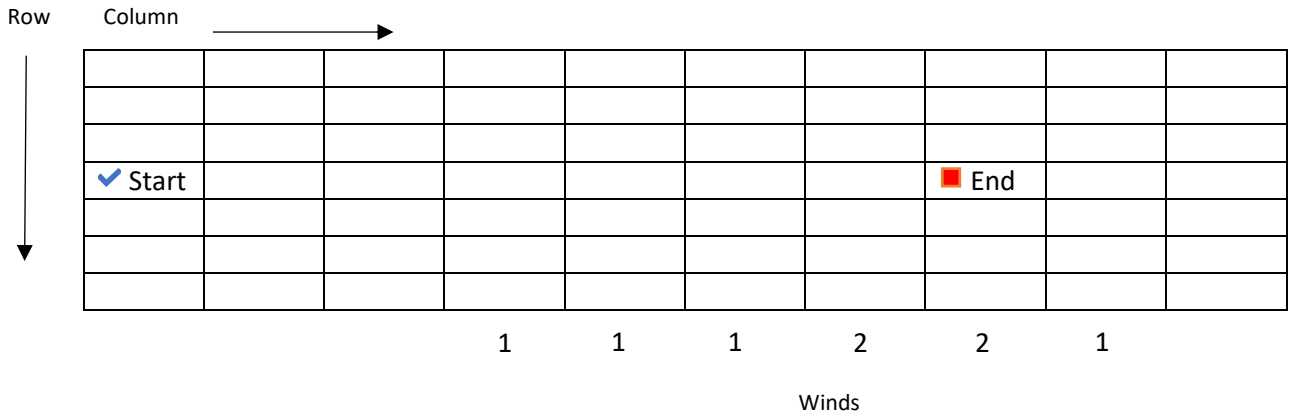
Homework 5

Environment Set up:

In this problem, the environment we are using is known as a "Windy Grid World." Here our agent starts initially at an index of (4,1), where '4' is the row and '1' is the column. Our goal is to reach an ending point of (4,8). In between, we encounter winds in specific columns that can push our agent one or two steps upwards depending on the column we would pass into. The whole grid is a 7x10 sized grid system. This all can be understood by the illustration given below.



Algorithm Explanation

Initializations :

For the working of my algorithm, I make use of SARSA on policy TD control, so hence I initialize the various algorithm parameters. These include alpha, which is the step size (set to 0.5), the row and column values and gamma set to '1' as it's not discounted here. Further, we assign the action-value function; here, we make use of MATLAB's multidimensional arrays. Wherein the various actions, 'up', 'down', 'right',' left,' given by the numbers 1,2,3,4 respectively, are also defined. This manner of initiations allows us to choose the best action for our index. Other parameters introduced here include the timesteps, steps and the number of episodes (which is set to 170).

Iteration :

For our 170 episodes, we will run the SARSA Td policy algorithm. Here we first select our best action based on our policy. The policy that we use is E-greedy policy where 90% of the time we choose the best action (maximum) and 10 % of the time, and we select a random action. Also, here we define our staring step-index (which is (4,1)). Next, we begin our first episode and using a while loop to continue the episode until we reach the end node /index. So, once we choose our first action, we need to update our current state-based off that action. Hence this is done here, and I will explain the various steps below :

- For the action "UP" we would only need to decrement a row value by 1
- For the action " down" we would need to increment the row value by 1

- Next, for the action of " Right, "we would need to increment a column by 1
- And lastly for the action of "left," we would need to decrement a column by 1

Once we have done this because this is a windy grid world, we would have to adjust for the windblown based on the columns. So, we go ahead and do that based on these steps given below :

- If we reached columns 1,2,3&10, we would need to do nothing, and our already updated values would hold good
- If we reached columns 4,5,6,9, we would need to update our row index by decrementing '1.'
- If we reached columns 7,8, we would need to update our row index by decrementing '2'.

Now once we have done this, we would need to check if our new index crossed the grid boundaries and if it did, we would have not to allow it to cross the boundaries. We define the upper, lower, right and left boundaries here. If the agent has crossed the boundaries, we give it back the nearest boundary index and reupdate its index values. Now in this type of environment, the reward is -1 for every step that the agent takes and hence it encourages the agent to find the endpoint as soon as it can. We assign this reward value and then proceed to calculate Q using the formula

Q(S[t],A[t])<---(Q(S[t],A[t])+alpha*(R[t+1]+gamma(Q(S[t+1],At[t+1]))-Q(S[t],A[t]))) .

This helps us update the previous states and preform policy improvement after every time step/step. Also, the policy used is used to help us determine the next action that we would take to help estimate and predict policy evaluation. Once we update the Q values. The newly calculated actions , index values are passed onto the next step to redo all the steps all over again.

These steps are terminated when we reach the final index (4,8). Now we use these computed Q values to do another episode and so on. We observe that around for the 1 st episodes; the agent takes about 1000 ish steps to calculate the end state. Whereas towards the end of 170 episodes, we observe how it takes it, agent, just 19-20 steps. Next, we need to update the values of episodes taken and the number of steps taken in each episode to plot the required graph

Pseudocode: of the algorithm used for the code

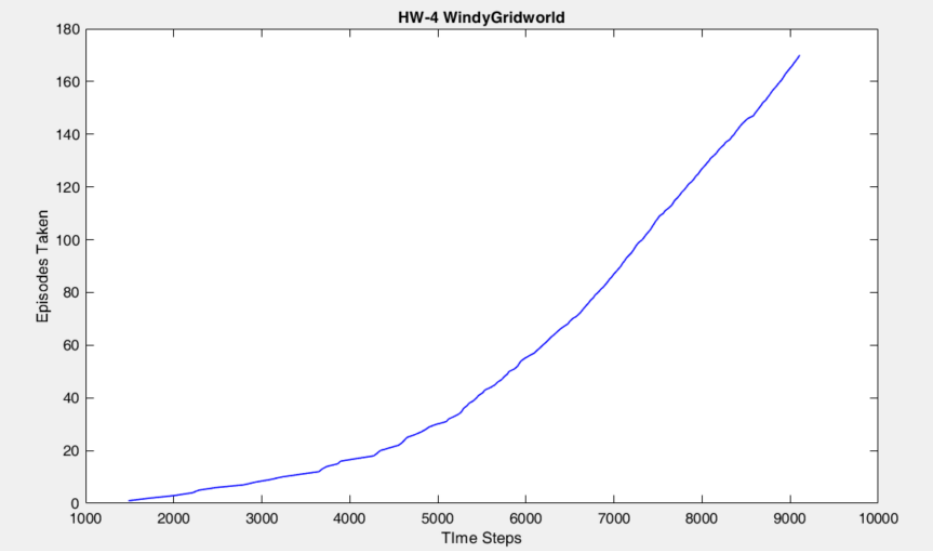
Initialize parameters :alpha,rows,columns,Q,actions
Loop for each episode
    While it not a terminal state
        Compute next step based off policy (e-greedy) and update the next steps
        Update steps if the wind is blown and check for boundary conditions
        Compute Q value using the formula
        Update the new steps and actions
Once a terminal state is reached loop again for 170 episodes and plot the graph of episodes vs. time steps

Time Observed :

11.030613 seconds.

Graphs :

Computed Graph:



Textbook Graph: