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. The learning rate is set to 0.5
- 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.

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.

2.1 Selecting the best Lambda

There could be two metrics which can be used to decide the best lambda

- The lambda which converges faster and gives a reasonable sub-optimal value.
- The lambda which gives a higher optimal value.

From the experiments it is found that for smaller lambda the algorithm converges slower but the value to which it converges is higher. On the other hand a larger lambda results in the algorithm to converge much faster but the value to which it converges is sub-optimal that is it is lower.

The two metrics can be thought of as follows. If I want the agent to learn the optimal path irrespective of the number of episodes it takes then the second metric should be considered. If on the other hand we want the total cumulative reward across the episodes to be higher in a finite time then we can go for the first option.

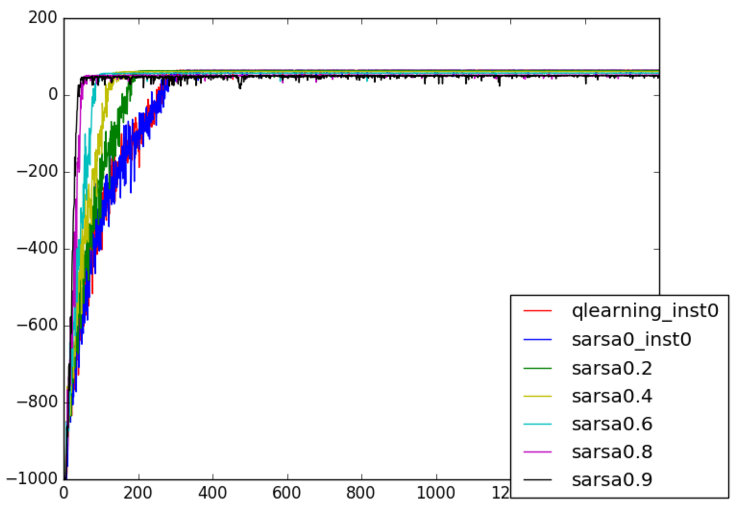

Figure 1: All plots together Instance 0

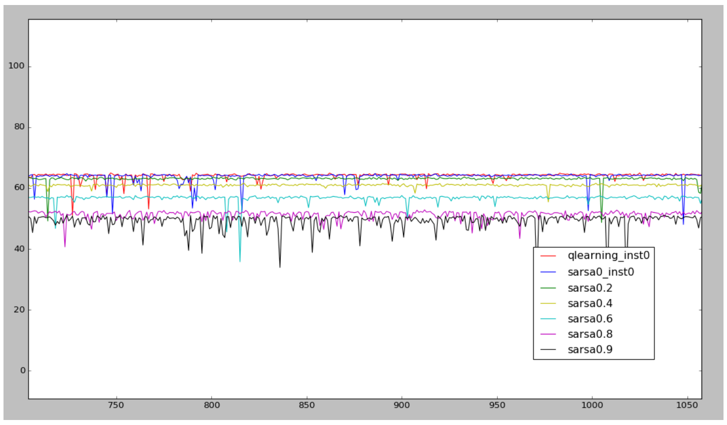

Figure 2: All plots together at convergence Instance 0

If the second metric is considered then $\lambda = 0$ should be chosen since it achieves the optimal value, if instead the first metric is considered $\lambda = 0.9$ should be considered. Results with both are shown in the plots.

3 Plotting Q-learning and Sarsa (λ)

The y-axis is the expected cumulative reward against x-axis which is episode number. Here the expectation is over the 50 random seeds.

