'''

CS 5180 Fall 2022

Exercise 5: Temporal-Difference Learning

Hongyan Yang

'''

1. RL2e 8.1 Planning vs. n-step returns

(a). Multi-step bootstrapping methods could do as well as Dyna method. Because these two approaches both apply information to update the estimate of not only the current state, but former states as well. By doing this, they can both extract ample information from limited complete episodes to approach the optimal policy and perform better than the one-step method without planning.

(b). We can also do n-step returns for the planning phase (f). It has advantages as follows: 1). n-step planning offers more stable estimates of the value function, thus makes the convergence less volatile. 2). n-step planning will apply the most recent $R, S' \leftarrow Model(S, A)$ updated by model more efficiently by taking recent updates into account, making it converge faster.

It has disadvantages: 1). It's more memory consuming if we planning by recording the last n-step trajectory following each state rather than only one step.
2). It's more complex to compute. 3). It reacts slower to the change of environment.

(c). ① Computational experiments to verify problem ⓐ. for Dyna-Q: we have updates for the Q function from interacting with the environment as well as simulations from the model: both are $Q(S, A) = Q(S, A) + \alpha[R + \gamma max_a Q(S', a) - Q(S, A)]$ to use latest information from the environment to update Q function for current and former state-actions. The underlining idea is similar in n-step case. Take n-step Sarsa as an example. We update Q functions by taking future steps into account: $G = \sum_{i=\tau+1}^{min(\tau+n, T)} \gamma^{i-\tau-1} R_i$, then $Q(S_\tau, A_\tau) \mathrel{+}= \alpha[G - Q(S_\tau, A_\tau)]$.
So they are expected to have similar performance.

(c). (b) Using n-step returns for the planning phase. We have

$G = \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$, and each $R_i$ is returned from the latest model of the environment, by $R, S' = Model(S, A)$.

In this way, we have a more stable of Q function updates in the planning phase with the help of up-to-date model and multiple steps correction.

2a). Tabular Dyna-$Q^+$

Initialize $Q(S,a)$ and $Model(S,a)$ for all $s \in S$ and $a \in A(s)$

Loop forever:

(a). $S \leftarrow$ current (nonterminal) state

(b). $A \leftarrow \varepsilon$-greedy $(S, Q)$

(c). Take action $A$; observe resultant reward, $R$, and state, $S'$

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

(e). $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)

(f). Loop repeat n times:

$S \leftarrow$ random previously observed state

$A \leftarrow$ random action $\in A(S)$

if $(S,A)$ has been previously observed:

$I$ = time steps since they are last observed

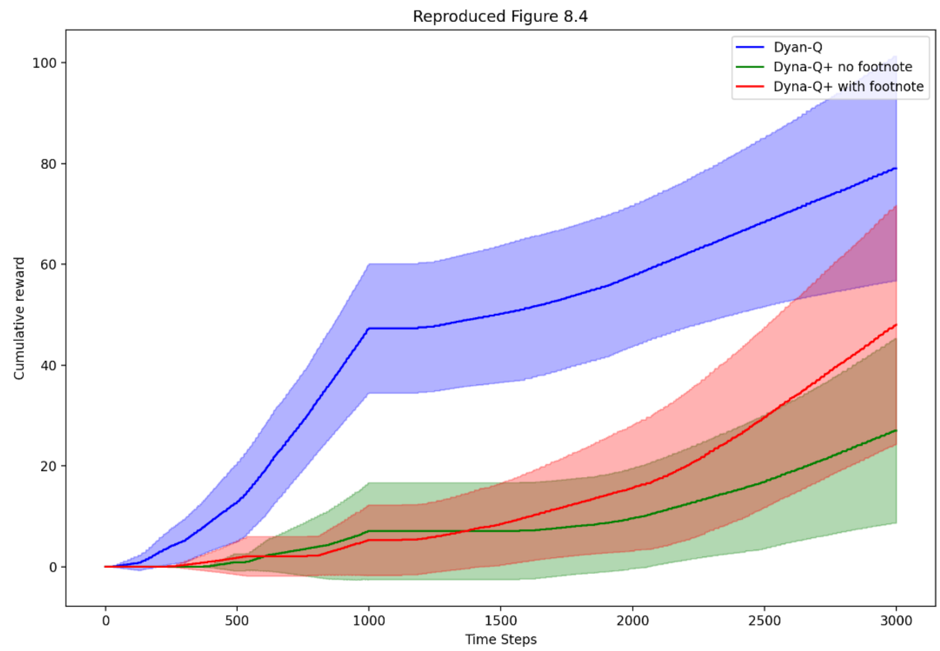
$R, S' \leftarrow Model(S,A)$

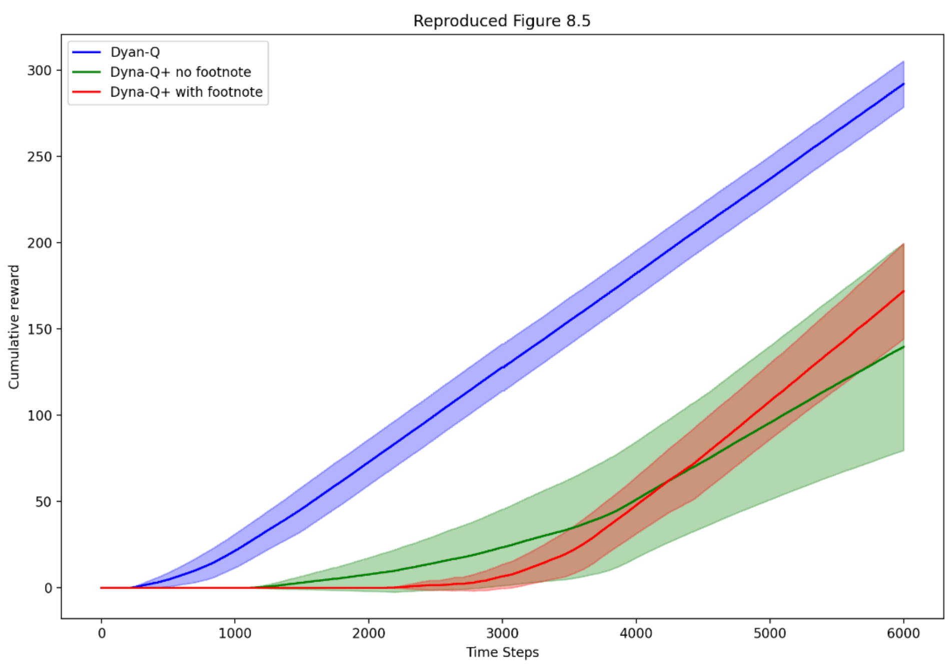
$Q(S,A) \leftarrow Q(S,A) + \alpha[R + K\sqrt{I} + \gamma \max_a Q(S',a) - Q(S,A)]$

else:

$Q(S,A) \leftarrow Q(S,A) + \alpha[0 + \gamma \max_a Q(S,a) - Q(S,A)]$

2.(b) Code/plot: Reproduce the Figures 8.4 and 8.5.



2. (c) Written: Did the footnote matter? Why or why not?

2(c) The footnote does matter, as can be seen from the plots above. This is because when the agent has "tried" actions never seen before in the planning step. It offers the agent more exploratory opportunities in the planning step. Thus, it enables the agent to outperform Dyna Q and Dyna $Q^+$ without footnote in a changing environment.

3. (a). Under the approach of applying bonuses to the reward function, the previous state-action pairs' Q values will all affected by the updated reward function, thus the exploratory of the agent will be "long-sighted" and likely to be adapted to a changed environment. On the other hand, the approach of applying bonuses during action selection will only affect the state-action pair at that specific time step and does little help to find a complete new optimal trajectory under a changed environment.

(b). Advantage of applying bonuses to the reward function: Add time bonuses to the Q values directly enables the model to be exploratory "long sightedly" to find a complete optimal new policy. Disadvantage: The model is sensitive to hyperparameter like kappa especially under this approach.

Advantages of applying during action selection: Being exploratory during early steps makes the model converge to the current optimal policy more quickly and it's easy to apply. Disadvantage: Being exploratory at only one seperate time step makes exploratory "short sighted" and cannot adapt to an environment that changed a lot.

4. 3 points. [5180] (RL2e 8.5) Dyna-Q for stochastic environments.

4(a). Tabular Dyna-Q in a stochastic environment

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in S$ and $a \in A(s)$

Loop forever:

(a) $S \leftarrow$ current (nonterminal) state

(b) $A \leftarrow \varepsilon\text{-greedy}(S, Q)$

(c) Take action $A$; observe $R$ and $S'$

(d) $Q(S,A) \mathrel{+}= \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$

(e) Count $(S,A,S',R) \mathrel{+}= 1$ in $Model(S,A)$

(f) Loop repeat $n$ times:

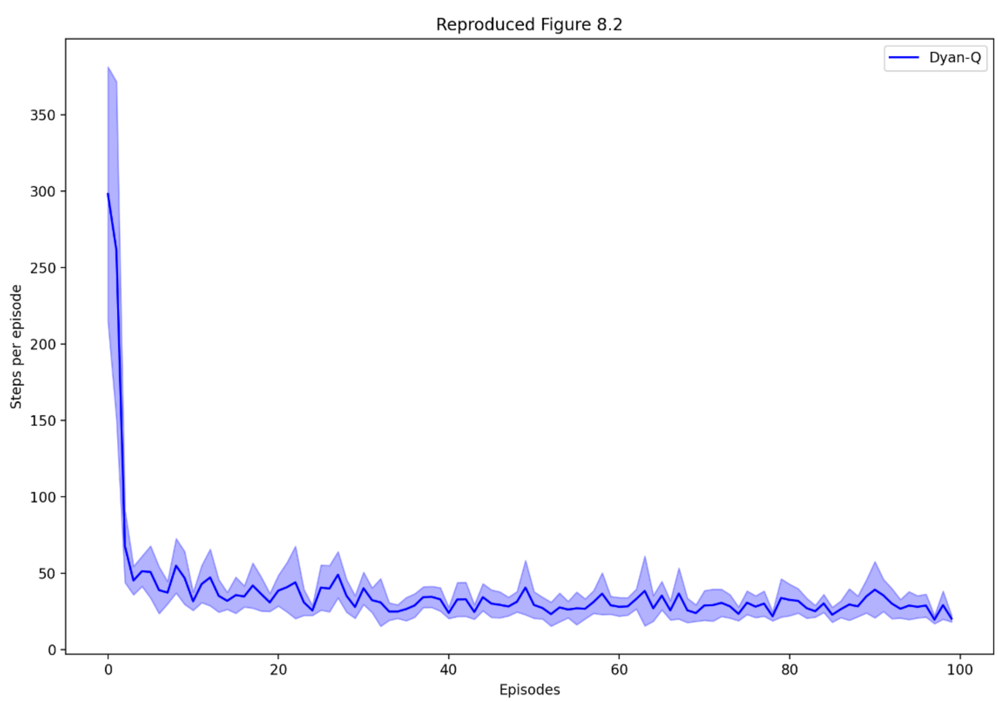
$S \leftarrow$ random previously observed state

$A \leftarrow$ random action previously taken in $S$

Pick $R, S'$ based on the frequencies of happening from $Model(S,A)$

$Q(S,A) \mathrel{+}= \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$

4. (b) Code/plot/written:

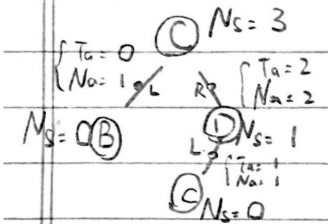


5. 5 points. Manual MCTS

Right = 1 , Left = 0

Iteration 4:

Previous Tree




Selection: $C \xrightarrow{R} D$

Expansion: 1 (do not use RNG) $\xrightarrow{R} E$

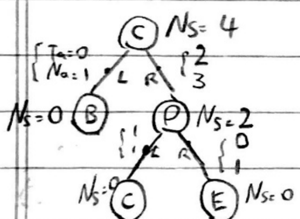
Simulation: $E \xrightarrow[0]{L} D \xrightarrow[0]{L} C \xrightarrow[1]{R} D \xrightarrow[0]{L} C \xrightarrow[1]{R} D \xrightarrow[1]{R} E \xrightarrow[0]{L} D \xrightarrow[0]{L} C \xrightarrow[0]{L} B \xrightarrow[1]{R} C \xrightarrow[1]{R} D \xrightarrow[0]{L} C \xrightarrow[0]{L} B \xrightarrow[1]{R} E \xrightarrow[1]{R} D \xrightarrow[0]{L} C \xrightarrow[0]{L} B \xrightarrow[0]{L} A \xrightarrow[0]{L}$ Terminal State

Reward = 0

Back up:

Iteration 5:

Previous Tree




Selection: $C \xrightarrow{L} B$

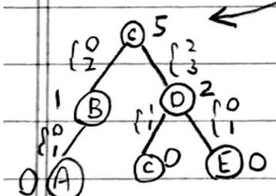
Expansion: 0 (use RNG) $\xrightarrow{L} A$

Simulation: $A \xrightarrow[0]{L}$ Terminal State

Reward = 0

Backup:

Iteration 6:




Selection: $C \xrightarrow{R} D \xrightarrow{L} C$

Expansion: 1 (use RNG) $\xrightarrow{R} D$

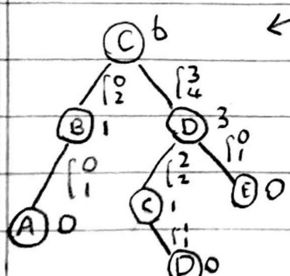
Simulation: $D \xrightarrow[1]{R} E \xrightarrow[0]{L} D \xrightarrow[1]{R} E \xrightarrow[1]{R}$ Terminal State

Reward = 1

Back up:

Iteration 7:

Previous Tree




Selection: $C \xrightarrow{R} D \xrightarrow{L} C$

Expansion: 0 (do not use RNG) $\xrightarrow{L} B$

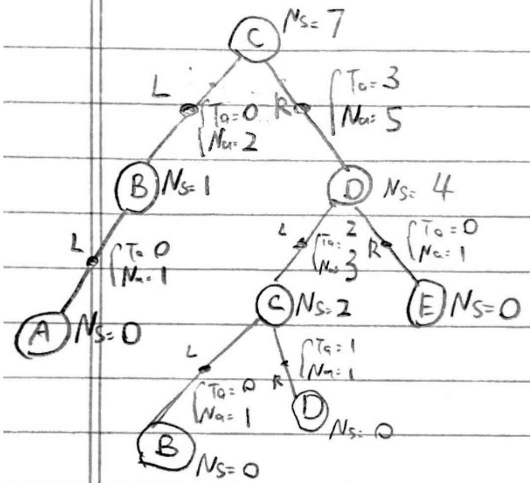
Simulation: $B \xrightarrow[0]{L} A \xrightarrow[1]{R} B \xrightarrow[1]{R} C \xrightarrow[0]{L} B \xrightarrow[1]{R} C \xrightarrow[0]{L} B \xrightarrow[0]{L} A \xrightarrow[0]{L}$ Terminal State

Reward = 0

Back up:

Iteration 8:

Previous Tree




Selection: $C \xrightarrow{R} D \xrightarrow{R} E$

Expansion: 1 (use RNG) $\xrightarrow{R}$ Terminal State

Simulation: None

Reward: 1

Backup: