Programming Assignment 4: Maze Task

Robert D. Grant

1 Introduction

In this assignment I experiment with three learning algorithms in a simple discrete maze environment: the 9x9 maze shown in Figure 1. In this figure, white squares represent walls and black squares represent free spaces. In all experiments, the goal square is in (7,7).

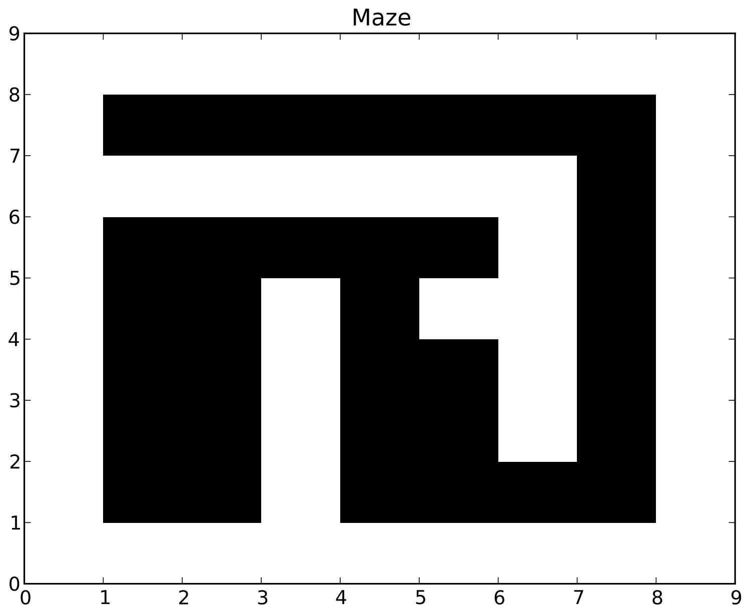

Figure 1: Maze

My learning agents use SARSA(0), Q(0), and Q(λ) with ϵ-greedy exploration and the following parameters (where applicable):

$$\alpha = 0.5 \tag{1}$$
$$\gamma = 0.99 \tag{2}$$
$$\epsilon = 0.3 \tag{3}$$
$$\epsilon_{\text{decay}} = 0.9999 \tag{4}$$
$$\lambda = 0.9 \tag{5}$$

ϵ is multiplied by ϵ_{decay} after every learning step. I use the algorithm implementations and experimental framework from PyBrain[1]. By inspection, the Q(λ) variant implemented appears to be Peng's[2].

An agent receives a reward of 1.0 for reaching the goal state and a reward of 0.0 at every other step. Running into walls does not result in an explicit penalty, it is implicitly penalized through the discount factor γ. This task is modeled as a continuing, discounted task; after an agent reaches the goal state, it is initialized to a new random start state. Unless explicitly stated, I optimistically initialize all action-values to 1.0.

2 State-Value Tables

First, I inspect the action-value tables of the three algorithms. In Figures 2, 3, and 4, I show a representation of the maximal action-value for each state after one "episode"; in other words, I run the algorithm until the first reward is received and show the resulting update. In these figures I do not initialize optimistically to make it more clear what the algorithms are doing.

In Figures 2 and 3, only the state previous to the goal state receives an update, whereas in Figure 4, several states previous to the goal state are updated after only one episode. This is as expected, and indicates that Q(λ) should converge more quickly to the optimal policy.

In Figures 5, 6, and 7, I show the same representation after 5000 steps, or "interactions" with the maze. As expected, Q(λ) seems to have already converged to a reasonable representation of the maze, with less complete representations computed by Q(0) and then SARSA.

3 Average Cumulative Reward

In this section I present graphs of the cumulative reward of each algorithm, averaged over five agents.

In Figure 8 I show the cumulative reward up to 5000 interactions. Q(λ) takes an early lead, and SARSA starts off dominating Q(0). This is expected, as Q(λ) is especially suited to this task; "On tasks with many steps per episode, or many steps within the half-life of discounting, it appears significantly better to use eligibility traces than not to", ([2], Section 7.11). SARSA dominates Q in the beginning because it is an on-policy method, and it's strategy takes ϵ into account.

This ordering changes in Figures 9, 10, and 11. As ϵ decreases due to the ϵ_{decay} factor, Q(0) is able to make better decisions than SARSA, as it has already learned (close to) the true optimal policy. SARSA(0) and Q(0) seem to converge to the same slope (policy), though Q(0)'s early lead means a permanent, constant, gap between their cumulative performance.