3.1 Plots for Instance 0 and 1 for Q-learning and Sarsa 0

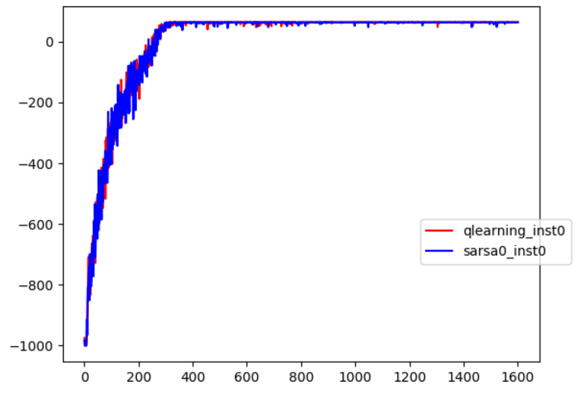

Figure 3: Q-learn and Sarsa0 for instance 0

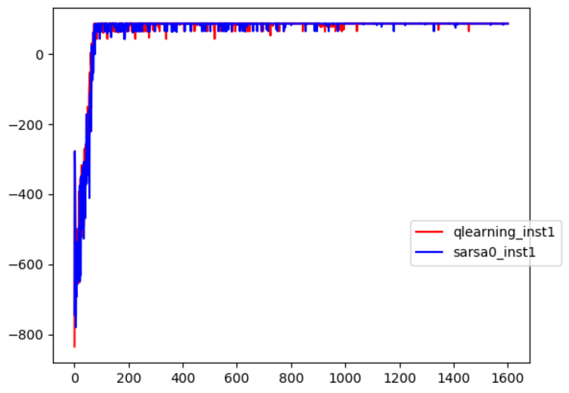

Figure 4: Q-learn and Sarsa0 for instance 1

3.2 Plots for Instance 0 and 1 for Q-learning and Sarsa 0.9

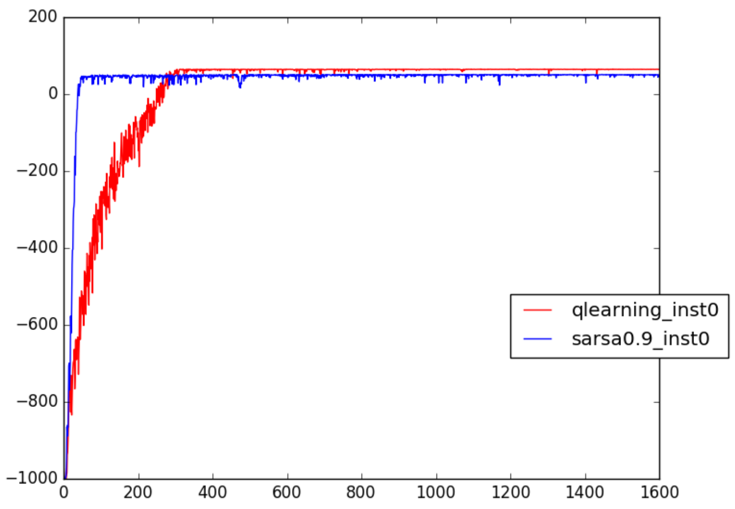

Figure 5: Q-learn and Sarsa0.9 for instance 0

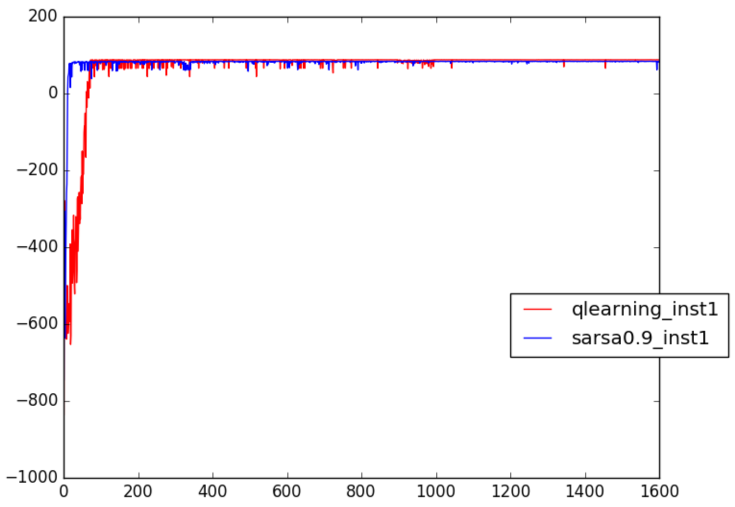

Figure 6: Q-learn and Sarsa0.9 for instance 1

3.3 Inference Q-learning vs Sarsa (λ)

- We note that q-learning and sarsa (0) follow almost the same trajectory. Here no annealing of learning rate is done and the same value = 0.5 is used in both cases. Also the epsilon greedy strategy used in both the cases are the same. For q-learning choosing the current action, while for sarsa (λ) it is the next action.
- Sarsa(0.9) converges must faster than q-learning but the final convergent value is suboptimal.

4 Expected Cumulative Reward over first 500 episodes vs λ

For each value of λ we take the expected cumulative reward over first 500 episodes. Here the expectation is over the episodes as well. Therefore we get one value for each value of λ

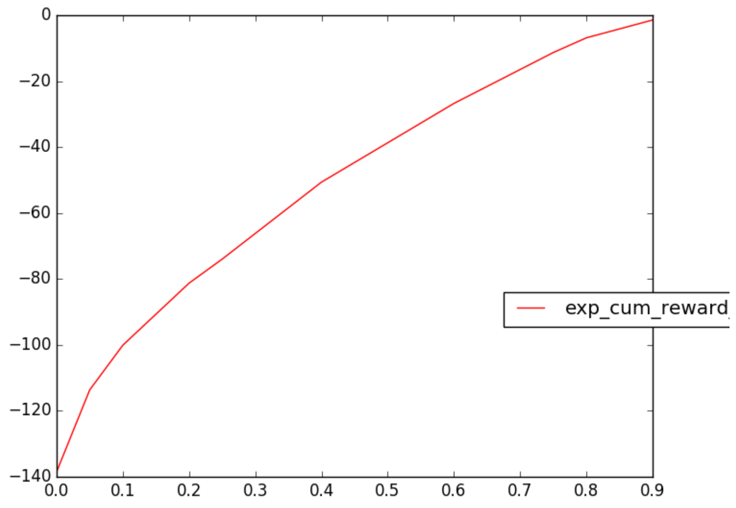

Figure 7: Sarsa instance 0

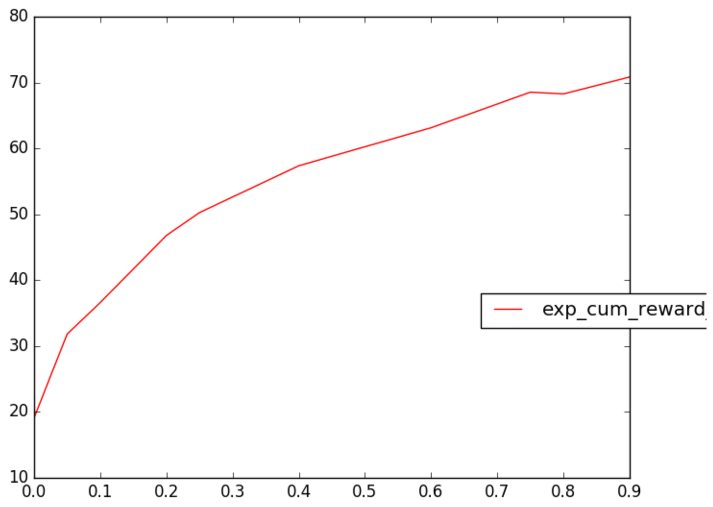

Figure 8: Sarsa instance 1

In both cases the optimal value is got as 0.9

4.1 Inference Expected Cumulative reward vs λ

- The λ values considered here are :$[0, 0.05, 0.1, 0.2, 0.25, 0.4, 0.6, 0.75, 0.8, 0.9]$
- When λ is small then the point where it converges is better but it takes more time. When λ is larger then it converges faster but to sub-optimal point.
- Running on $\lambda = 1$ results in overflow error, which is because in our case $\gamma = 1$ and as a result the eligibility trace blows up because of the large number iteration being done. Therefore it is reasonable to believe the maxima of the curve would be near to 0.9 and hence this value is chosen.

5 Other Inference and Observations

- The most interesting thing was the learning rate parameter. Having a small learning rate lead to much slower convergence, but increasing it lead to faster convergence but occassionally sub-optimal convergent values for some values of λ in sarsa.
- Next is the annealing of ϵ in ε-greedy method. Aggressive annealing didn't lead to much of a differnce for some reason.
- Replacing and Accumulation trace gave no difference at all. Doing a diff across two files corresponding to two random seeds gave no difference at all and the trajectory turns out to be exactly identical. Therefore all further experiments related to λ tuning were done using accumulative trace.