CS747: Assignment 3 Report

Manan Sharma (170040067)

November 13, 2020

Task 1

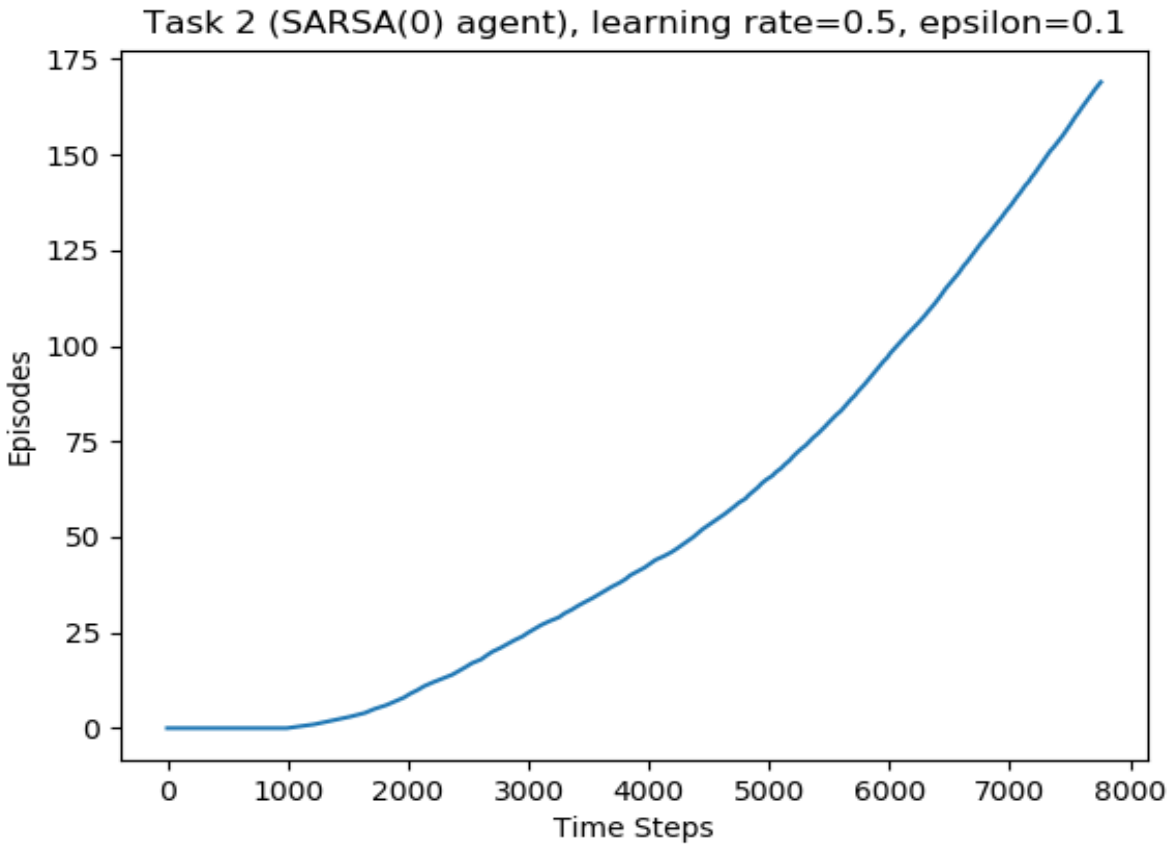
This task involved implementing Windy Gridworld as an episodic MDP. Following the same, the grid (environment) and the agent are implemented as separate interacting entities. The environment is characterised by columnar wind intensity, and has it's own reward function. All rewards for any transition are set to -1, except for the case when the final state is the end state, in which case reward is 0. Notably, while exploring, when the agent reaches a border cell, and if it takes an action, that leads it out of the grid, the environment returns the current state as it's next state along with a reward of -1. Though, for a diagonal move in a border where one component of movement is within the grid, that component of move is allowed to happen. The agent implementation includes the agent type and the epsilon greedy action selection.

All code for agent, environment, experiment runner and plot generator resides in *main.py* file. For all experiments, following the book's example, values of learning rate, epsilon and discount factor are kept as 0.5, 0.1 and 1 respectively. The number of episodes is kept 170 for task 2-4 and 700 for task 5 (to compare convergence).

For each plots, the values are averaged over 10 runs with different seeds.

Task 2

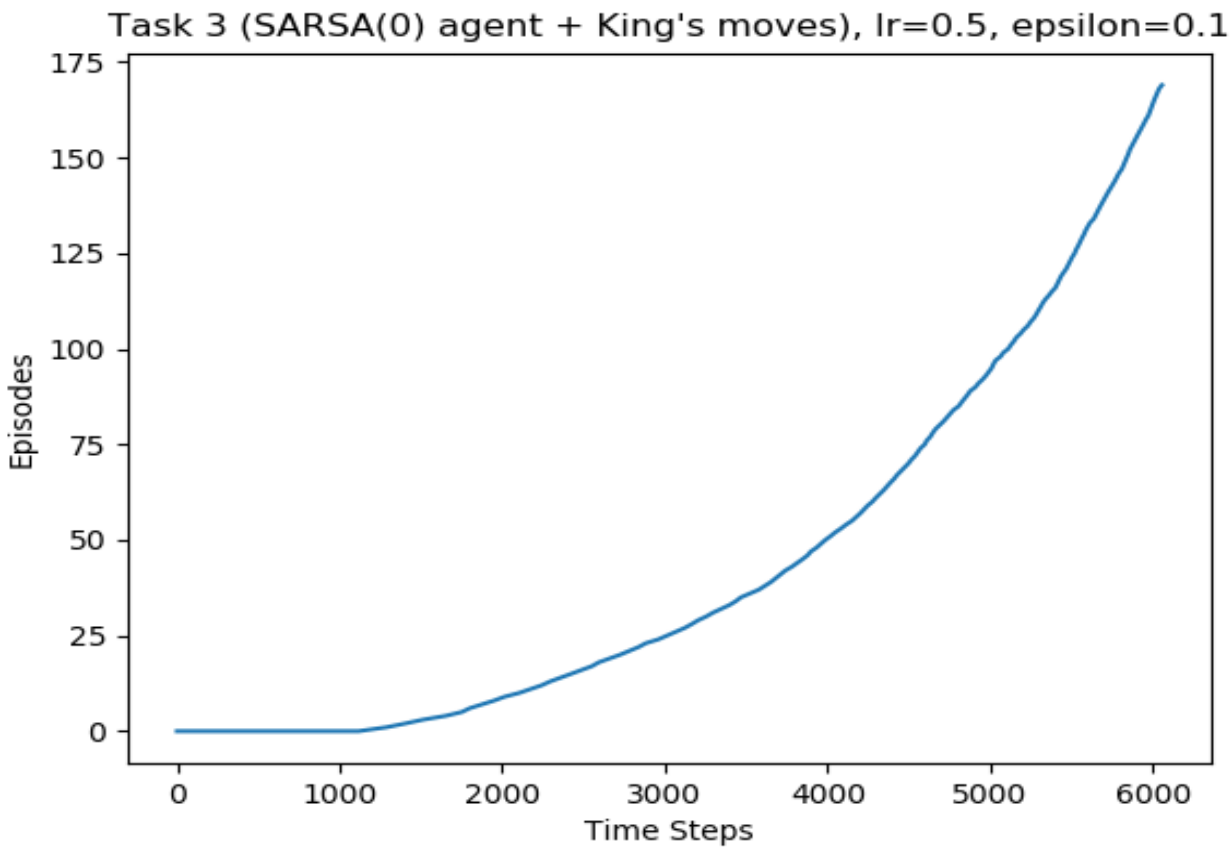
A Sarsa(0) agent was implement just like in Example 6.5 of Sutton and Barto. All parameters are kept exactly same as in the example. The baseline plot thus obtained looks like:



The slope increases in the beginning and reaches a constant value subsequently, indicating convergence to the optimal policy. Hence, after sufficient timesteps, the agent figures out the optimal way to reach the goal through the wind (minimizing the cost).

Task 3

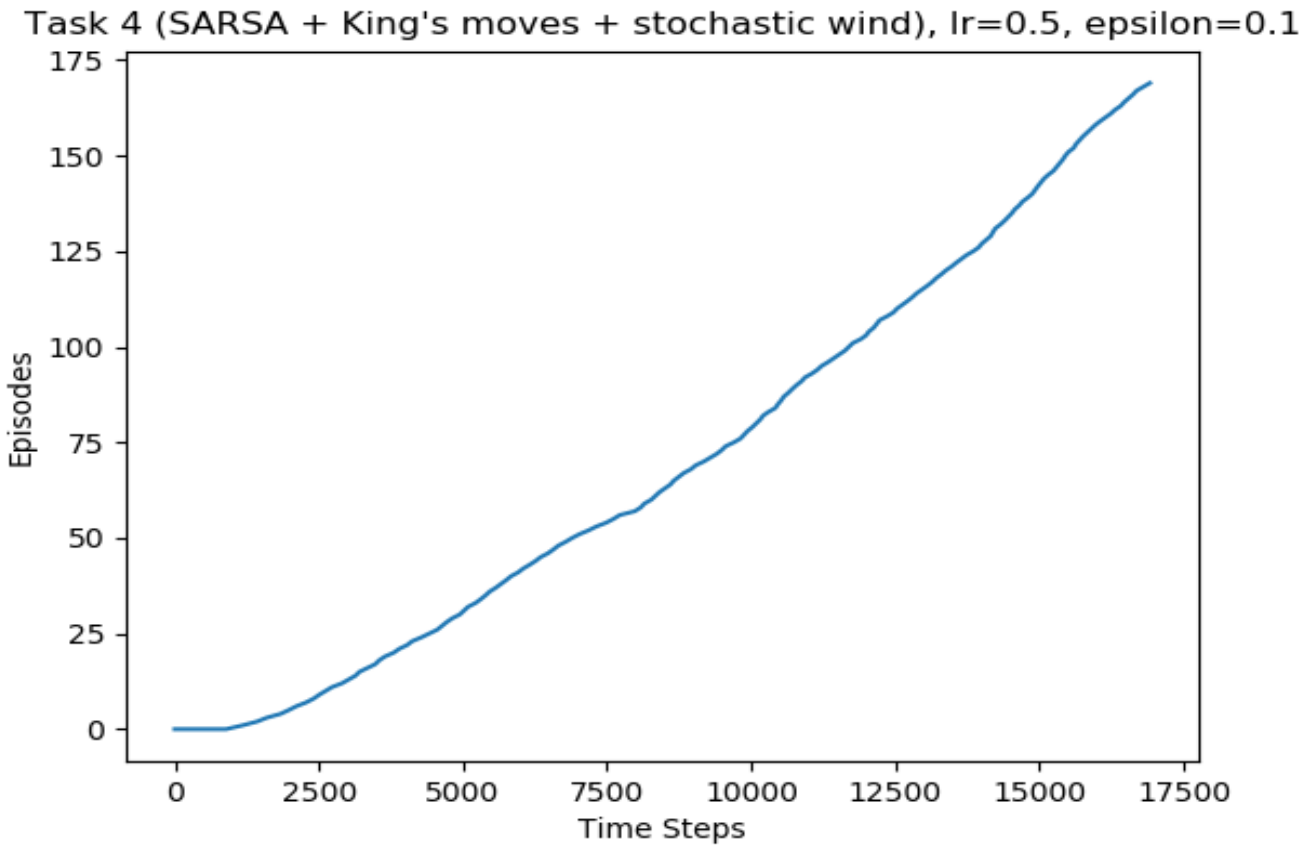
This differs from task 2 in a way such that, now the agent has 8 possible immediate cells where it can jump to, rather than 4 in previous case. When the agent is in a border cell, and it chooses to take one of the 4, now possible, diagonal steps, there can be the case where full movement is restricted. A diagonal step is equivalent of taking one horizontal and one vertical step (2 components). If the move is such that movement in neither component is feasible, the agent remains in its present state and gets a reward of -1. If movement in one component is free, agent moves only in that component. Following plot is thus obtained:



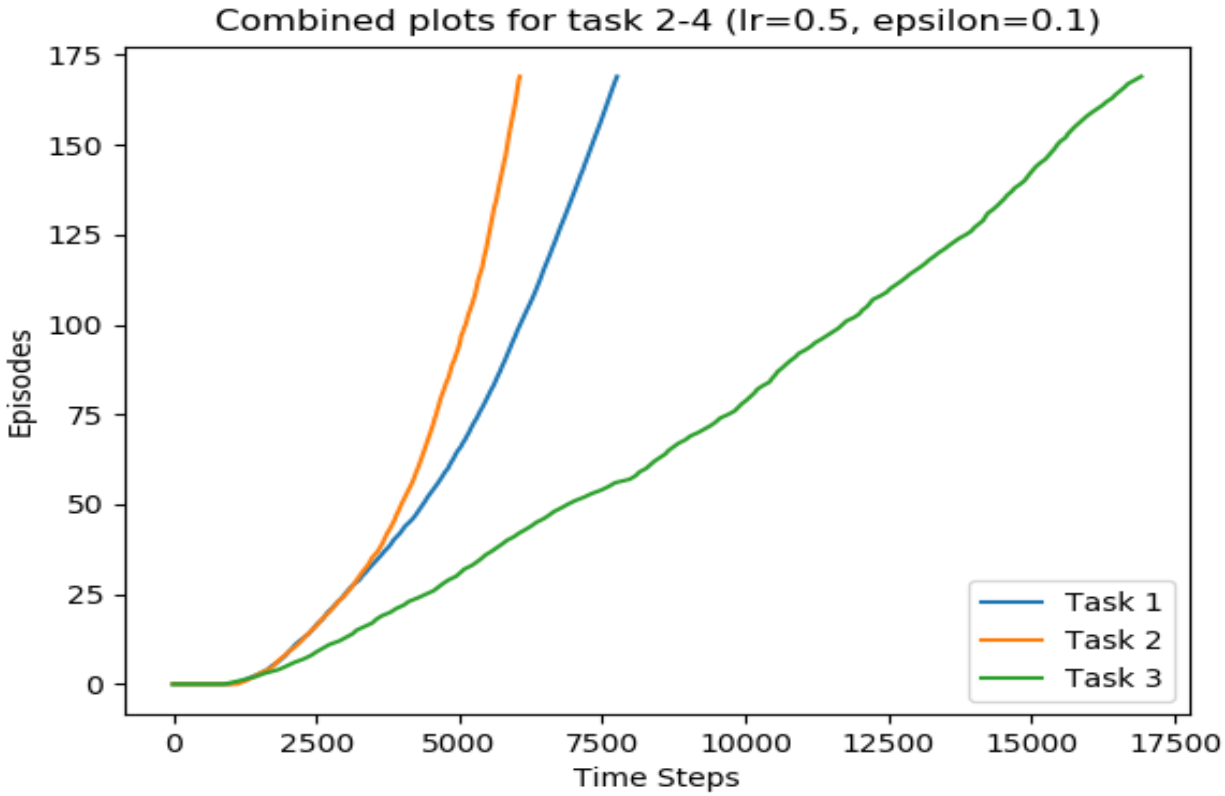
Note that while the shape is similar to the previous graph, the convergence is quite earlier (~6000 timesteps for same number of episodes). This was expected, since agent now has more choices available to itself which would be favourable in many states.

Task 4

This was same as task 2, but a stochastic wind is added. That is, wind in a column has 1/3 chance of increasing in strength by 1, 1/3 chance of decreasing in strength by 1, and 1/3 chance of staying the same. Same border policies are followed as in last task. The following plot is obtained:

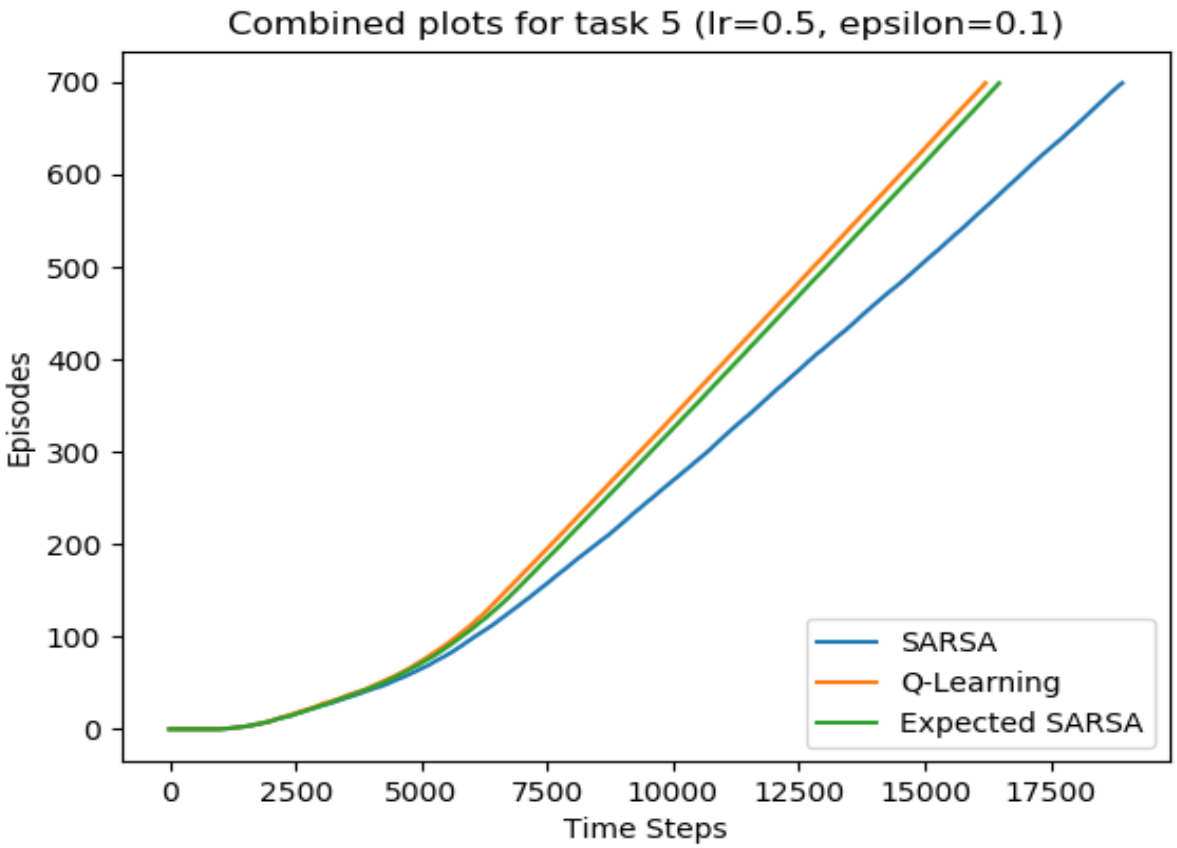


As expected, the convergence is much slower as compared to both the tasks (note the number of timesteps, episode number is same, it has not even converged yet as evident from undulations). This was because, it becomes difficult for the agent to figure out a path for unpredictable wind in the column, a policy for a state learned earlier, might not work the best at a later timesteps depending upon the wind strength. The following plot combines the result of all the tasks:



Task 5

This task involved implementing task 2 with different agents (Q-Learning and Expected-SARSA). The plots generated are as follows (episodes are kept higher to demonstrate relative convergence):



Q-Learning agent converged faster than Expected Sarsa, followed by Sarsa in the end. Expected Sarsa is little bit more computationally intensive than Sarsa, but it eliminates the variance due to random action selection in Sarsa, and as evident, converges faster than Sarsa.