USING Q-LEARNING AND MODULES TO TRAVERSE A TWO-DIMENSIONAL GRID

Joel Walsh
Department of STEM Education
University of Texas at Austin
Austin, TX, USA

ABSTRACT

Q- learning is a particular subset of Reinforcement Learning, well adapted to situations with finite Markov Decision Processes. For this anaylsis, Q-learning was used to train an agent to walk across a virtual environment simulating a sidewalk.

1 INTRODUCTION

Q-learning belongs to the broader class of algorithms called Reinforcement Learning (RL.) RL is primarily concerned with "how to map situations to actions as to maximize a numerical reward signal." (Sutton, 2018) RL is somewhat of a flashy field these days for a lot of reasons. Some find that the trial and error and optimization in the presence of rewards mimics how animals learn. RL advances by companies like DeepMind have allowed researchers to crack some of the holy grails of Artifical Intelligencr Research, beating a top-ranked human in the game of Go. While the training process might appear as trial and error to some, what is actually happening under the hood is quite miraculous- high dimensional conditional probability distributions are being fleshed out and used for inference.

2 METHODS

The keys to RL are an attention to the tradeoff between ***exploration*** and ***exploitation***, and the relationship between states and actions. RL is very well suited for learning settings where states and actions to reach those particular states have clear rewards. Exploitation involves selecting the highest benefit action at each state to maximize reward. While this certainy seems reasonable, it is actually quite myopic. If an agent only seeks to exploit the highest reward by their actions, they will not learn as much about the state space as they would have under another protocol. One such protocol is called the ***epsilon- greedy*** approach. The action with the highest reward is taken most of the time, while occasionally a sub-optimal action is taken to increase the fleshing out of the state space. The end goal is to acheive optimal ***policy***, or the ideal mapping from states to probabilities.

For finite MPDs, Q-learning is tried and true method of learning. Q- learning is but one of what are referred to as Temporal Difference (TD) learning methods. A key tenet is that TD methods allow the user to estimate parts of a value function using indirect methods, such as observing states close to it, often without reaching the end of an episode.

One assumption of Q-learning is that the agent has a bead on the expected reward for events one time period in the future. In practice this can be inferred or randomly initialized, as the associated values that come from a future action will eventually become clearer. The mechanism by which this functions is the reward matrix R(s,a) . This multi-dimension array is a function of states and actions. Inexplicably, I chose to represent states in two dimensions (row x columns), so my reward matrix was actually three dimensional, with the third dimension representing each possible action from each state. Actions were restricted to the five forward-moving directions, and some movements were permitted depending on where the agent was currently at on the board, so as not to fall offthe board.

As the assignment called for a module based approach, I sought first to create several two dimensional boards. One had nothing but grass on the top and bottom, one had only litter, and one had

only obstacles. All had an ending reward strip, to motivate the agent to go from one side to the other. From these boards I constructed rewards matrices. If a particular square contained litter, with a reward of +1, I would affix a +1 reward to all of the surrounding cells that represented places that the reward could be reached from.

Once all of the reward matrices were constructed, I was able to begin the Q-learning algorithm:

$$
\begin{array}{l}
\text{Algorithm parameters: step size } \alpha \in (0,1], \text{ small } \varepsilon > 0 \\
\text{Initialize } Q(s,a), \text{ for all } s \in \mathcal{S}^+, a \in \mathcal{A}(s), \text{ arbitrarily except that } Q(\text{terminal},\cdot) = 0 \\
\text{Loop for each episode:} \\
\text{Initialize } S \\
\text{Loop for each step of episode:} \\
\text{Choose } A \text{ from } S \text{ using policy derived from } Q \text{ (e.g., } \varepsilon \text{ -greedy)} \\
\text{Take action } A \text{ , observe } R, S' \\
Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right] \\
S \leftarrow S' \\
\text{until } S \text{ is terminal}
\end{array}
\tag{1}
$$

