CSE 410 Project 1: Building Reinforcement Learning Environment

Peter M. VanNostrand
Department of Computer Science
University at Buffalo
Buffalo, NY 14261
pmvannos@buffalo.edu

Abstract

In this project we define a simple grid world environment consisting of 4x4 grid, for a total of 16 states. In the first section we define this environment to be deterministic with the four possible actions: up, down, left, and right. Then we modify this environment to be stochastic, with the above actions occasionally "failing" and transitioning to a different state than expected. Finally, we implement a Q-Learning agent which is trained in both environments to find an optimal path between a starting and ending location.

1 Environment

The reinforcement learning environment consists of four main parts: an observation space, an action space, a reward function, and a termination condition. The observation space is the set of all possible observable state in a given environment, these comprise all possible states an agent could exist in. The action space is the set of all allowed action that an agent can take to perform a task within an environment, in our case the agent will be limited to movement actions but other environment could include other options such as inspection actions, action that modify the environment. The reward function is a function that uses the current state of the environment – the state of the agent, other entities, the duration of execution, etc – to generate a quantitative reward for the agent. The agent can then use this reward as feedback to learn to perform a specified task. Lastly a termination condition is an optional component in some environment which terminates the execution of an environment. Not all environments contain such a component, particularly those involving online learning, but the environments in this project are deterministic and therefore must terminate. Below we will outline how these four components for two separate environments designed as part of this project.

1.1 Observation Space

The environments of this project are both simple grid worlds consisting of a 4x4 grid of cells, for a total of 16 states. States are identified by their row, column coordinates with columns and row numbered 0-3 from top to bottom and left to right as shown in the following figure.

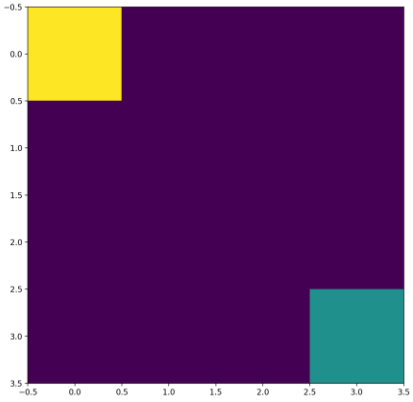

Figure 1: 4x4 Grid World

1.2 Action Space

In these environments only four possible actions are possible up $a_0 = up$, $a_1 = right$, $a_2 = down$, and $a_3 = left$. When one of these actions is executed the agent transitions from on state to another. In the deterministic environment a_0 moves the agent one row up the grid, a_1 moves the agent one column to the right of the grid, a_2 moves the agent one row down, and a_3 moves the agent one row left. All four actions are bounded by the edges of the grid such that if the agent attempts to move off of the grid it remains in place by transitioning back to the same state.

1.3 Reward Function

In both environments the reward function is established in the same way. Both environments are designed to contain one agent, beginning from a defined starting point, as well as a specified goal location. At each time step the current Euclidean distance between the agent and the goal location is computed, let's refer to this a d, then the agent performs an action and transitions to a new step. At this point the agent-goal distance d' is recomputed and a reward is assigned based on the following function

$$R(s,s') \begin{cases} -1, & if\ d' > d \\ 0, & if\ d' = d \\ +1, & if\ d' < d \end{cases}$$

This function rewards the agent positively as it approaches the goal, and negatively as it moves away from the goal. The reward of zero is used for cases when the agent remains a constant distance from the goal.

1.4 Termination Conditions

The environments in this project are programmed to stop under two conditions. Firstly, if the agent reaches the goal the environment terminates, this is determined by checking if the distance computed during the reward process is zero. Secondly, the environment will terminate if the agent has taken significantly more than an optimal number of time steps to reach the goal, for our 4x4 grid environment this is a total of nine time steps. This can result is some runs where the environment terminates before reaching the goal, particularly in the stochastic environment, but prevents the agent from wandering for long periods of time during early training episodes.

1.5 Deterministic Environment

In the deterministic environment the four possible actions (up, right, down, left) result in a reliable and specified state transition as defied in Section 1.2. This allows an agent to consistently transition between states, and through training ultimately learn an optimal path. Once learned the same path can be continually followed without deviation as the system is entirely deterministic, such an optimal path in this environment guarantees that the agent can reach the goal within the maximum allowed number of time steps. The complete state-action transition table is shown below.

Table 1: State-Action Transitions

State			Next State	
Row	Col	Action	Next Row	Next Col
0	0	UP	0	0
0	0	RIGHT	0	1
0	0	DOWN	1	0
0	0	LEFT	0	0
0	1	UP	0	0
0	1	RIGHT	0	2
0	1	DOWN	1	1
0	1	LEFT	0	0
0	2	UP	0	2
0	2	RIGHT	0	3
0	2	DOWN	1	2
0	2	LEFT	0	1
0	3	UP	0	3
0	3	RIGHT	0	3
0	3	DOWN	1	3
0	3	LEFT	0	2
1	0	UP	0	0
1	0	RIGHT	1	1
1	0	DOWN	2	0
1	0	LEFT	1	0
1	1	UP	0	1
1	1	RIGHT	1	2
1	1	DOWN	2	1
1	1	LEFT	1	0
1	2	UP	0	2
1	2	RIGHT	1	3
1	2	DOWN	2	2
1	2	LEFT	1	1
1	3	UP	0	3
1	3	RIGHT	1	3
1	3	DOWN	2	3
1	3	LEFT	1	2
2	0	UP	1	0
2	0	RIGHT	2	1
2	0	DOWN	3	0
2	0	LEFT	2	0
2	1	UP	1	1
2	1	RIGHT	2	2
2	1	DOWN	3	1
2	1	LEFT	2	0
2	2	UP	1	2
2	2	RIGHT	2	3
2	2	DOWN	3	2
2	2	LEFT	2	1
2	3	UP	1	3
2	3	RIGHT	2	3
2	3	DOWN	3	3
2	3	LEFT	2	2
3	0	UP	2	0
3	0	RIGHT	3	1
3	0	DOWN	3	0
3	0	LEFT	3	0
3	1	UP	2	1
3	1	RIGHT	3	2
3	1	DOWN	3	1
3	1	LEFT	3	0
3	2	UP	2	2
3	2	RIGHT	3	3
3	2	DOWN	3	2
3	2	LEFT	3	1
3	3	UP	2	3
3	3	RIGHT	3	3
3	3	DOWN	3	3
3	3	LEFT	3	2

1.6 Stochastic Environment

In the stochastic environment the four possible actions (up, right, down, left) do not always result in the same state transitions as expected from the deterministic environment. In this case there is a probability that the agent with transition to each of the four states above, below, left, and right of the current state.

This is accomplished by creating a small probability that the agent's action will "fail," essentially when the agent requests an action to perform, e.g. UP, there's a chance that one of the other actions will be performed, e.g. RIGHT, DOWN, or LEFT. These probabilities are randomly assigned for each state when the environment is instantiated, but are bounded by some rules. Firstly, the desired action is always at least 80% likely to occur, for example if the action UP is requested then there is at least an 80% chance that the UP action will occur. Secondly, the sum of the probability of all possible actions at any given state is always 1, this ensures that there is always a valid action.

Once the final action to be performed is selected, whether it is the requested action or a probabilistically selected one, the state-action transition occurs as in the deterministic environment. The probabilities for an action to occur given the current state-action pair are stored in a large table. As the values are randomly generated on each run, the exact contents cannot be included here, but the values from one run are reproduced below. For the values used in during your execution please see the transition_probabilities.txt file generated during execution.

Table 2: Action Probabilities in Percent

row	col	act	U	R	D	L
0	0	U	87.65	12.08	0.21	0.05
0	0	R	0.41	86.59	8.24	4.76
0	0	D	6.91	0.32	92.6	0.17
0	0	L	5.56	0.88	0.09	93.48
0	1	U	88.26	8.15	3.04	0.55
0	1	R	4.76	87.69	7.22	0.33
0	1	D	6.36	2.32	91.02	0.3
0	1	L	4.09	3.18	0.61	92.12
0	2	U	85.68	4.85	8.32	1.14
0	2	R	13.96	82.62	2.44	0.98
0	2	D	9.76	0.26	89.49	0.49
0	2	L	4.96	0.51	1.45	93.08
0	3	U	99.84	0.03	0.07	0.06
0	3	R	12.65	85.59	1.74	0.01
0	3	D	3.19	9.3	86.6	0.92
0	3	L	2.48	5.47	0.02	92.03
1	0	U	86.21	3.2	4.76	5.84
1	0	R	2.26	95.53	0.53	1.68
1	0	D	5.65	0.18	93.84	0.32
1	0	L	3.87	3.9	0.14	92.09
1	1	U	81.31	8.78	9.36	0.55
1	1	R	9.34	88.83	1.61	0.21
1	1	D	0.27	0.15	99.57	0.01
1	1	L	7.16	1.94	0.35	90.54
1	2	U	89.04	6.54	2.26	2.16
1	2	R	0.34	97.62	1.75	0.3
1	2	D	9.73	1.35	88.34	0.58
1	2	L	1.83	1.01	0.1	97.06
1	3	U	85.98	10.02	1.87	2.12
1	3	R	0.99	95.26	1.24	2.51
1	3	D	18.39	0.22	81.34	0.05
1	3	L	8.94	0.08	0.05	90.93
2	0	U	91.24	4.7	0.14	3.93
2	0	R	11.68	81.82	3.86	2.63
2	0	D	14.99	0.75	83.91	0.35
2	0	L	0.56	0.01	0.01	99.42
2	1	U	84.67	2.38	8.04	4.92
2	1	R	1.24	94.66	1.06	3.04
2	1	D	7.74	0.5	89.02	2.74
2	1	L	9.34	5.42	2.12	83.12
2	2	U	95.95	0.98	1.83	1.23
2	2	R	1.5	90.41	7.39	0.7
2	2	D	1.09	0.11	98.5	0.29
2	2	L	3.11	0.56	0.03	96.3
2	3	U	93.82	2.07	3.11	1
2	3	R	0.53	89.11	3.24	7.12
2	3	D	6.49	1.47	90.99	1.05
2	3	L	0.84	6.44	1.03	91.7
3	0	U	91.42	4.41	3	1.17
3	0	R	0.05	99.66	0.16	0.13
3	0	D	0.47	0.08	89.29	10.16
3	0	L	2.95	0.99	0.25	95.8
3	1	U	86.97	3.59	9.06	0.38

3	1	R	2.64	83.55	12.25	1.56
3	1	D	4.74	0.28	94.78	0.2
3	1	L	4.18	0.01	0.34	95.48
3	2	U	90.56	6.14	2.38	0.92
3	2	R	6.86	92.38	0.76	0
3	2	D	3.68	1.97	91.05	3.3

3	2	L	18.06	0.07	0.05	81.81
3	3	U	87.53	8.95	0.99	2.53
3	3	R	5.58	89.69	4.36	0.37
3	3	D	8.71	5.32	83.25	2.71
3	3	L	6.54	1.1	1.38	90.97

Note that because the state transitions are non-deterministic, the final learned path of the agent will vary depending on the randomness of the episodes it learns from. Despite this the agent will ultimately converge to an optimal path. While an optimal path can be learned, as the state transitions are stochastic is it possible that some episodes will not result in the agent reaching its goal. This occurs when a sufficient number of non-requested actions occur that consume time and push the agent off its optimal path. If this happens, rerunning the program will result in a new episode being generated which will likely reach the goal within the maximally allowed number of timesteps.

2 Learning Agent

To make use of these environments we designed a simple Q-Learning agent. This agent is designed to learn an optimal path between its starting position and a pre-defined goal position based upon the rewards it receives.

2.1 Q-Learning

To learn an optimal path to the goal the agent attempts to learn an estimate of the discounted cumulative future reward for every state-action combination. These estimated values are called Q-Values and stored in a so-called Q-Table which is updated at each step during training. The relation used to estimate the Q-Value for a given state-action pair is given below

$$Q'(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \max_a Q(s_{t+1}, a)\right)$$

Where s_t is the state at time t, a_t is the action taken at time t, α is the learning rate, and γ is the discount factor. For these experiments $\alpha = 0.1$ and $\gamma = 0.8$. This equation is used as the update function of our agent and updates the estimated Q-Value for the given state-action pair to incorporate the new information it obtains from the current episode. The learning rate α controls how significant the new samples information is in relation to the previously learned value. The discount factor γ adjusts the balance between immediate and future rewards. These variables allow the agent to learn a good approximation of the Q-Value and then make appropriate actions to maximize the total cumulative reward.

To ensure that every state-action pair is explored and learned the Q-Learning agent uses an epsilon-greedy strategy. This strategy is named after the value ϵ used during execution. Essentially in this strategy a random number between 0 and 1 is selected at each step, if the number is less than a predefined ϵ value a random action is taken, otherwise the action is taken that has the largest associated Q-Value. This allows the designer to balance exploration and exploitation and prevents the agent from continually repeating the first path it finds and not discovering a more optimal path. In this project with start with $\epsilon = 1.0$ so that the agent explores widely during the beginning of the training. As training continues ϵ decays exponentially based upon the function $\epsilon = \epsilon_0 e^{-0.005t}$, where t is the episode number. This is designed so that the value of epsilon gradually decreases, transitioning the agent from exploration to exploitation as training progresses. The decay rate of -0.005 was selected such that the agent performs almost entirely exploitation at the end of a 1,000 episode training period.

2.2 Q-Agent in Deterministic Environment

Using the deterministic environment described above we trained our Q-Learning agent for 1,000 episodes, and observed the results. During training we tracked the value of ϵ at each episode, a plot of this is shown below.

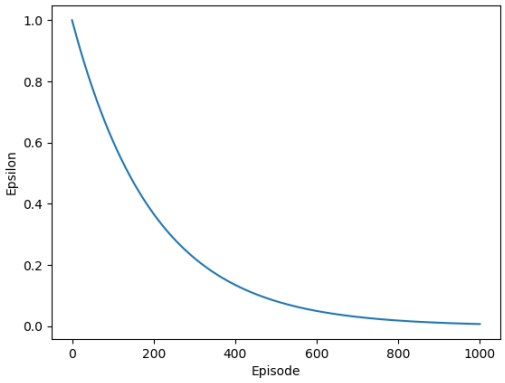

Figure 2: Epsilon per Episode for Deterministic Environment

Here we can see the desired epsilon curve, beginning with a large value to encourage exploration at the start of training and decaying to a small value as training completes for the optimal exploitation once trained.

We can observe the learned action of the agent in the following series of environment renderings.

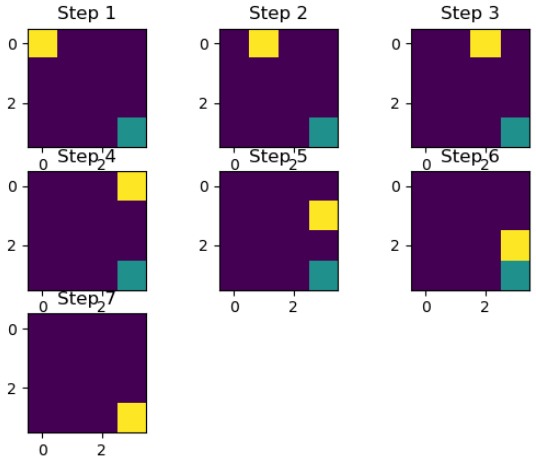

Figure 3: Q-Agent Movements in Deterministic Environment

Here we observe that the agent has learned to move directly the right of the environment and then down the edge of the grid towards the goal. We note that this series of actions takes seven steps, which is the optimal number of actions for our environment.

To examine the remaining learned actions, we can determine the action with the maximal associated Q-Value for each state and print the results as shown below

Table 3: Deterministic Learned Actions

	0	1	2	3
0	R	R	R	D
1	R	R	R	D
2	R	D	D	D
3	R	R	R	U

From this table we can see that for every possible action path the agent will reach the goal in an optimal number of steps by moving only right or down at any given time and never left or up.

2.3 Q-Agent in Stochastic Environment

We then repeated the training of our agent in the stochastic environment described above. And achieved the following results

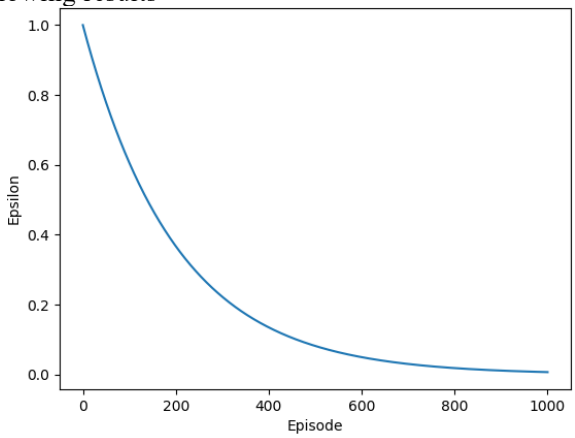

Figure 4: Epsilon per Episode for Stochastic Environment

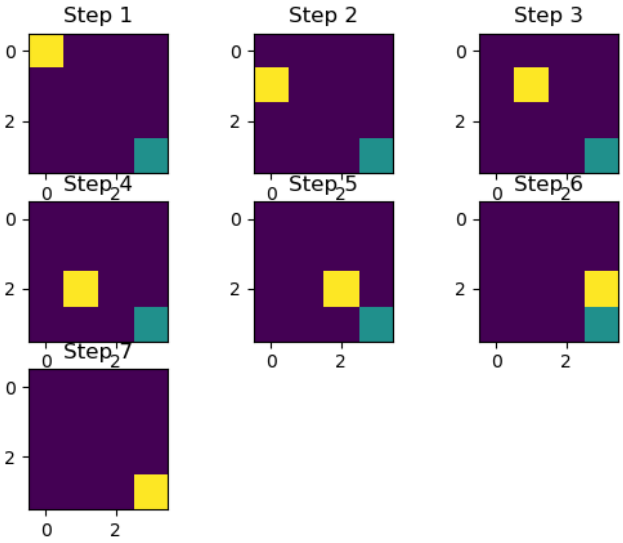

Figure 5: Q-Agent Movements in Stochastic Environment

Table 4: Deterministic Learned Actions

	0	1	2	3
0	D	R	R	R
1	R	D	R	D
2	R	R	R	D
3	R	R	D	U

From the data above we can see that the agent has learned a different, but still optimal path from the starting position to the goal. The differences in learned actions are the result of the probabilistically selected actions during training and execution. Due to these probabilistic actions the same optimal path may not be followed every time during execution as shown in the following figure

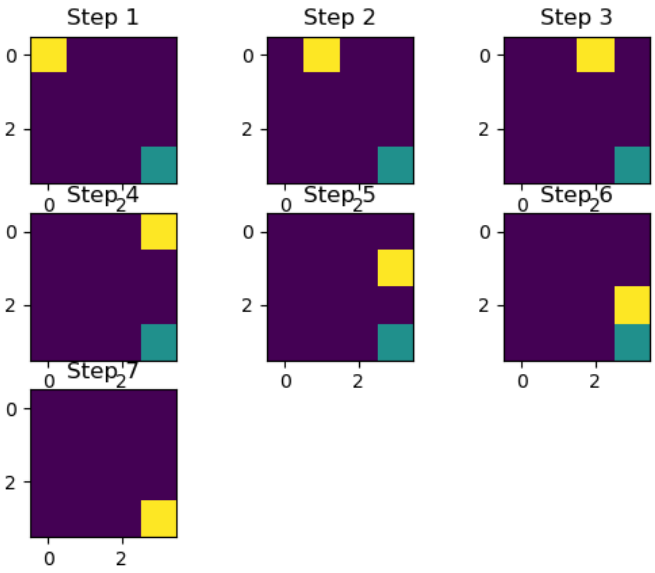

Figure 6: Stochastic Impact on Q-Agent Movements

Here we can see that in step 1 instead of moving down as expected, the agent moves to the right, throwing the agent onto a different path. In this case the agent continues to complete the task in an optimal number of steps.

In other cases an optimal time solution may not occur due to the probabilistic nature of the agent's movements as shown below

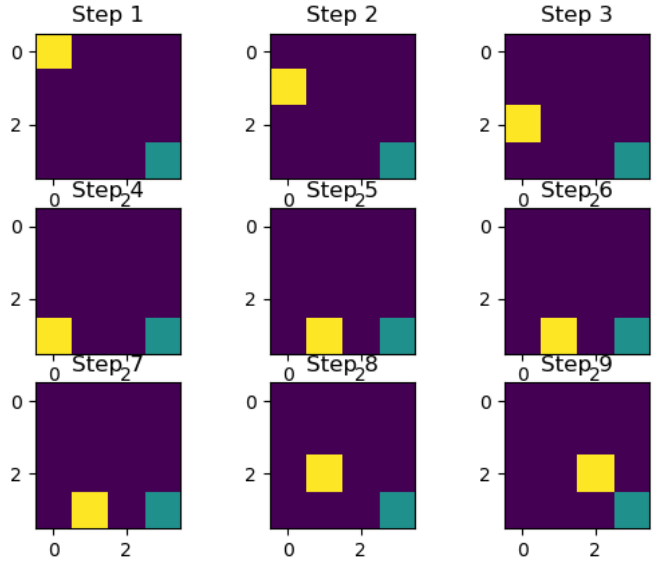

Figure 7: Q-Agent Non-Optimal Execution

Here we can see that due to probabilistic action execution the agent has achieved a non-optimal path to the goal. In the extreme case the agent may take more than nine steps to reach the goal, resulting in the environment terminating due to runtime before the goal is reached.

For the transition probabilities see Section 1.6. All figures and tables included in this report can be generated by executing attached main.py script and checking the generated results directory. For multiple paths of the Q-Learning agent in the stochastic environments please rerun the script.