CS747
Programming Assignment 3

Parth Shettiwar 170070021

November 2020

Contents

1 Overall Implementation

Contains 2 codes:

`1)Windy.py`
`2)data_gen.py`

The Windy.py files contains code of all algorithms and the formulation of episodic MDP of Windy Gridworld. The Gridworld as defined in book, is a 7*10 array with the wind directions and values shown in the example. The `data_gen.py` code contains the script to produce all the plots. The algorithms are ran for 171 episodes and the plots are saved in the same directory. An average of 10 seeds is taken for each plot.

2 Corner Cases and Convention

The Convention in the program is as follows:

- If any action is taken from a grid cell, then the wind value is added of the current cell.
- For each action taken, the final cell is computed as by adding the individual x and y component to the initial cell.
- After adding these x and y components to the cell, if any of them goes out of bounds of the grid, then they are limited by the grid dimensions and that is noted as the x and y coordinate of the new cell.
  For example, if from second row, first column of the grid, (where wind component is 0), an action of top left(using king moves) is taken, then it will reach the first row, first column cell: as here we will compute as following:
  $x_{old} = 1, y_{old} = 2$

  $$x_{new} = max(x_{old} - 1, 1) \tag{1}$$

  $$y_{new} = max(y_{old} - 1, 1) \tag{2}$$

  Hence, $x_{new} = 1$ and $y_{new} = 1$
  Convention is x coordinate is the column number and y coordinate is the row number. Similar computation for other cases.
- Policy is taken as argmax over the actions. So if many actions have same Q value for a state, nearest or lowest indexed action is taken.
- Rewards are kept as -1 for all cells.
- Q value and Policy are initialised to be 0.
- For stochasticity, the formulation is done exactly done acording to the example given in book.
- The Windy.py code contains 3 functions:

  ```
  1)MDP :
  2)action_take:
  3)new_st
  ```

MDP : Main code and Formulates the MDP. Runs the iterations and updates Q values for each state. It takes 6 paramters: algorithm name(0 for Q-learning,1 for Sarsa,2 for Expected Sarsa), number of actions, aplha or learning rate value, epsilon,flag of stochasticity(if 0, then no stochasticity) and seed. `action_take`: Decides what next action to be taken given the current state, Policy, epsilon and number of actions. `new_st:`Decides what is the next state given an action, current state and whether stochasticity is present or not.

3 Experiments

As mentioned earlier, for all experiments, 171 episodes are considered and average over 10 seeds are taken.

3.1 Standard Sarsa

In this experiment, the Second task of 4 action, epsilon = 0.1, alpha = 0.5 and no stochasticity is implmented and plotted on graph.

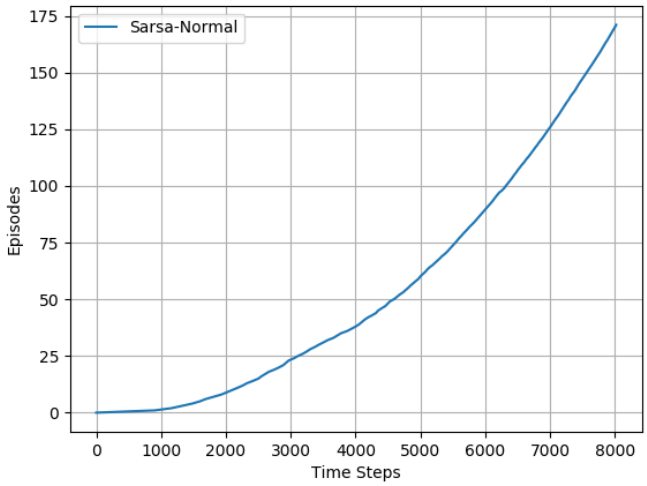

Figure 1: Sarsa: Episodes vs Time steps curve

- We can observe that the algorithm comes to a straight line after about 7000 steps.
- 171 episodes are completed in about 8000 time steps.
- We know for 4 move case, minimum moves are 15 to reach end goal.
- But we can observe that final slope at high time step is not equal to 1/15, but it is indeed less. The inverse of slope shows the number of time steps required to complete 1 episode on an average. Here sarsa actually due to epsilon exploration and because its on-policy has lower slope(more time steps on an average to reach final goal). Also, occasionally, it does achieve min moves of 15 but it gets subsumed when taken average.

3.2 Sarsa with Kings moves

In this experiment, the Third task of 8 action or kings moves, epsilon = 0.1, alpha = 0.5 and no stochasticity is implemented and plotted on graph.

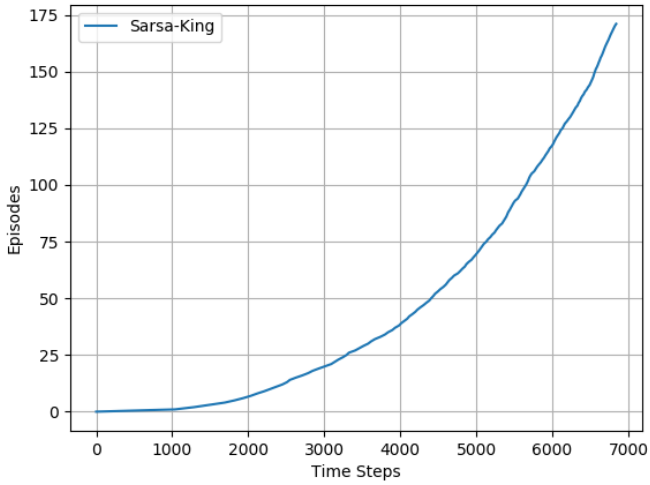

Figure 2: Sarsa-king: Episodes vs Time steps curve

- As we observe, about 6800 steps are required for 171 epsiodes completion, much faster than 4 moves case.
- The slope is also high at high episode, showing that less time steps or less number of moves are required to go from start cell to end cell(This is obvious because king moves give more flexibility or dimension of movement and hence less number of moves are required to reach end state)
- We know for 8 move case, minimum moves are 7 to reach end goal. But we can observe that final slope at high time step is not equal to 1/7, but it is indeed less. This is again due to same reason spotted in first case. But again, as printed on terminal, we do observe cases where it takes only 7 moves to reach goal occasionally.

3.3 Sarsa with Kings moves and Stochasticity

As mentioned in the Example in book, this experiment plots the sarsa curve for kings 8 moves and stochasticity as mentioned in book. Other parameters of epsilon and alpha are same. This is corresponding to Fourth task.

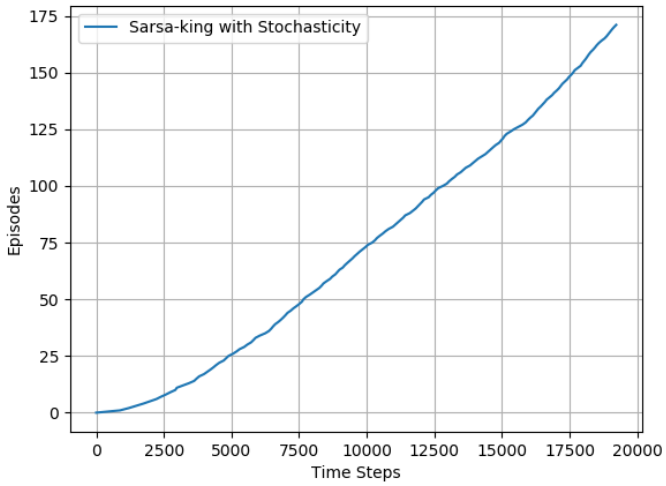

Figure 3: Sarsa-king-stochasticity: Episodes vs Time steps curve

- As clearly seen, this experiment large amount of time steps(19000 around) as compared to previous 2 experiments. Mainly this is because of addition of stochasticity, which makes the learning of optimal Policy slow. Also since its stochasticity, there cannot be fixed policy since for a fixed action, the next state is not certain.
- There is lot of noise as seen in graph, since anytime we cant say how the wind will behave with stochasticity and hence sometimes a lot of time steps can be taken to complete the episode and sometimes less.
- The slope of curve is much much lesser than that of previous 2 experiments (for large time step) again due to same reason.

3.4 Combined Standard 4 move

In this experiment, the Fifth task of 4 action moves, epsilon = 0.1, alpha = 0.5 and no stochasticity is implemented and plotted on graph for all algorithms (Q-learning, Sarsa and Expected Sarsa).

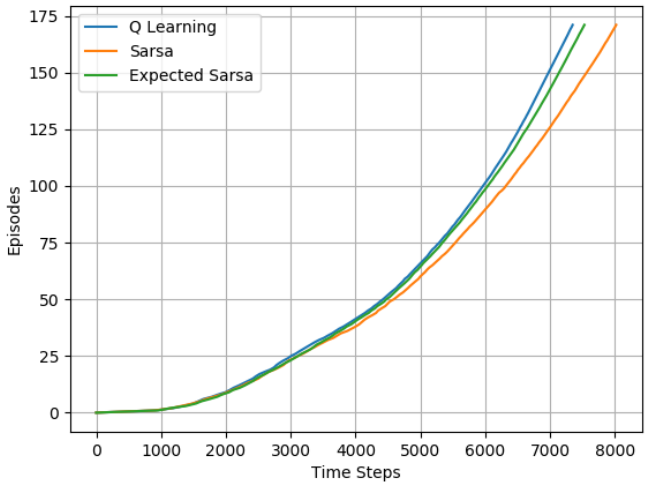

Figure 4: Combined plots for 4 moves: Episodes vs Time steps curve

- Firstly, we can observe that Q-learning converges the fastest(time for 171 episodes). It takes about 7400 steps. Followed by Expected Sarsa(about 7500 steps) and then by Sarsa(8000 steps).
- We can observe the slope of curves of sarsa is lowest. But expected sarsa and Q-learning have almost same slope. In fact at very huge time step (for example for 5000 episodes), the Expected Sarsa curve crosses Q-learning curve.
- As mentioned in the Sutton and Barto book, Expected Sarsa subsumes and generalizes Q-learning while reliably improving over Sarsa. Except for the small additional computational cost, Expected Sarsa completely dominate both of the other algorithms. We are seeing the behaviour of Q-learning performing better than expected sarsa as it is plotted over low number of episodes. At high time steps, either Expected sarsa will be more or equal to Q-learning..
  This point becomes more apparent by observing the following curve plotted for 10,000 episodes for kings moves.

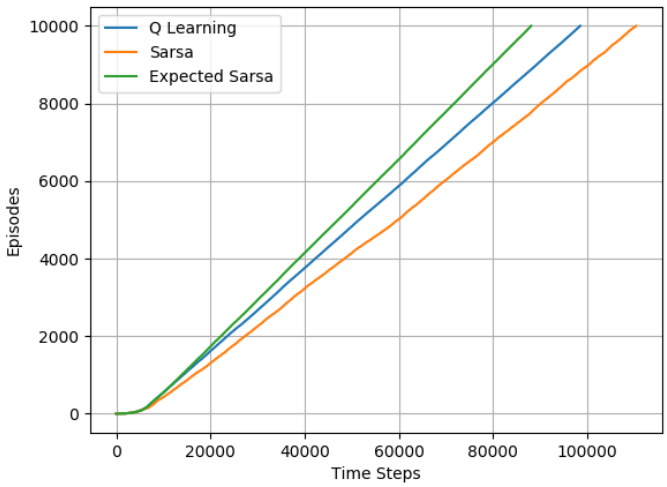

Figure 5: Combined plots for 8 moves for 10,000 episodes: Episodes vs Time steps curve

At start we do observe, both curves being almost equal, but afterwards, the Expected sarsa overtakes Q-learning.

- Also sarsa is less as(or more time steps for 171 episodes) as its completely on-policy and explores more. This is true for subsequent combined plots too.

4 Ablation Studies

4.1 Combined standard 4 move with stochasticity

Observation of Plotting of all three algorithms for the case when stochasticity is there. Rest parameters are epsilon = 0.1, alpha = 0.5 and 4 moves case.

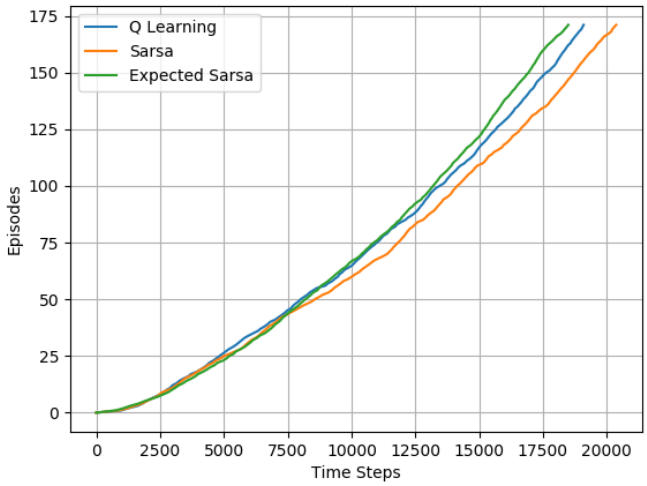

Figure 6: Combined plots for 4 moves with stochasticity: Episodes vs Time steps curve

- As expected, we have lot of noise in the graphs and not exactly straight line. Q-learning slope(average) and expected sarsa are almost same followed by sarsa. However nothing conclusive can be said. On an average Q-learning and expected sarsa always perform better than sarsa.

4.2 Combined Kings 8 move

Observation of Plotting of all three algorithms for the case when kings moves are there. Rest parameters are epsilon = 0.1, alpha = 0.5.

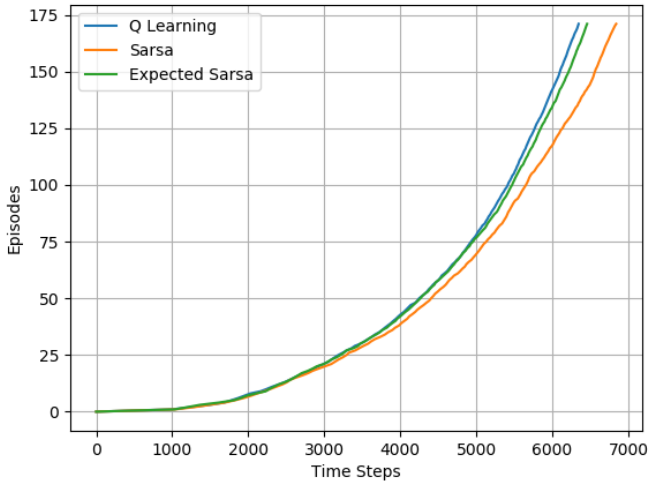

Figure 7: Combined plots for 8 moves: Episodes vs Time steps curve

- As mentioned before also, Q-learning performs always better than other 2 algorithms for small time steps. For large time steps, expected sarsa might overtake.
- Also as expected, with kings moves all algorithms are taking lesser times steps as compared to 4 move case for reaching 171 episode.(Q learning takes 6200 around, expected sarsa takes around 6300 and sarsa takes 6800 steps)

4.3 Epsilon Variation

This experiment is where I varied the epsilon value keeping the algorithm fixed to be sarsa. 4 epsilon values namely, 0.05, 0.1, 0.15, 0.2 were taken.

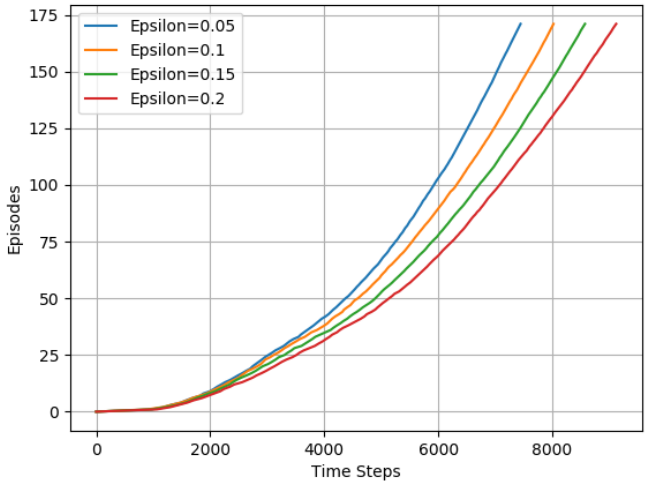

Figure 8: Sarsa 4 move with changing epsilon values: Episodes vs Time steps curve

- With decreasing epsilon(decreasing exploration), the graphs complete 171 episodes in lesser times steps.
- Mainly because, in our case we have Q values initialised to be all 0, and reward are negative, hence the algorithm inherently explores all arms atleast once always without the need for setting epsilon to be high.(if lets say Q value are initialised to be -100, then such observation wont occur and at very low epsilon values, it would take large amount of time steps)

4.4 Alpha or learning value parameter Variation

Lastly in this experiment is where I varied the alpha value keeping the algorithm fixed to be sarsa. 4 alpha values namely, 0.2, 0.35, 0.5, 0.65 were taken.

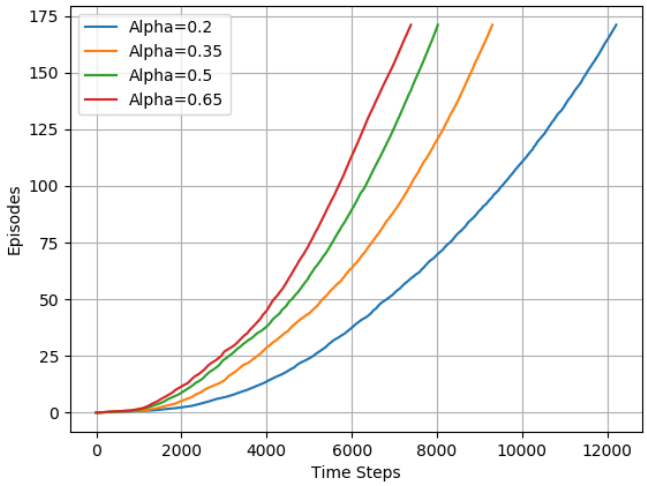

Figure 9: Sarsa 4 move with changing alpha values: Episodes vs Time steps curve

- With increasing alpha values, we get better convergence or lesser time steps to complete 171 episodes. This is because, if alpha is low, it means we arent trusting the current reward and taking more of the already known value. Leading to inclusion of more bias.
- However as observed, at higher and higher alpha values, the different alpha curves start getting closer and probably one time at very high alpha value, the convergence would be affected, leading to more time steps. Hence an optimum alpha should exist.

5 Conclusion

In this assignment, we implemented 3 algorithms control algorithms for Windy Gridworld example. Firstly the MDP was formulated followed by a comprehensive experiment and ablation study was done for the same changing various parameters and commenting on the observations from graph.