SSU : Assignment 5
Windy world, Reinforcement learning

Teymur Azayev

1 Overview

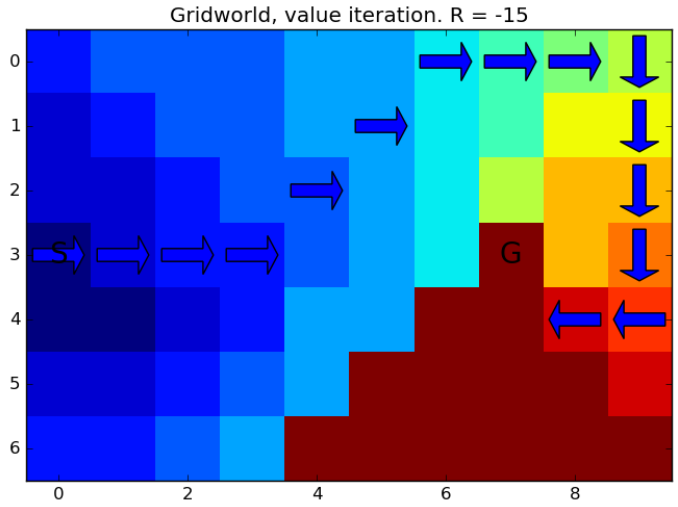
This document is a short report for the fifth assignment from the subject. It includes a theoretical overview of the task, an explanation of the implementation, results and a short discussion.

2 Task

The task is to find an optimal policy function in the windy griworld environment. The task is posed as a standard reinforcement learning problem for the MDP defined by the gridworld environment. We will take a look at using a few different algorithms for finding the optimal policy, both directly and indirectly.

3 Value iteration

We will first use a value iteration approach to finding the optimal Q-function (action value function). The algorithm converges in 683 epochs to the optimal Q-value, meaning the Q-matrix for which we get the maximal reward out of our MDP, in this case -15. We use a maximum of 100 iterations per epoch. Below is the Q matrix for each of the actions in order and the policy plotted in the gridworld.



Action: Left

$$\begin{bmatrix} -13. & -13. & -12. & -11. & -11. & -10. & -10. & -9. & -8. & -7. \\ -14. & -13. & -12. & -12. & -11. & -11. & -10. & -9. & -9. & -6. \\ -14. & -14. & -13. & -12. & -12. & -11. & -10. & -6. & -9. & -5. \\ -15. & -14. & -13. & -13. & -12. & -12. & -10. & 0. & -7. & -5. \\ -15. & -15. & -14. & -13. & -12. & -11. & 0. & -8. & -1. & -2. \\ -14. & -14. & -13. & -12. & -10. & 0. & 0. & -7. & -1. & -1. \\ -13. & -13. & -13. & -12. & 0. & 0. & 0. & 0. & -1. & -1. \end{bmatrix}$$

Action: Right

$$\begin{bmatrix} -12. & -12. & -11. & -11. & -10. & -10. & -9. & -8. & -7. & -7. \\ -13. & -12. & -11. & -11. & -10. & -10. & -9. & -8. & -5. & -6. \\ -14. & -13. & -12. & -11. & -11. & -10. & -9. & -6. & -5. & -5. \\ -14. & -13. & -12. & -12. & -11. & -10. & -9. & 0. & -5. & -4. \\ -15. & -14. & -13. & -12. & -11. & -10. & 0. & 0. & -3. & -2. \\ -14. & -13. & -12. & -11. & -10. & 0. & 0. & 0. & 0. & -1. \\ -12. & -12. & -11. & -10. & 0. & 0. & 0. & 0. & 0. & 0. \end{bmatrix}$$

Action: Up

$$\begin{bmatrix} -12. & -11. & -12. & -11. & -11. & -11. & -9. & -9. & -7. & -7. \\ -13. & -12. & -11. & -11. & -10. & -10. & -9. & -9. & -7. & -6. \\ -14. & -13. & -12. & -11. & -11. & -10. & -10. & -8. & -5. & -6. \\ -14. & -14. & -13. & -12. & -11. & -11. & -9. & 0. & -5. & -4. \\ -15. & -14. & -13. & -12. & -12. & -11. & 0. & 0. & -5. & -2. \\ -14. & -14. & -12. & -12. & -11. & 0. & 0. & 0. & 0. & -2. \\ -12. & -12. & -12. & -11. & 0. & 0. & 0. & 0. & 0. & 0. \end{bmatrix}$$