I do not know why SARSA(0) and Q(0) dominate Q(λ) in the long term. As described in Sutton and Barto, Peng's Q(λ) should converge to the optimal policy if the policy is gradually made more greedy, which I am doing here using the ϵ_{decay} factor. Looking at its slope on the longest-term plot, Figure 11, it appears that Q(λ) has converged to a suboptimal policy. There may be an error in the PyBrain implementation of Q(λ), my usage of it, or perhaps the ϵ_{decay} factor is not being used (though I do specify it). I have investigated all of these possibilities in the time I have had available, but I do not yet have an answer.

References

[1] Tom Schaul, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rückstieß, and Jürgen Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 2010.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An Introduction.* MIT Press, 2005.

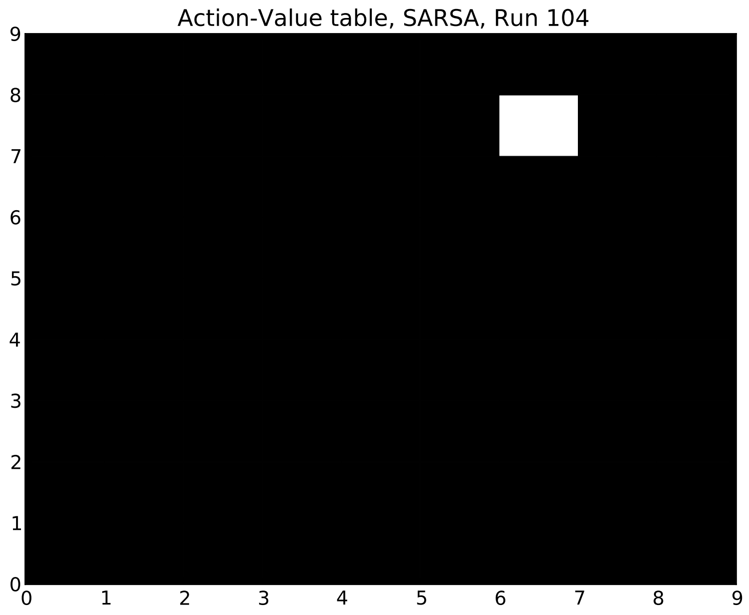

Figure 2: SARSA, Single Episode

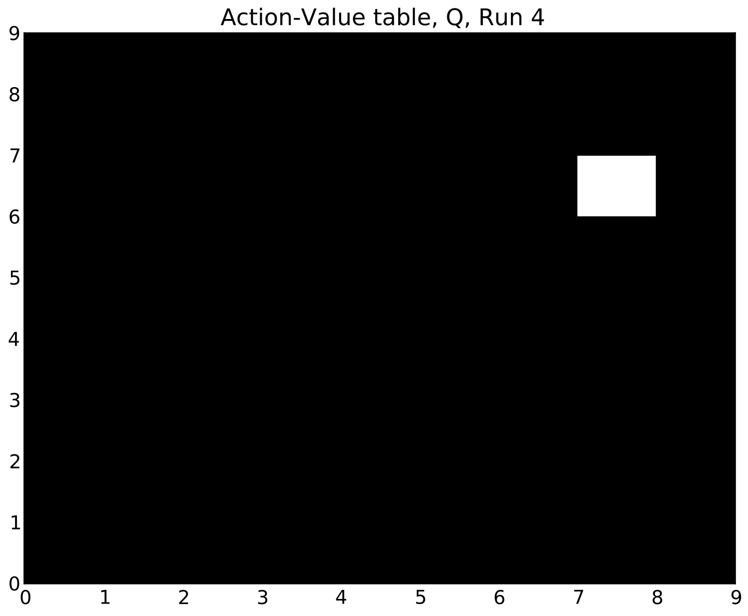

Figure 3: Q(0), Single Episode

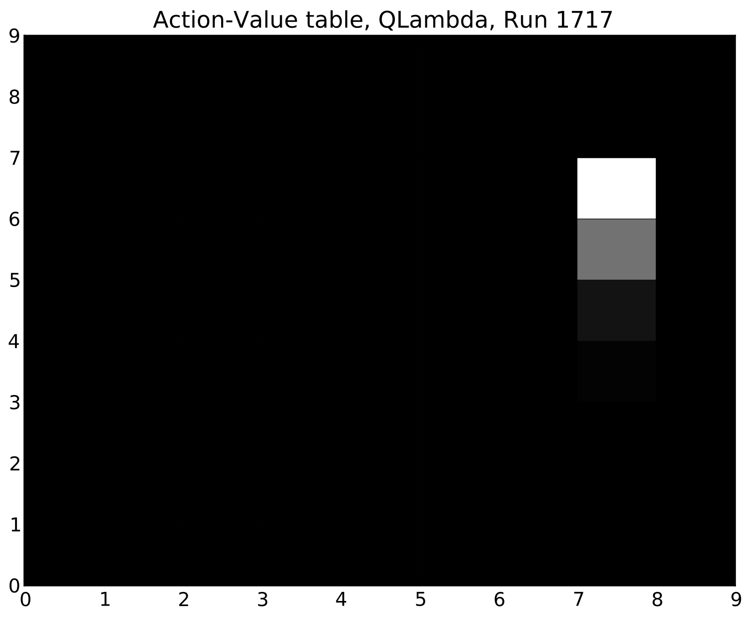

Figure 4: Q(λ), Single Episode

