, and An] Gnanasekaran, A.; Faba, J. F.; and An, J. Reinforcement learning in pacman.

Acknowledgments

We would like to thank Prof. Shivaram Kalyanakrishnan, CSE Department, Indian Institute of Technology Bombay who encouraged us to pursue this difficult and relatively unexplored topic.