Action: Down

$$\begin{bmatrix} -12. & -12. & -11. & -12. & -10. & -10. & -10. & -8. & -8. & -6. \\ -14. & -14. & -13. & -12. & -10. & -10. & -9. & -8. & -6. & -5. \\ -13. & -13. & -12. & -12. & -12. & -10. & -10. & -6. & -4. & -4. \\ -15. & -13. & -12. & -12. & -12. & -10. & -10. & 0. & -4. & -3. \\ -14. & -14. & -13. & -12. & -10. & -10. & 0. & 0. & -2. & -2. \\ -13. & -13. & -12. & -12. & -10. & 0. & 0. & 0. & 0. & -1. \\ -12. & -12. & -12. & -10. & 0. & 0. & 0. & 0. & 0. & 0. \end{bmatrix}$$

4 SARSA

For the SARSA algorithm we use a greedy policy with eploration noise coefficients ϵ in ranges of $\{0.01, 0.1, 0.3\}$ and learning rates α of $\{0.01, 0.1, 0.3\}$. Below are the plots for each setting. We can see that an ϵ of 0.01 gives the most stable results, with higher epsilon giving divergent behavior for higher learning rates. We can also note that a learning rate of 0.01 is too little and take too long to converge.

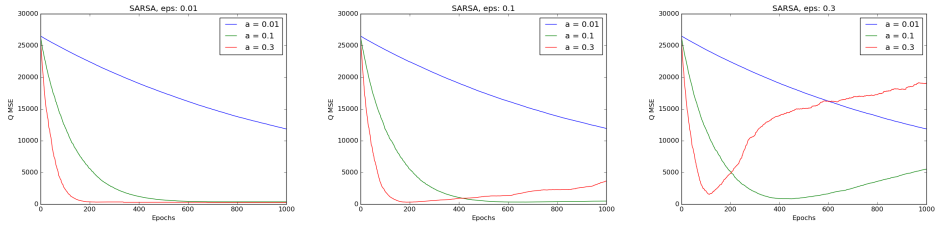


Figure 1: SARSA convergence

5 Q-learning

For the Q-learning (SARSA-max) algorithm we also use a greedy policy with eploration noise coefficients ϵ in ranges of $\{0.01, 0.1, 0.3\}$ and learning rates α of $\{0.01, 0.1, 0.3\}$. Below are the plots for each setting. We can see that in contrast to SARSA, we can push the learning rate to be much higher in Q-learning and get faster convergence rates without divergent properties as we saw in SARSA for high learning rates. It is also to be noted that Q-learning is less sensitive to the exploration noise parameter.

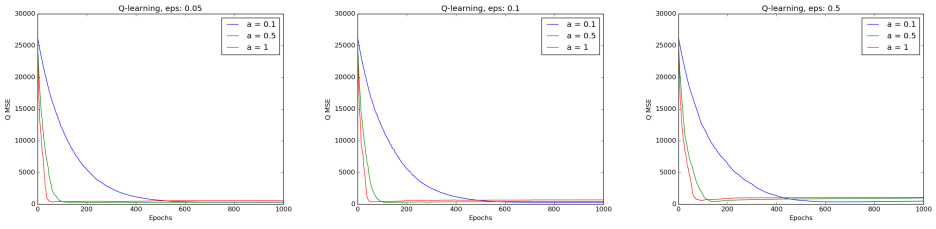


Figure 2: Q-learning convergence

6 Discussion

We saw that we can find the optimal Q matrix easily using the value iteration method. This method is unfortunately limited to smaller discrete environments such as the one in this assignment and takes quite a while to converge. Using SARSA and Q-learning algorithms we can get approximatly optimal Q-value functions (matrices) using ϵ-greedy exploration and variable learning rates. It can be seen that Q-learning is superior to SARSA in this environment due to the ability to use a higher learning rate and insensitivity to the exploration parameter. The policy iteration algorithm was also implemented but unfortunately due to a bug could not be made to work so we cannot compare the results.

7 Conclusion

This concludes the report on this task.