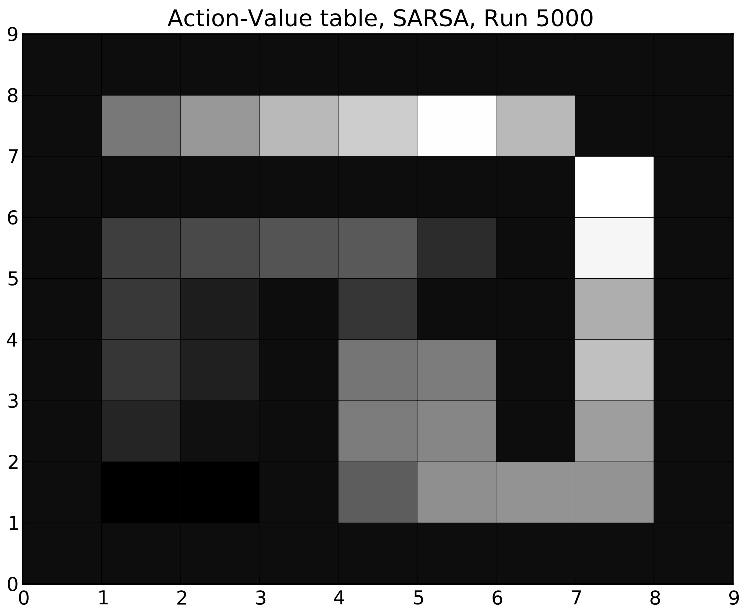

Figure 5: SARSA, 5000 Interactions

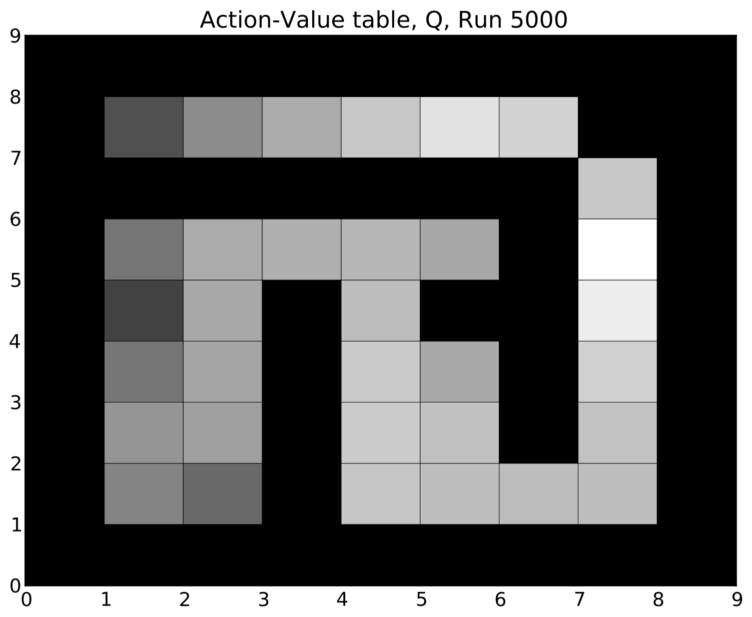

Figure 6: Q(0), 5000 Interactions

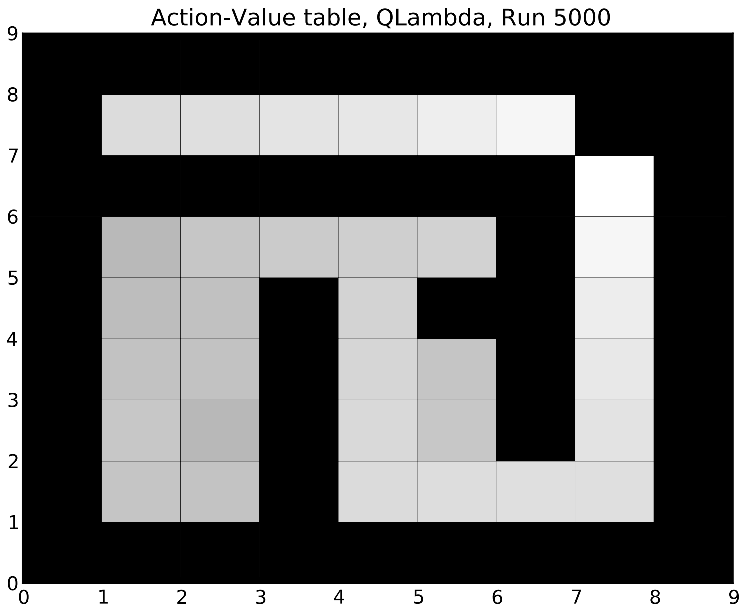

Figure 7: Q(λ), 5000 Interactions

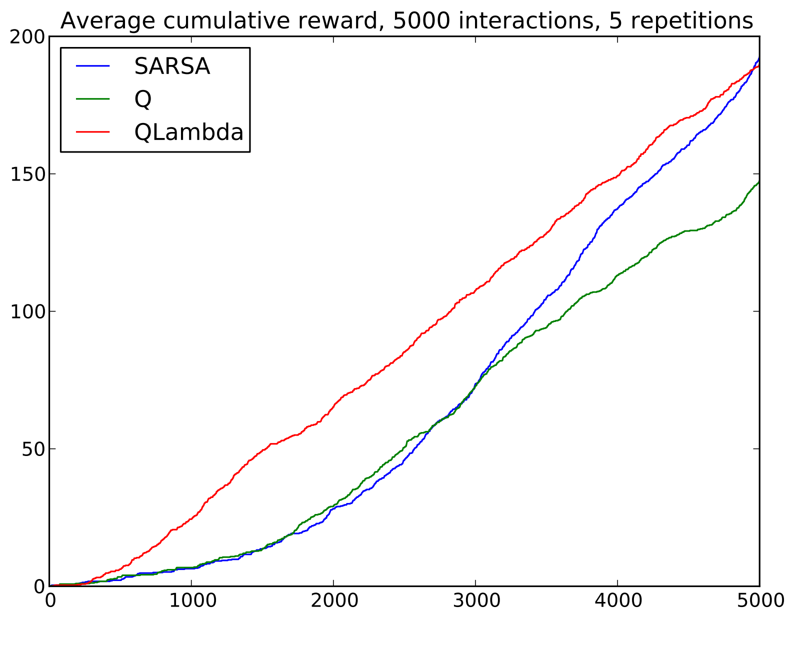
Average cumulative reward, 5000 interactions, 5 repetitions
SARSA
Q
QLambda
0
50
100
150
200
0
1000
2000
3000
4000
5000

Figure 8:

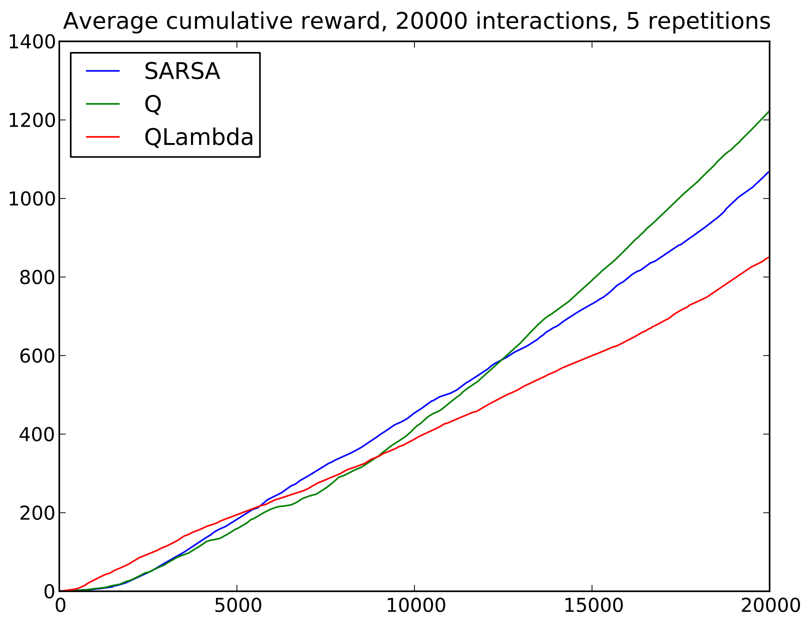
Average cumulative reward, 20000 interactions, 5 repetitions
SARSA
Q
QLambda
0
200
400
600
800
1000
1200
1400
0
5000
10000
15000
20000