As you can see, once you have fleshed out the reward matrix it is possible to populate the Q matrix, which is the same size due to also being a function of states and actions. I also made this as a row x columns x directions three dimensional matrix. This process was repeated for each module; each module had its own rewards and Q-matrices. After 1000 episodes of training, I tested some of the individual modules in a variety of board configurations. (The grass and forward modules were trained on the same board, but each episode was initialized at a random spot on the board.) Once suitable q matrices were formed, I wrote a function to take the maximum Q value for each action at each state. This left me with a row x column grid. I then used another function to take these directions and trace a path filling in each cell of the optimal path with a number (0.7), and then converted this grid into a heatmap.

Once all of the individual cells were done training, I combined and averaged the individual normalized Q matrices to form a new Q matrix. I then traced a path over the combined sidewalk. Just for comparison's sake, I trained the agent using one Q matrix and a full sidewalk map as well for comparison.

3 RESULTS

For a choice of rewards I chose an end goal of +5, +1 for litter, and -1 for both grass and obstacles. In the individual training modules, it appears as if the pull of the end goal was very strong. (See Figure 1) All agents eventually straightened out and made a bee line for the end goal. If I had more time I might try to vary rewards, making the tradeoffs between picking litter and avoiding obtacles more disparate. I might also try varying learning rate and discount. That being said, when I recombined Q matrices from each module, the outcome was very similar to the optimal policy when the Q matrix was trained on the entire sidewalk map. (See Figure 2) The reward was the same (+6), and the approach only vaired slightly. (Both plots are located at the end of the iPython notebook.) This would suggest that the module-based approach was actually quite good at finding an optimal policy.

4 SUMMARY

The module-based approach to Q-learning was reasonably succesful in this approach; and certainly a consideration for situations that can be modeled by finite MDP where the computational cost of training the entire environment at once is prohibitive. It did however, take much longer to develop. In situations where the state space is finite but very large, I might consider this approach again.

5 REFERENCES

Richard S. Sutton and Andrew G. Barto. 2018. Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA.

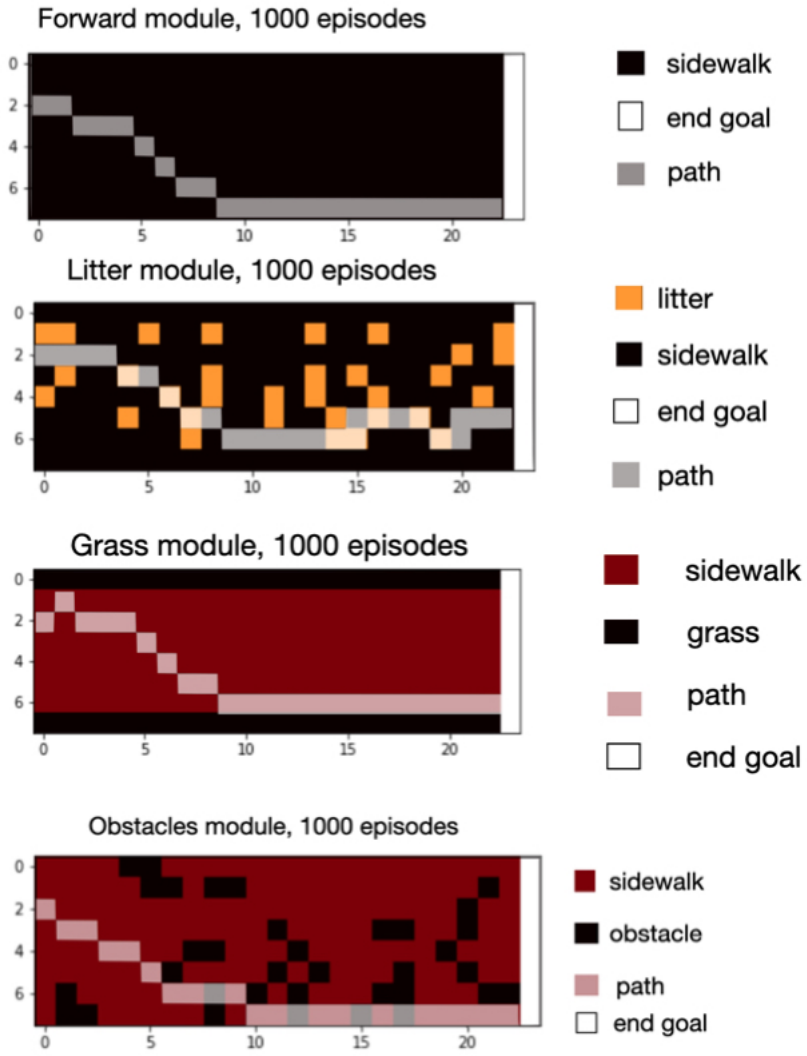

Figure 1: Individual model performance

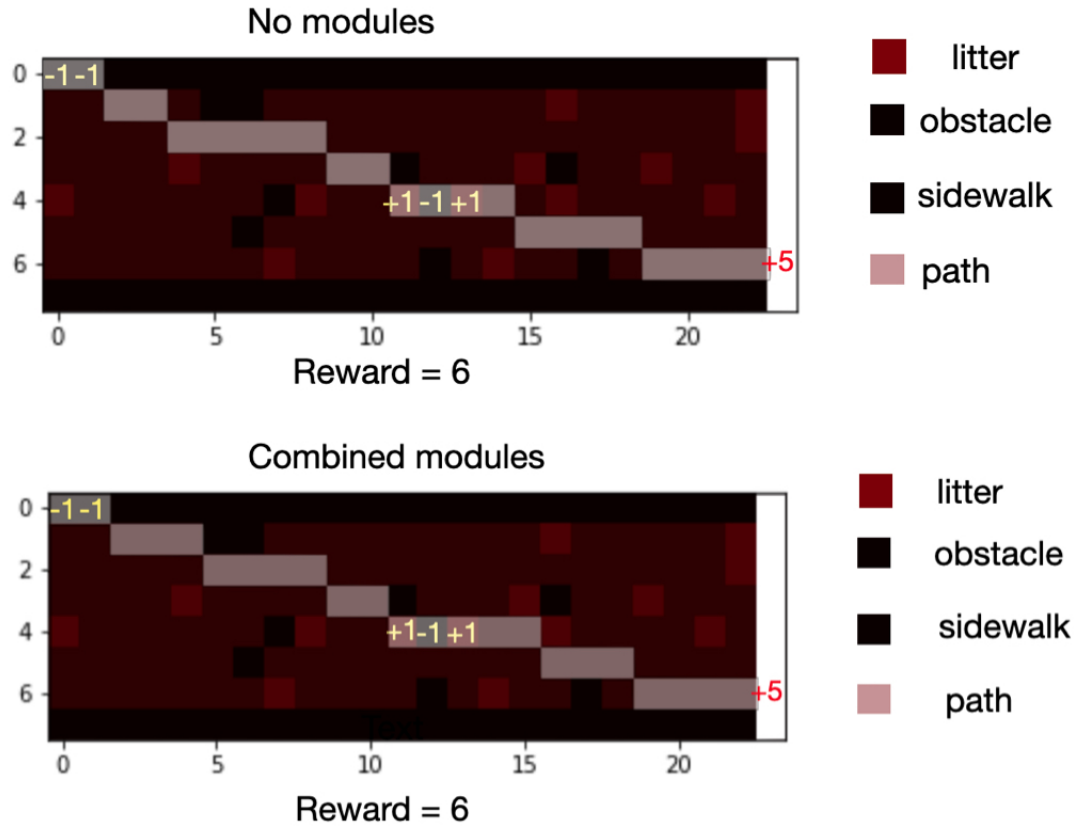

Figure 2: No modules versus combined module Q-matrix