CS 747 : Foundations of Intelligent Learning Agents
Assignment 3

Arka Sadhu - 140070011

October 10, 2017

1 Q-learning

For q-learning the following has been done:

- We continue a particular episode until a terminal state is reached, or the number of iterations for the particular episode has crossed a certain threshold, in this case 1000.
- When the agent is initialized, a q-table for each corresponding state and action is initialized. Here we initialize it with all 0's. A learning rate is suitably chosen.
- The first action is given as random. The reward, next state and event is received from the server. If the goal is reached, the reward is 100, anything else is awarded a reward of -1. The event is either continue, terminated or goal reached.
- We note the events. If it is either terminated or goal reached we update the corresponding q-table and then start a new episode. If it is continued we only udpate the q-table.
- The update of q-table includes two steps. First the action which gives the highest q-value in the next state is chosen. This is then used to update the value in the q-table corresponding to current state and current action.

$$Q[s,a] = Q[s,a] + \alpha * (reward + \gamma * \max_a Q(s',a) - Q(s,a))$$

- When asked for the next action, the agent chooses the action in an epsilon greedy manner from the q-table values corresponding to the current state. The epsilon is decayed in $\frac{1}{n}$ manner where n is the episode number.
- For plotting the graphs, we have used 50 random seeds for each instance 0 and 1, and then taken the average reward for each episode.

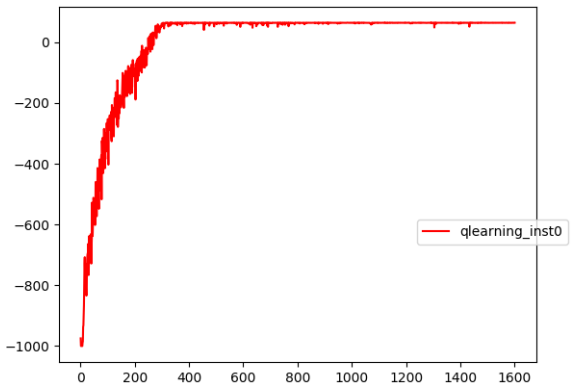

Figure 1: Q-learning for instance 0

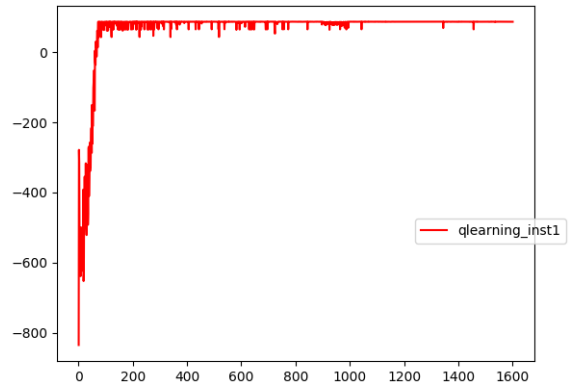

Figure 2: Q-learning for instance 1

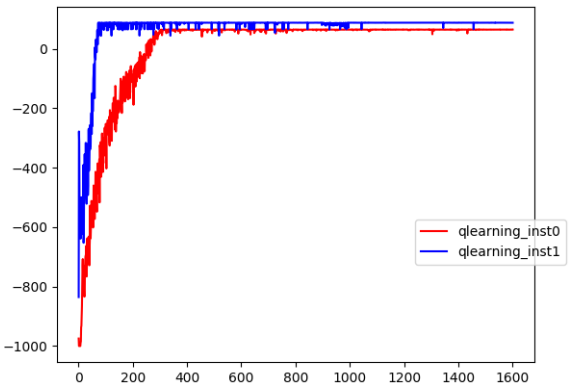

Figure 3: Q-learning both instance 0 and 1

2 Sarsa (λ)

For sarsa (λ) the following has been done:

- All the initializations are the same as in Q-learning. The only difference is that we initialize a new matrix called the eligibility matrix of the same size of the q-table to all zeros. Also we choose the first action at random.
- The same steps are followed as in Q-learning when the reward, next state and the event is given. As an extra we re-initialize the eligibility trace to all zeros when the episode either terminates or goal is reached. Next we update the q-table and the eligibility trace.
- Now given the new state, we choose a new action using some epsilon greedy method on the q-table. Here this strategy is kept the same as that followed for q-table. We call this action as a'.
- We use this a' for computing the update. First we initialize another variable δ which is computed as :

$$\delta = reward + \gamma * Q(s', a') - Q(s, a)$$

- Next depending on the trace strategy we update the eligibility trace corresponding to the current state and action. If accumulative trace is used then we add 1 to the existing value. If replacing trace is used we re-assign the existing value to 1.

- Then for all states and all actions (basically the whole matrix) we do the following udpates:

$$Q = Q + \alpha * \delta * eligibility\_trace$$

$$eligibility\_trace = \gamma * \lambda * eligibility\_trace$$

- For the next action we choose a'.
- Again for the plots we have averaged over 50 random seeds for each episode.

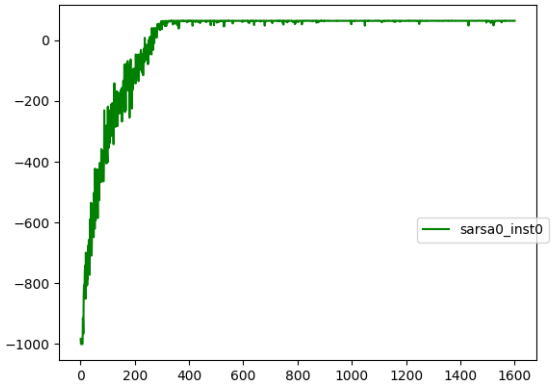

Figure 4: Sarsa 0 for instance 0

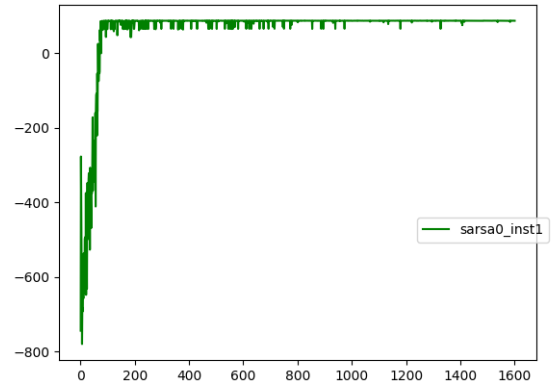

Figure 5: Sarsa 0 for instance 1

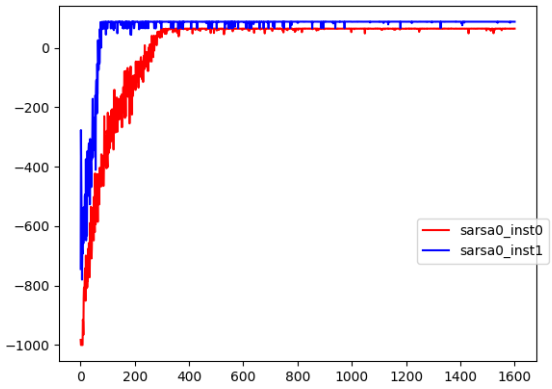

Figure 6: Sarsa 0 for instance 0, 1

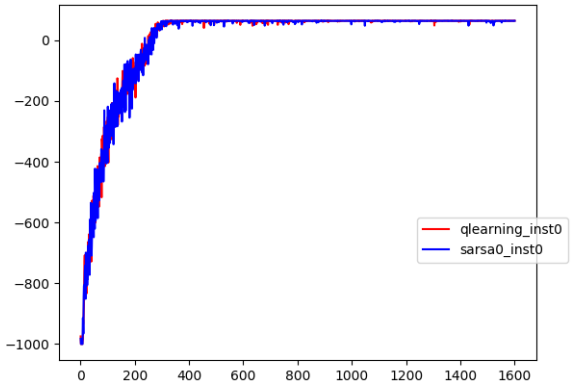

Figure 7: Q-learn and Sarsa0 for instance 0

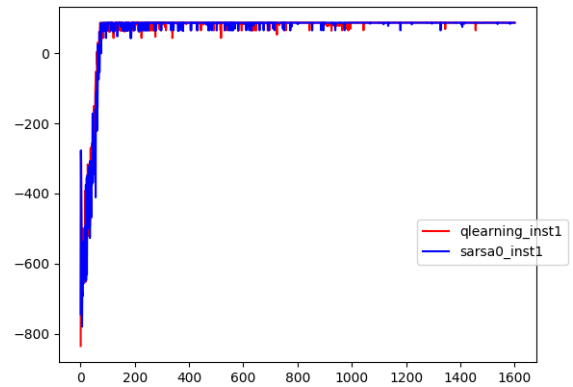

Figure 8: Q-learn and Sarsa0 for instance 1

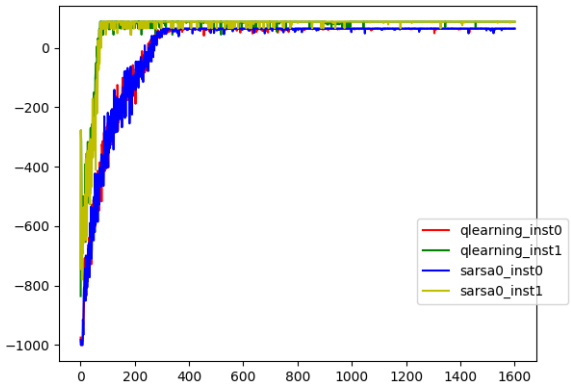

Figure 9: Q-learn and Sarsa0 for both instance in the same graph

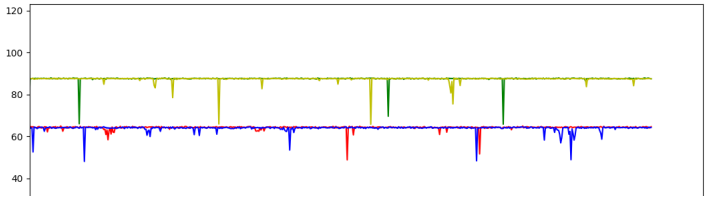

Figure 10: Q-learning and Sarsa0 at optimal value