Figure 9:

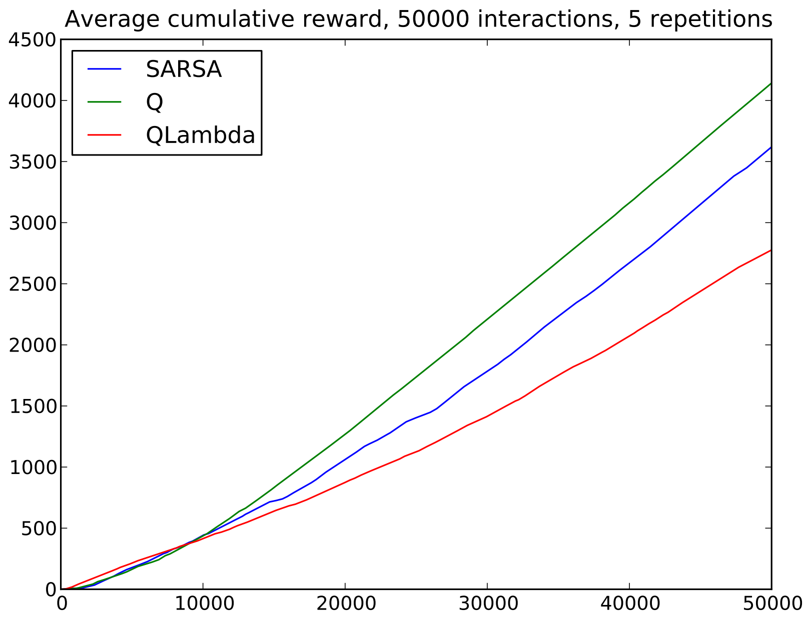
Average cumulative reward, 50000 interactions, 5 repetitions
SARSA
Q
QLambda
0
500
1000
1500
2000
2500
3000
3500
4000
4500
0
10000
20000
30000
40000
50000

Figure 10:

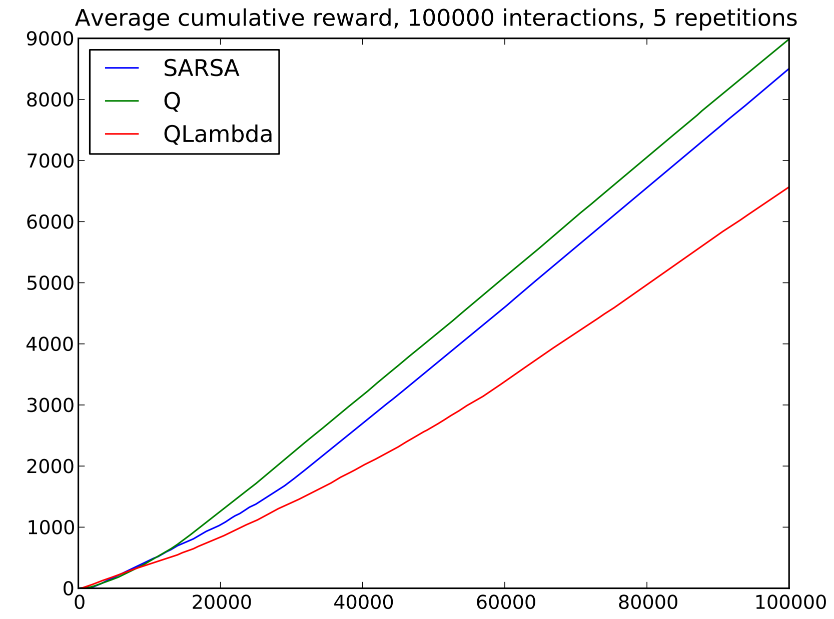
Average cumulative reward, 100000 interactions, 5 repetitions
SARSA
Q
QLambda
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
0
20000
40000
60000
80000
100000

Figure 11: