National Taiwan University
Department of Electrical Engineering
Name: Chen-Che Huang
ID: r10946005

112-1 (Fall 2023)
Reinforcement Learning

Assignment 1 Report

Q1. What methods have you tried for async DP? Compare their performance.

I have tried the following three methods for async DP:

- **In-place Policy Iteration**
  Unlike the state values are updated after all states are swept in the vanilla policy iteration, this method updates the state value of a state just after that state is visited. In specific,

$$v(s) \leftarrow \max_{a \in \mathcal{A}} \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v(s') \right) \tag{1}$$

- **In-place Value Iteration**
  This algorithm is very similar to the vanilla value iteration, while the state values are updated immediately after a state is visited by Eq. 1, rather than after all states are swept.

- **Prioritized Sweeping Value Iteration**
  It utilizes a priority queue to store all states and sorts them by the values of their Bellman errors, which is defined by

$$\delta(s) = \left| \max_a \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v(s') \right) - v(s) \right| \tag{2}$$

  , and backups the state with the largest Bellman error. Once the state value of a state s is updated, the Bellman errors of the predecessor states of the state s are re-computed, and then the priority queue is updated and re-sorted.

- **Real-time Value Iteration**
  Instead of sweeping all states, real-time value iteration backups the states along an episode $(S_0, A_0, R_1, S_1, A_1, \ldots, S_T)$ that an agent experiences by Eq. 1.

- **Q-learning**
  Q-learning is an off-policy algorithm, defined by:

$$Q(S, A) \leftarrow Q(S, A) + \alpha \left[R_s^a + \gamma \max_a Q(S', a) - Q(S, A) \right], \tag{3}$$

  where α is the learning rate.

- **Dyna-Q**
  The Dyna-Q algorithm is based on Q-learning algorithm. It employs a model to simulate experiences, and therefore less step count may be required. In this assignment, the model is designed as a table with $|\mathcal{S}|$ rows and $|\mathcal{A}|$ columns, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of all states and actions, respectively. For each transition $S_t, A_t \rightarrow R_{t+1}, S_{t+1}$ that the agent experiences, the model records (R_{t+1}, S_{t+1}) pair in the (S_t, A_t)-th entry. Therefore, the agent can update the action-value function Q not only with the current transition but also with the experiences

stored in the model by Eq. 3. The following is the pseudo code for Dyna-Q:

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left[R_s^a + \gamma \max_a Q(S',a) - Q(S,A) \right]$$

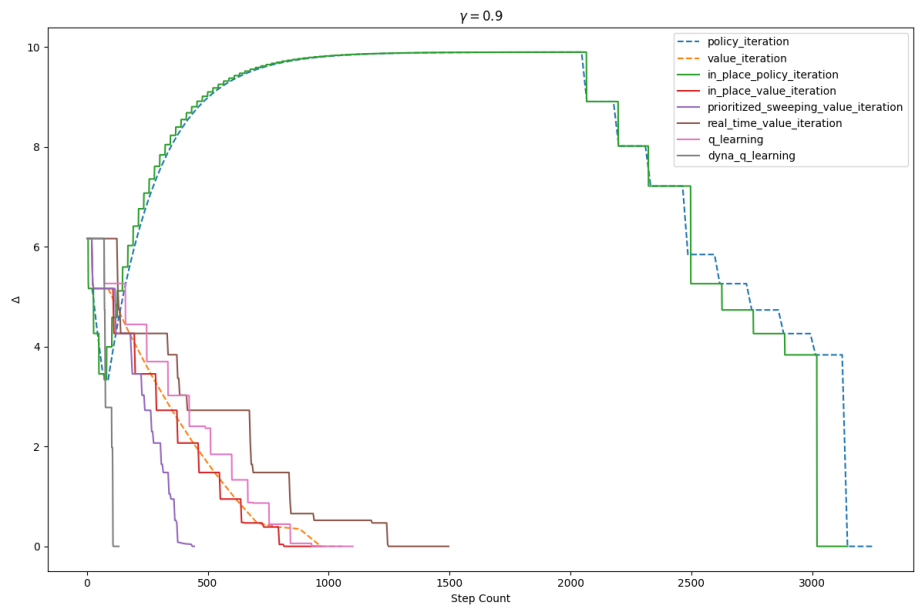
Loop repeat n times:

$S, A \leftarrow$ random sampled observed experience

$R, S' \leftarrow Model(S,A)$

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left[R + \gamma \max_a Q(S',a) - Q(S,A) \right]$$

The figure below shows the error curves of different methods, including synchronized DP methods like policy iteration and value iteration represented by the dashed curves, and the async DP methods mentioned above represented by the solid curves. The y-axis indicates the Δ value, which is the largest absolute difference between the ground-truth state value and the estimated state value in the specific step count. The converged state value estimated by the vanilla value iteration method is regarded as the ground-truth state value. The decreasing of Δ value indicates the efficiency of each algorithm.



We can conclude that in general the asynchronous methods are better then the corresponding synchronized methods. For example, the in-place policy iteration method is better than the policy iteration method, and the in-place value iteration method also converges faster than the value iteration method.

In the value-iteration-based methods, the prioritized sweeping value iteration method converges fastest. The in-place value iteration method and the vanilla value iteration method are the second and third fastest, respectively. The real-time value iteration method is the slowest because the un-converged state value may lead episodes too long to arrive the terminal states.

The Dyna-Q method is the fastest because it learns the action value efficiently with the help of the repeating simulations from the stored experience. It even defeats the prioritized sweeping value iteration method because the latter needs to interact with the environment for all possible actions in each state to find out the best action value, while Dyna-Q chooses only one action in each state by applying ϵ-greedy strategy to its action values.

Q2. What is your final method? How is it better than other methods you've tried?

My final method is **Dyna-Q** with learning rate and , which are the default arguments in the class **DynaQLearning**.

The step counts when the state values converge for each methods that have been tried are presented in the following table:

Method	Step Counts
Policy Iteration	3,256 (×24.7)
Value Iteration	1,144 (×8.7)
In-place Policy Iteration	3,146 (×23.8)
In-place Value Iteration	968 (×7.3)
Real-time Value Iteration	1,496 (×11.3)
Prioritised Sweeping Value Iteration	444 (×3.4)
Q-learning	1,100 (×8.3)
Dyna-Q	132 (×1.0)

The value iteration method visits the states in the order of the state indices, while the prioritized sweeping value iteration method visits the state with the largest Bellman error to update the corresponding state value. A significant Bellman error indicates that the state has greater potential for improvement compared to other states. Therefore, visiting such states first and updating their state values can dramatically boost the convergence.

However, the prioritized sweeping value iteration method needs to consider all possible actions in each state, while Dyna-Q only consider one action by the ϵ-greedy strategy in each state. Therefore, the latter needs less step count.