ROB 537 Homework 3: Reinforcement Learning

Nathan Butler | butlnath@oregonstate.edu

Overview

The objective of this assignment was to train an agent to explore a 5x10 gridworld with the objective of finding a door. The environment includes:

- a door initialized at coordinates (9,1) with a reward of 20;
- a solid wall (gray) the agent cannot move across;
- a reward of -1 for every time the agent is in a state other than the red door state.

The agent starts at a random location and has four actions (move in four directions). The state of the system is the location of the agent (x,y), and an episode is 20 time steps.

Unless otherwise specified, the models trained in this assignment were trained over 200 epochs with a step size of 0.85, a discount factor of 0.95, and an epsilon value of 0.9 per the training setup discussed in [1] and [2]. Throughout testing the epsilon value was modified to examine the results of emphasizing exploration vs exploitation.

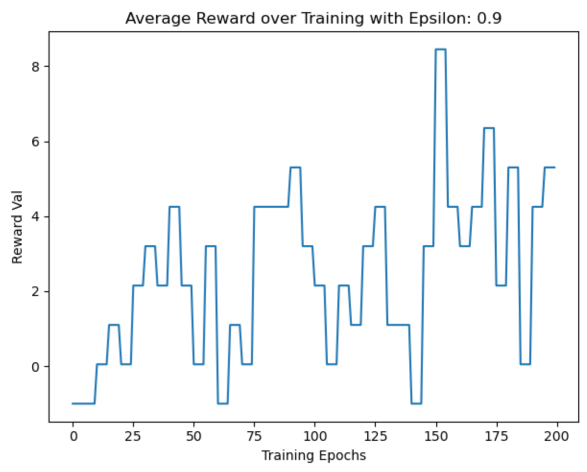
Academic Statement

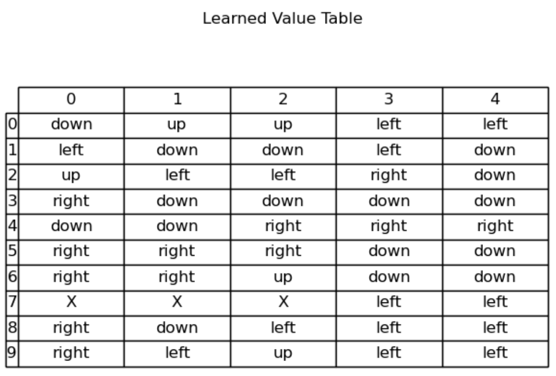
Outside of the provided code, all code written for this assignment is my own. The cited resources were used as reference materials for my own implementation. Additionally, I worked with Noah Boehme to determine a method to evaluate our models' learning curves.

SARSA Performance

Figure 1 displays the learning curves and value tables gained by varying SARSA training configurations. Learning success metrics were calculated every 5 epochs by conducting 20 tests with the agent initialized in random positions. The average final reward of these tests is plotted in Figure 1. During training, an epsilon-greedy approach was taken to encourage exploration [2]. However, during testing epsilon was disabled so that the algorithm always selected the action with the highest reward. The percentage of successful results gained during testing is also displayed in Figure 1.

The performance of the SARSA learning approach relied largely upon the balance of exploration and exploitation captured by the epsilon parameter. A larger epsilon encouraged the model to explore less optimal actions, but as a result it seemed unable to capture meaningful reward values. Lower epsilons produced behaviors in which the model prioritized higher-reward actions more frequently during training.

