CS 747 : Programming Assignment 3

Shalabh Gupta (180050095)

12 November 2020

1 Task : Windy Gridworld

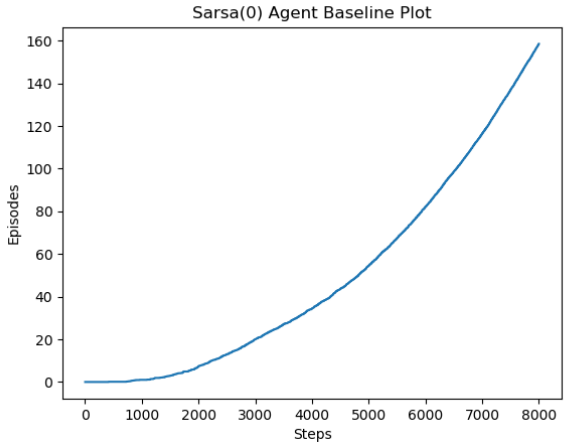
Solving gridworld in a windy setting using Sarsa, Expected Sarsa and Q-Learning algorithms employed by agents. agents.py contains all the 3 algorithms along with epsilon greedy policy implemented while environment.py contains functions which know the wind array and return next state and reward according to state and action recieved. Wind array, as in a general reinforcement learning problem setting, is not known to agents.py

1.1 Environment

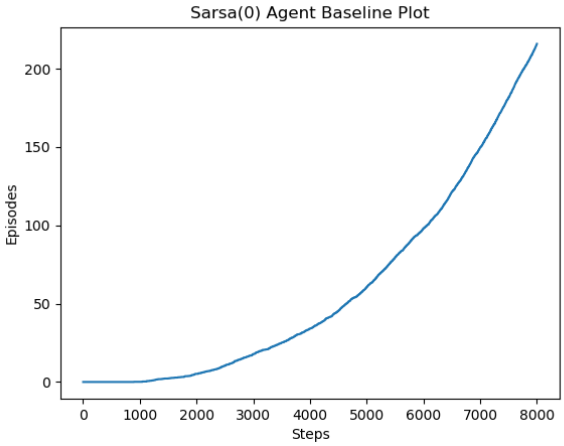
- The environment part of code takes current state and action as inputs, and with the help of wind array decides the next state and reward.
- State for any (x,y) cell is simply $n * x + y$, where n = number of columns(10 in the given case). Hence $x = s//n$ and $y = s\%n$ when s is state number.
- Handling x coordinate is simple since the wind does not affect it. For left and right boundary points, x is being kept the same in case the action is such that it forces us to go out of the maze (left for left boundary and right for right boundary).
- For handling y coordinate, I am adding both the affect of wind and action, and finally checking the bounds by taking minimum with n-1(bottom row) and maximum with 0(top row). Corner cases will automatically be handled using this formulation.
- Stochasticity has only been applied if the initial cell state has non zero wind.

1.2 Agent Algorithms

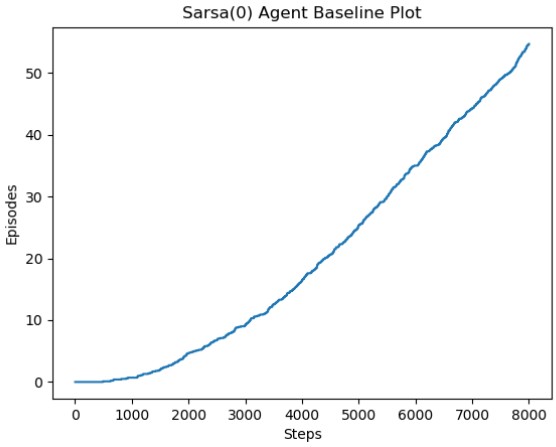
- Running till steps = max steps and updating state to start state once an episode reaches the end state i.e. state == end state. Finally plotting episodes vs steps.
- Point to note in expected sarsa algorithm is the way probabilities $\pi(s, a)$ distributed amongst actions a. Since this is desired to be epsilon-greedy, we distribute 1-epsilon among all the optimal actions(having highest Q[s]), and remaining epsilon among all actions.
- All algorithms have been run on 10 random seeds uniformly between 5 to 50 and average statistics have been plotted i.e. average number of episodes for a given number of steps.
- Epsilon set to 0.1, alpha = 1 and gamma = 1 for all experiments.
- Average number of steps per episode has been calculated by taking average over the last 300 steps i.e. 300/(episode[last] - episode[last-300]).



(a) Non Stochastic Non king



(b) Non Stochastic king moves



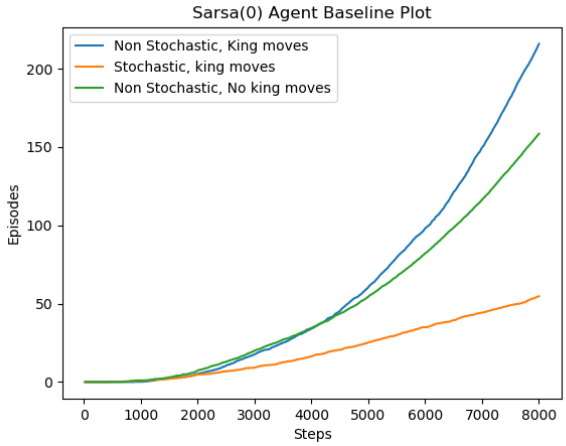
(c) Stochastic king moves

Figure 1: Sarsa(0) Agent

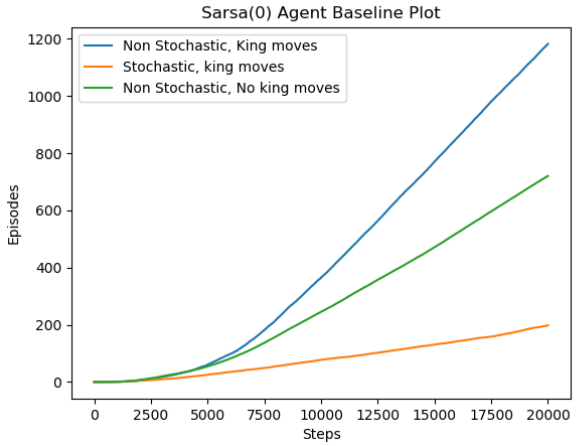
Average number of steps to complete episode in non kings non stochastic = 23.076923076923077
Average number of steps to complete episode in 8 actions non stochastic = 14.925373134328362
Average number of steps to complete episode in 8 actions stochastic = 69.76744186046504

1.3 Interpretation

- The slope of the graph indicates number of episodes per unit time steps.
- In both 4 actions and non stochastic 8 actions, we see the slope increasing. This shows that the number of episodes per unit time step are increasing as we expect it to be since initially there is exploration and with increasing time steps, sarsa learns optimal policy thus taking less steps in going from start to end.
- We see that the number of episodes are higher for 8 actions non stochastic than 4 actions. This is clearly because the 4 actions are a subset of the 8 actions, we have more freedom and thus reaching end state is easier in less number of moves. This justifies the average steps written in both cases.



(a) Small Timesteps

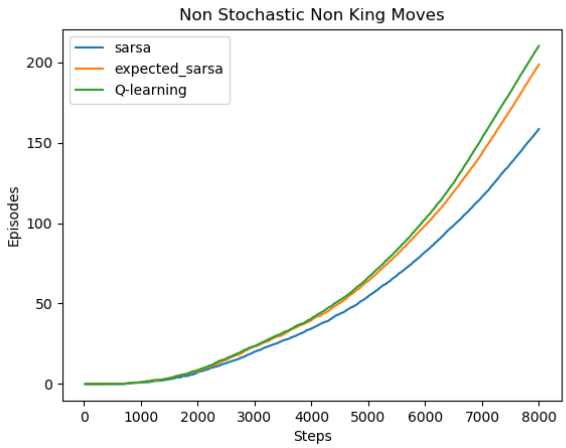


(b) Large timesteps

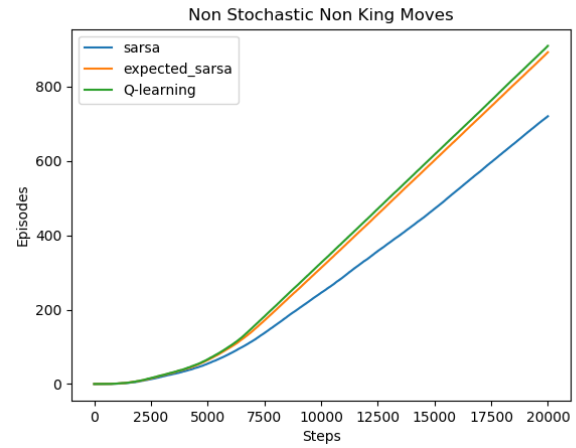
Figure 2: Sarsa(0) Agent combined baseline plots

1.4 Interpretation

- In 20000 timesteps plot we can see that finally the plots become linear, which shows once the optimal policy has been learnt, the average number of steps per episode remain the same.
- Number of episodes per step are much lower in case of stochastic setting, since due to selecting randomly next state, there is high variance in the new coordinates we reach and it takes much more time to explore and reach the terminal state. Therefore average step size is much higher in this case.



(a) Small Timesteps



(b) Large timesteps

Figure 3: Algorithms comparison

Average number of steps to complete episode in sarsa 8000 steps = 23.076923076923077
Average number of steps to complete episode in expected sarsa = 17.964071856287436
Average number of steps to complete episode in Q-learning = 17.751479289940853

1.5 Interpretation

- Q-Learning works better as expected since it is off policy and converges to Q*.
- Since Expected sarsa does not have the stochastic factor introduced in sarsa due to random selection of action A_{t+1}, it has less variance and is generally expected to work better than sarsa. This is indeed the case as can be seen in the plot.
- From the plot with 20000 time steps, we can see all algorithms going linear later as expected.
- Expected sarsa and Q-learning perform almost similar as can be seen from the plot. Average number of steps to complete an episode is also almost same in case of 8000 steps as noted.