Grid World Implementation with Q-learning

Behzad Javaheri

Introduction: We have formulated a grid world using the Markov Decision Process (MDP), previously. Environments adhering to MPD consist of A) a state (S) which is a set of agent observable states [2]; B) action which is a set of possible actions A in state S [3]; C) state transition probability matrix that defines transition from state S to S' numerically; D) reward define as a numerical value (positive or negative) given to agent for taking action A; E) discount factor as a deprecation value between 0 and 1 to reduce the value of future rewards and F) policy [4].

There are two main types of reinforcement learning (RL) algorithms. A model-based that utilises transition and reward to estimate optimal policy and value function, whereas a model-free estimates the optimal policy without using or estimating transition and reward functions of the environment. One of the RL model-free algorithms is Q-learning, a value-based approach aiming to obtain the corresponding Q-value of a given action-state pair and subsequently updating the value function without modelling the environment explicitly [5].

An agent uses the Q-function to decide on the type of function. The Q-function is expressed as the Bellman equation (Fig. 1) describing a numerical relationship between value function of state S and vale function of state S'. It represents the value function of a given state as the cumulative discounted rewards obtained when an agent transitions to state S' affected by the agent's present policy. It equates performing action A at state A to the reward awarded in state S' plus a maximum of all future rewards at a discounted rate. The Bellman equation is used with the Bellman optimality equation to identify optimal policy (Fig. 2). The latter used to update values in the Q-table populated initially with zeros. This iterative process consists of a) initialising the Q-table; b) choosing an action; c) performing an action; d) measuring reward and e) evaluating and updating Q-table [4, 6, 7]. In this task, we will implement Q-learning using the described environment.

$$Q(s,a) = r(s') + \gamma \cdot \max\left(Q(s',a')\right)$$

Figure 1. Bellman equation [1]

$$Q(s,a) \rightarrow Q(s,a) + \alpha(r(s') + \gamma \cdot \max\left(s',a'\right) - Q(s,a))$$

Figure 2. Updating Bellman optimality [1]

Environment: The environment description is mostly similar to task 1. For this task, we aimed to introduce maximum stochasticity for the agent (mouse), end-goal terminal (cheese) and the trap (negative terminal). A bigger grid of 10x10 cells was defined. The location of the agent, end-goal and terminal trap was instructed to start at random states within the grid at the beginning of each episode. End-goal and trap were defined as terminal.

Agent, action and rewards: Similar to task 1, there is a single agent with four potential actions of up, down, left and right and for each step, apart from cheese and trap, a negative reward of -1 is awarded. Big rewards are assigned to terminal states of trap (-100) and cheese (+100).

Exploration vs exploitation: The type of action that an agent takes is determined by a balance between exploration & exploitation, achieved using epsilon greedy where epsilon is defined for the beginning and to decrease subsequently using epsilon decay. Therefore, early actions are more random and for later actions to be more directed by the Q-table. Epsilon 0.9, epsilon decay of 0.9999, a learning rate of 0.1, a discount factor of 0.95.

Implementation: The implementation was divided into the following steps: **1)** Setting parameters related to animation. This was to visualise the location of agent, trap and cheese for a selected display frequency. A colour dictionary was defined to assign colours to agent (blue), trap (red) and cheese (green). **2)** The environment and actions were defined. The x and y coordinates of agent, trap and cheese were set to random for every episode within the grid and the reward for actions (up, down, right, left) was defined to be -1 unless the transition is into either trap or cheese which as described will award rewards of +100 and -100 respectively. **3)** A function to define "move" was implemented to allow cheese and trap to move within an episode either individually or together. **4)** Q-table and training are initialised in a loop and updated (Figs. 1-2). **5)** reward-episode graph plotted.

Results: For all implementations, the initial location of agent, cheese and trap were randomly assigned. We aimed to compare the performance of Q-learning in our grid world two-fold: 1) to introduce additional stochasticity by allowing cheese and/or trap to move during each episode and 2) to modify learning rate, discount factor and epsilon to examine their effects on the performance of training. In total 12 trainings (2000 episodes each) were performed.

Initially, we compared the performance of Q-learning in static (cheese & trap were static), trap moving, cheese moving and cheese/trap moving simultaneously. Our data (Fig. 3) show that at the beginning of training the score was between -70 to -90 and gradually increased. The least performing group was when both cheese and trap were moving during training and the best performing group was when the cheese was moving. For example, in episode 2000, the score for the former was ~20, whereas for the latter ~50. This trend was evident almost during all episodes. The performance of the trap moving group vs static was similar, suggesting that having a dynamic trap did not significantly alter the reward our agent obtained (Fig. 3).

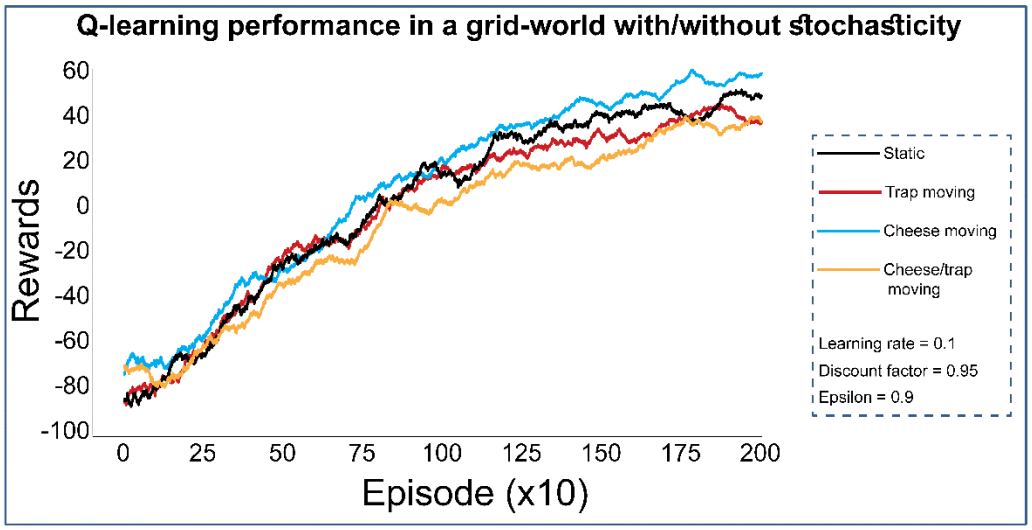

Figure 3. Q-learning performance in a grid with/without stochasticity. The location of agent, cheese and trap randomly assigned at the beginning of each episode. Training for 2000 episodes for each line; black: cheese and trap were static, red line: trap moving; turquoise line: cheese moving, and orange line: both trap and cheese moving. Learning rate 0.1, discount factor at 0.95 with epsilon at 0.9.

We next examined the effect of 10x lowering learning rate and discount factor (to 0.01 and 0.095), respectively. Our data (Fig. 4) indicate that when the cheese and trap were static the score was the lowest amongst the groups starting at around -100 and finishing around similar values. Providing additional stochasticity by allowing the cheese and trap to move either individually or in combination enhanced the rewards, although these were lower than those

observed in Figure 1. The performance of trap moving, and combined trap/cheese moving was similar. For example, both groups scored ~-30 at episode 2000. The cheese moving group resulted in higher rewards from around episode 900 to the end episode. The rewards obtained at episode 2000 for the cheese moving group was ~0, higher than other groups (Fig. 4).

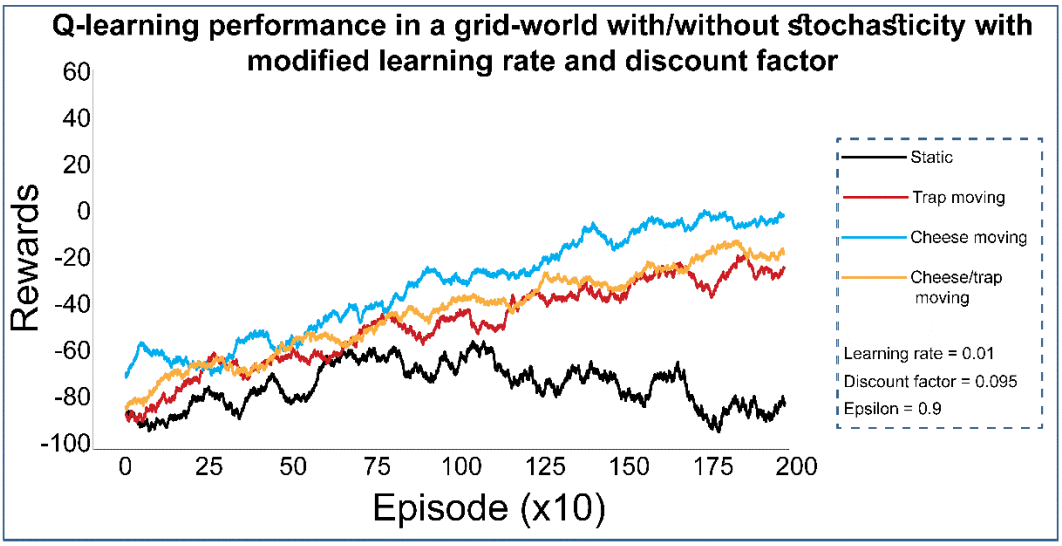

Figure 4. Q-learning performance in a grid with/without stochasticity & modification to learning rate and discount factor. The location of agent, cheese and trap were randomly assigned at the beginning of each episode. The training run for 2000 episodes; black: cheese and trap static, red line: trap was moving; turquoise line: the cheese was moving, and orange line: both trap and cheese were moving. Learning rate set at 0.01, discount factor at 0.095 with epsilon at 0.9.

Does decreasing epsilon affects performance? We examine this question by 10x decrease in epsilon and find some improvement in the performance of the static group compared to Figure 1 was evident. For example, at episode ~350 the reward for the static group was ~-20 whereas in Figure 1 with 10x higher epsilon this was ~-50. The performance of this static group was higher than others with the least performing group was when both cheese and trap moving during the training. The cheese moving group performed similarly to the static group apart from the end episodes where it obtained similar rewards to the trap moving group.

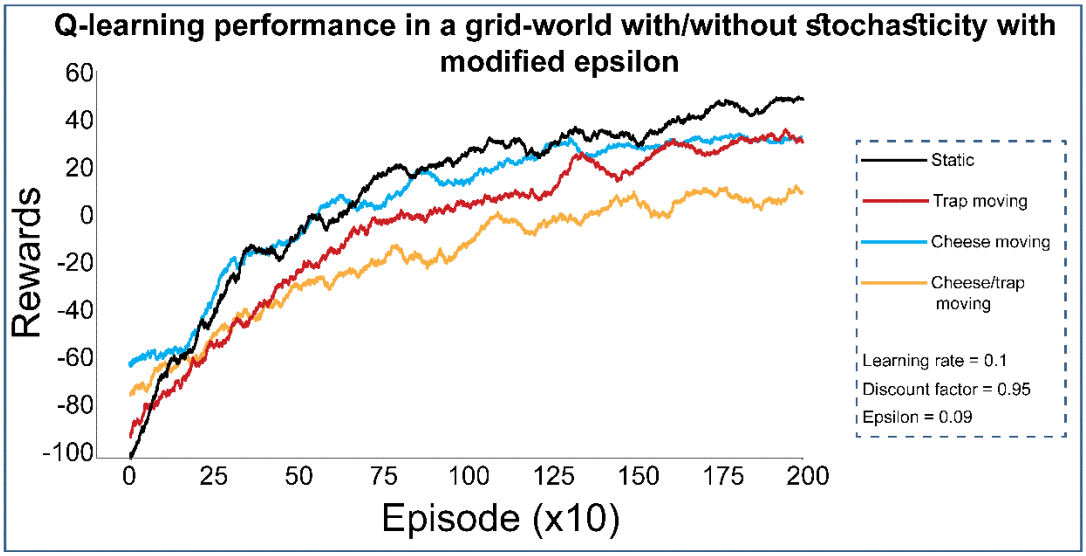

Figure 5. Q-learning performance in a grid with/without stochasticity and modification to epsilon. The location of agent, cheese and trap randomly assigned. Training for 2000 episodes; black:

the cheese and trap were static, red line: trap was moving; turquoise line: cheese moving; and orange line: both trap and cheese moving. Learning rate set at 0.1, discount factor at 0.95 with epsilon at 0.09.

Discussion: Herein, we have implemented Q-learning and evaluated the performance in our grid world. We have introduced stochasticity by assigning random states to the agent, and two terminal states (cheese & trap) as well as additional stochasticity by allowing the trap and cheese to move randomly either individually or together. We also examined the effect of changing training parameters.

Our data show increasing reward as the number of episodes increases suggesting a fine balance between exploration and exploitation wherein the beginning more random actions are taken to explore and later actions are directed by Q-table leading to higher rewards. Furthermore, our data suggest that additional stochasticity for the cheese improved the performance. Although no statistical tests have been performed, a trend is evident. Allowing the cheese and trap to move simultaneously resulted in a lower score.

The performance is negatively impacted when we reduce learning rate and discount factor resulting in significant reduction in reward (Fig. 4). The learning rate regulates how fast estimates are modified and discount factor modifies the future rewards [8]. This may suggest that Q-values are not updated due to lower learning rate and therefore no optimal learning is achieved. This negative impact is somewhat compensated when additional stochasticity by moving cheese, trap and combination are introduced, particularly at the later stages of training suggesting that additional iterations might allow to further increase the cumulative rewards [8].

In addition, reducing epsilon by 10-fold did not alter the rewards obtained for the static group, in fact, resulted in improvement particularly at earlier episodes. It, however, reduced rewards in the groups where one element or combined elements were moving. Epsilon controls exploitation vs exploration by regulating randomness [9] and therefore our data suggest that increased exploitation improves rewards only when the end-goal (cheese) and negative terminal state (trap) are static. A grid search to identify optimal hyperparameters would provide additional information on the effect of hyperparameters on the performance of Q-learning.

References

[1] S. K. Basaveswara. "Q-Learning: A Maneuver of Mazes." https://becominghuman.ai/q-learning-a-maneuver-of-mazes-885137e957e4 (accessed 09/03/2021.

[2] A. R. Cassandra, "Exact and approximate algorithms for partially observable Markov decision processes," 1998.

[3] M. L. Littman, "Value-function reinforcement learning in Markov games," *Cognitive systems research,* vol. 2, no. 1, pp. 55-66, 2001.

[4] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *NIPs*, 1999, vol. 99: Citeseer, pp. 1057-1063.

[5] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning,* vol. 8, no. 3-4, pp. 279-292, 1992.

[6] J. Christopher, "Learning from delayed rewards," *PhD thesis, Cambridge University,* 1989.

[7] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *Journal of artificial intelligence research,* vol. 13, pp. 227-303, 2000.

[8] E. Even-Dar, Y. Mansour, and P. Bartlett, "Learning Rates for Q-learning," *Journal of machine learning Research,* vol. 5, no. 1, 2003.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.