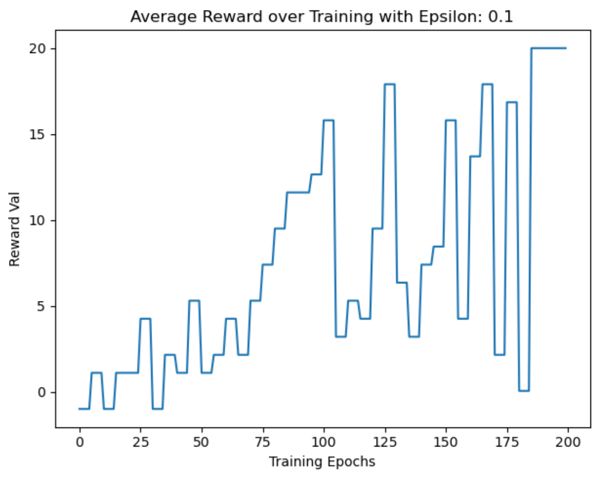


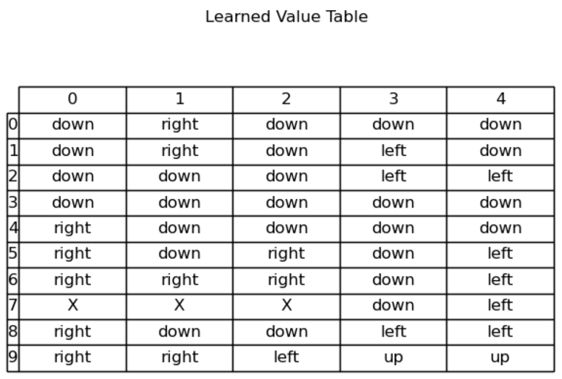

Learned Value Table

	0	1	2	3	4
0	down	up	up	left	left
1	left	down	down	left	down
2	up	left	left	right	down
3	right	down	down	down	down
4	down	down	right	right	right
5	right	right	right	down	down
6	right	right	up	down	down
7	X	X	X	left	left
8	right	down	left	left	left
9	right	left	up	left	left

Percent of Solutions Solved using Trained Q-Table: 19.5%

(a)




Learned Value Table

	0	1	2	3	4
0	down	right	down	down	down
1	down	right	down	left	down
2	down	down	down	left	left
3	down	down	down	down	down
4	right	down	down	down	down
5	right	down	right	down	left
6	right	right	right	down	left
7	X	X	X	down	left
8	right	down	down	left	left
9	right	right	left	up	up

Percent of Solutions Solved using Trained Q-Table: 100.0%

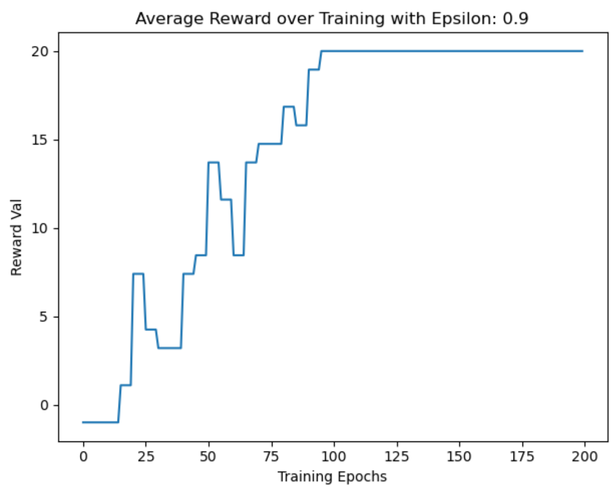
(b)

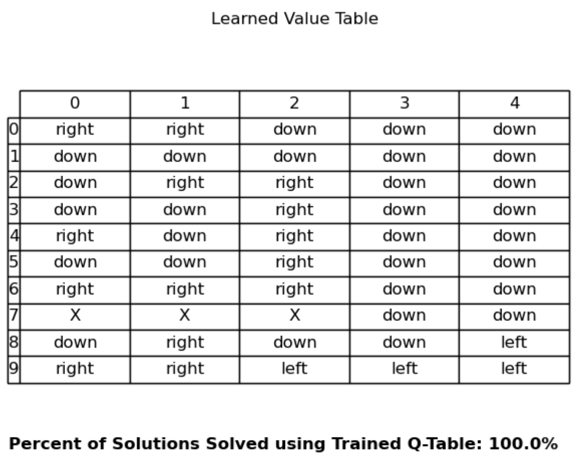
Figure 1. Learning curves and value tables for SARSA models with (a) representative example of epsilon = 0.9 and (b) representative example of epsilon = 0.1

As observed in Figure 1, prioritizing exploitation during training yielded a value table capable of producing correct solutions 100.0% of the time, as compared to the higher epsilon values that yielded correct solutions only around 20% of the time.

Q-Learning Performance

Figure 2 displays the learning curves and value tables gained by varying Q-learning training configurations, generated using the same process as SARSA. Q-learning saw much better performance during both training and testing, reaching a 100.0% success rate for models trained using a wide range of epsilon values.

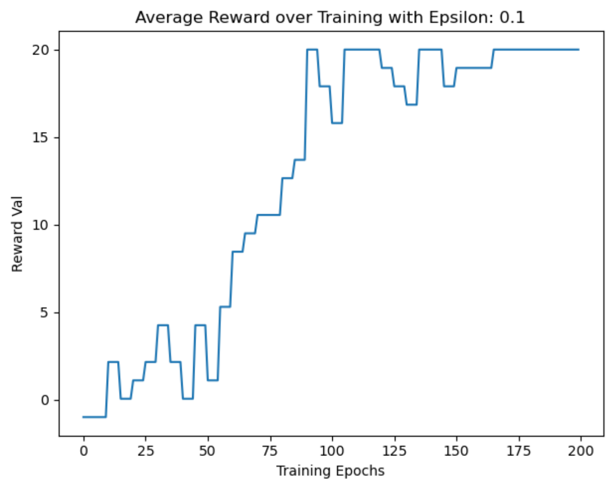


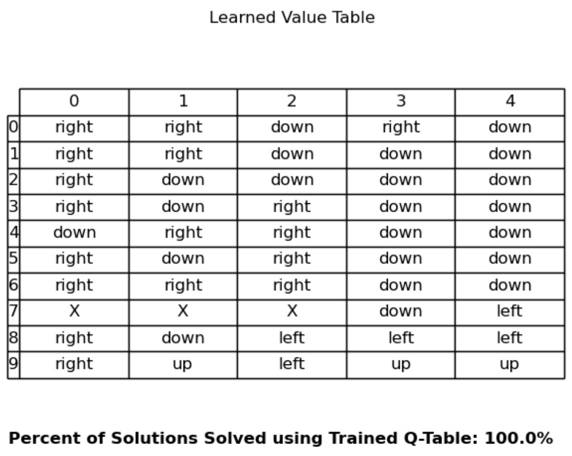

Learned Value Table

	0	1	2	3	4
0	right	right	down	down	down
1	down	down	down	down	down
2	down	right	right	down	down
3	down	down	right	down	down
4	right	down	right	down	down
5	down	down	right	down	down
6	right	right	right	down	down
7	X	X	X	down	down
8	down	right	down	down	left
9	right	right	left	left	left

Percent of Solutions Solved using Trained Q-Table: 100.0%

(a)




Learned Value Table

	0	1	2	3	4
0	right	right	down	right	down
1	right	right	down	down	down
2	right	down	down	down	down
3	right	down	right	down	down
4	down	right	right	down	down
5	right	down	right	down	down
6	right	right	right	down	down
7	X	X	X	down	left
8	right	down	left	left	left
9	right	up	left	up	up

Percent of Solutions Solved using Trained Q-Table: 100.0%

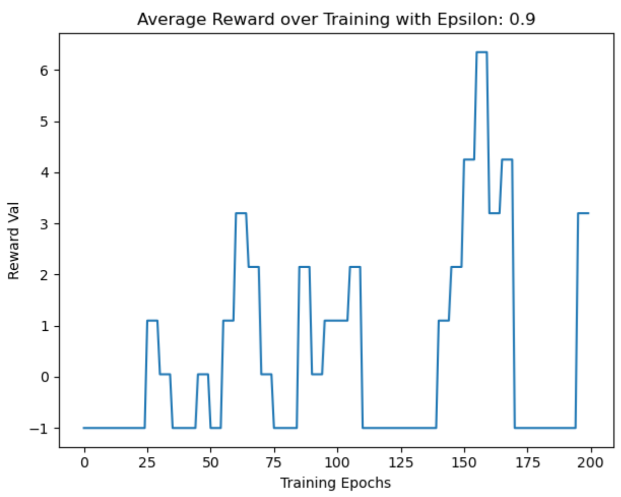
(b)

Figure 2. Learning curves and value tables for Q-learning models with (a) representative example of epsilon = 0.9 and (b) representative example of epsilon = 0.1

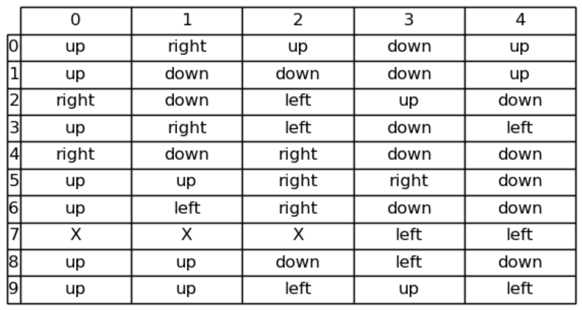
The results for Q-learning indicate much improved performance over the SARSA algorithm. This may be due to the off-policy nature of Q-learning, which permits the algorithm to learn the state-action values of policies other than the one that it is currently following [3]. This framework encourages the algorithm to prioritize policies that moves through the bottleneck on the gridworld, unlike SARSA.

Moving Door Evaluation

Using the same algorithms and parameters evaluated in the previous parts of this assignment, new models were trained in an environment in which the door moves randomly by 1 cell each time step (it is initialized in the same location). Figure 3 captures the learning curve, value table, and testing performance for trained SARSA models and Figure 4 contains the same information for Q-learning algorithms.



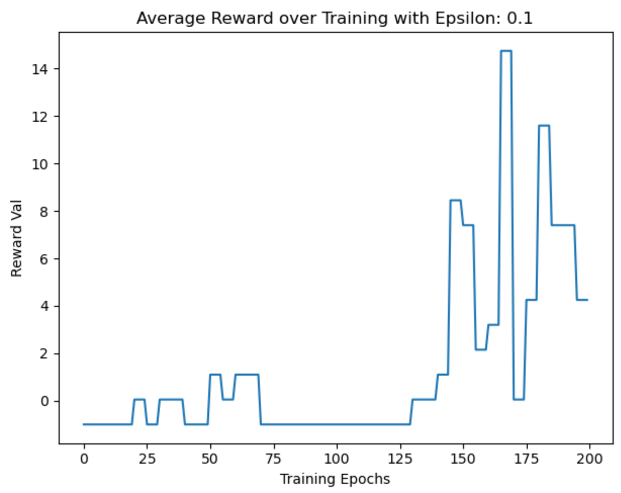
Learned Value Table



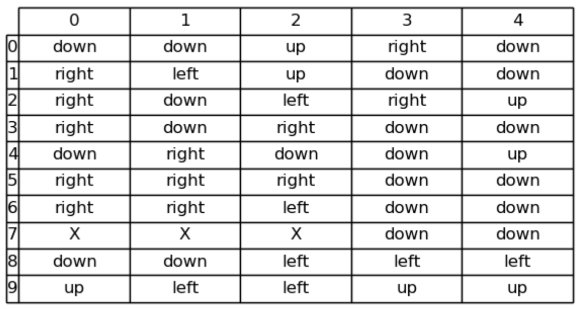
	0	1	2	3	4
0	up	right	up	down	up
1	up	down	down	down	up
2	right	down	left	up	down
3	up	right	left	down	left
4	right	down	right	down	down
5	up	up	right	right	down
6	up	left	right	down	down
7	X	X	X	left	left
8	up	up	down	left	down
9	up	up	left	up	left

Percent of Solutions Solved using Trained Q-Table: 26.5%

(a)



Learned Value Table

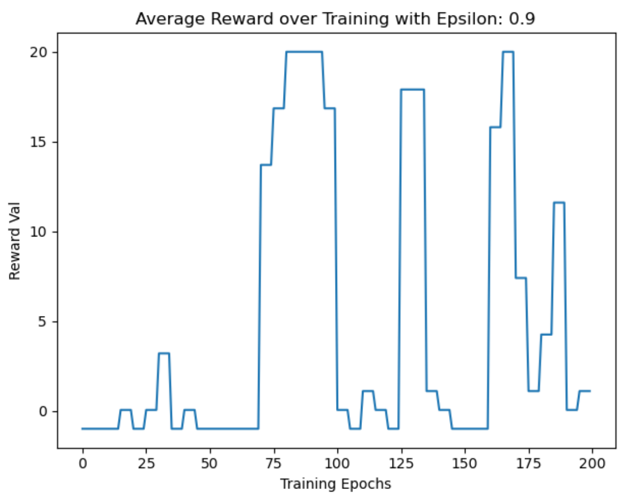


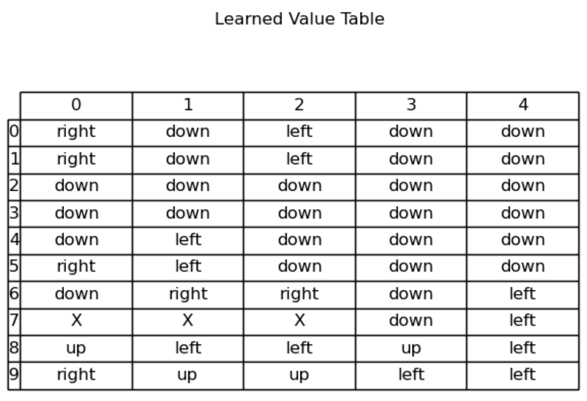
	0	1	2	3	4
0	down	down	up	right	down
1	right	left	up	down	down
2	right	down	left	right	up
3	right	down	right	down	down
4	down	right	down	down	up
5	right	right	right	down	down
6	right	right	left	down	down
7	X	X	X	down	down
8	down	down	left	left	left
9	up	left	left	up	up

Percent of Solutions Solved using Trained Q-Table: 46.0%

(b)

Figure 3. Learning curves and value tables for SARSA models with moving door using (a) representative example of epsilon = 0.9 and (b) representative example of epsilon = 0.1

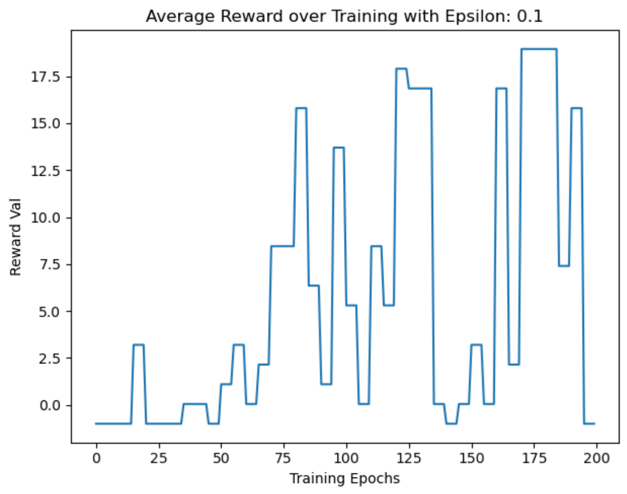


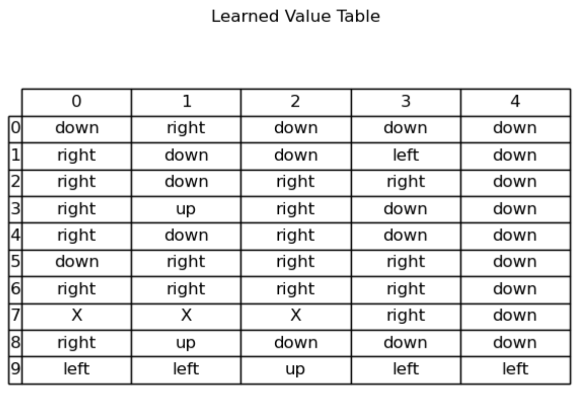

Learned Value Table

	0	1	2	3	4
0	right	down	left	down	down
1	right	down	left	down	down
2	down	down	down	down	down
3	down	down	down	down	down
4	down	left	down	down	down
5	right	left	down	down	down
6	down	right	right	down	left
7	X	X	X	down	left
8	up	left	left	up	left
9	right	up	up	left	left

Percent of Solutions Solved using Trained Q-Table: 25.0%

(a)




Learned Value Table

	0	1	2	3	4
0	down	right	down	down	down
1	right	down	down	left	down
2	right	down	right	right	down
3	right	up	right	down	down
4	right	down	right	down	down
5	down	right	right	right	down
6	right	right	right	right	down
7	X	X	X	right	down
8	right	up	down	down	down
9	left	left	up	left	left

Percent of Solutions Solved using Trained Q-Table: 53.5%

(b)

Figure 4. Learning curves and value tables for Q-learning models with moving door using (a) representative example of epsilon = 0.9 and (b) representative example of epsilon = 0.1

In this setting the agent struggles to learn a policy that is successful as those learned in the first parts of the assignment, often performing with a success rate below 50%. Inspecting the learned value tables, it appears that the learned policy largely moves the agent down and to the right. However, a mobile door may have moved far from its original position, rendering this general policy useless in many scenarios. There may also be situations where the agent reaches the door by taking a unique or unexpected move that prompts the learner to favor an action that may – for most cases – be suboptimal. In general, this example highlights a main drawback of these reinforcement learning algorithms – that they are not able to learn effective policies in dynamic environments. In this instance, the mobile door hinders the reproducibility of the benefits of the learned value functions.

It seems that better policies may be ones that place a high priority on exploration, though even these would rely largely on luck for finding the door. Perhaps a more effective model would be one that learns specialized policies according to the position of the door, drawing on some MAP-Elites concepts. This may enable the model to learn more specialized behaviors in response to environmental changes.

References

[1] AlindGupta, " SARSA Reinforcement Learning," GeeksforGeeks, 2023. [Online]. Available: https://www.geeksforgeeks.org/sarsa-reinforcement-learning/

[2] samishawl, " Epsilon-Greedy Algorithm in Reinforcement Learning," GeeksforGeeks, 2023. [Online]. Available: https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/

[3] A. Gautam, "SARSA Reinforcement Learning Algorithm: A Guide," Builtin, 2023. [Online]. Available: https://builtin.com/machine-learning/sarsa

Appendix

Link to github homework repository: https://github.com/natbut/rob537_hw