“To succeed, planning alone is insufficient.
One must improvise as well.”

Isaac Asimov, *Foundation*

CMPUT 365
Introduction to RL

Marlos C. Machado

Classes 20-21/35

Coursera Reminder

You **should be enrolled in the private session** we created in Coursera for CMPUT 365.

I **cannot** use marks from the public repository for your course marks. You **need** to **check**, **every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

At the end of the term, I **will not port grades** from the public session in Coursera.

If you have any questions or concerns, **talk with the TAs** or email us `cmput365@ualberta.ca`.

Reminders and Notes

- What I plan to do today:
  - Start talking about Planning and Learning with Tabular Methods (Chapter 8)
- Practice quiz and programming assignment are due on Monday.
- Midterm 2 is Wednesday.
  - It will be written here, in the classroom (50 minutes).
  - I need you to bring some piece of ID.
  - There will be three versions of the same exam, since the classroom is so “cozy”.
  - It is closed-book, no calculators are allowed.
  - There will be a formula sheet.

Please, interrupt me at any time!

https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Chapter 8

Planning and Learning with Tabular Methods

Motivation

- Can we be more sample efficient?
- Do we really need to learn only from the transitions we observe?
- Can we learn how the world works and then imagine what we would do in different situations?

Model-based Reinforcement Learning

Models and Planning

- Model: Anything that an agent can use to predict how the environment will respond to its actions. Given a state and an action, a model produces a prediction of the resultant next state and next reward.
- There are different types of models, such as *sample models*, *expectation models*, and *distribution models*.
- Models can be used to mimic or *simulate* experience.
- Planning is any computational process that takes a model as input and produces or improves a policy for interacting with the modeled environment:

$$\begin{tikzcd}\text{model} \arrow[r, "\text{planning}"] & \text{policy}\end{tikzcd}$$

Planning

- State-space planning: a search through the state space for an optimal policy or an optimal path to a goal. Actions cause transitions from state to state, and value functions are computed over states. ← **Our focus!**
- Plan-space planning: search through the space of plans. Operators transform one plan into another, and value functions, if any, are defined over the space of plans. Plan-space planning includes evolutionary methods and “partial-order planning”.

A Common Structure for State-Space Planning

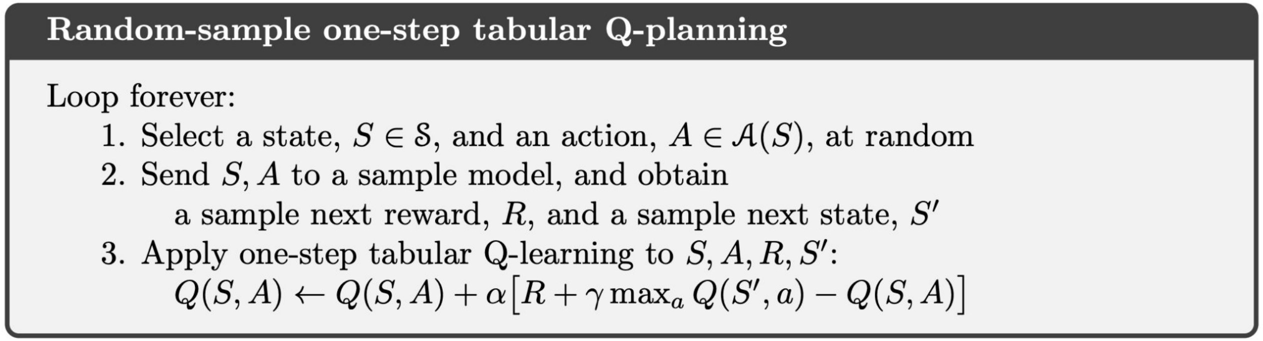
1. All state-space planning methods involve computing value functions as a key intermediate step toward improving the policy,
2. they compute value functions by updates or backup operations applied to simulated experience.

$$\begin{tikzcd}\text{model} \arrow[r] & \begin{array}{c}\text{simulated}\\\text{experience}\end{array} \arrow[r, "\text{backups}"] & \text{values} \arrow[r] & \text{policy}\end{tikzcd}$$

Arguably, other state-space planning methods differ only in the kinds of updates they do, the order in which they do them, and in how long the backed-up information is retained.

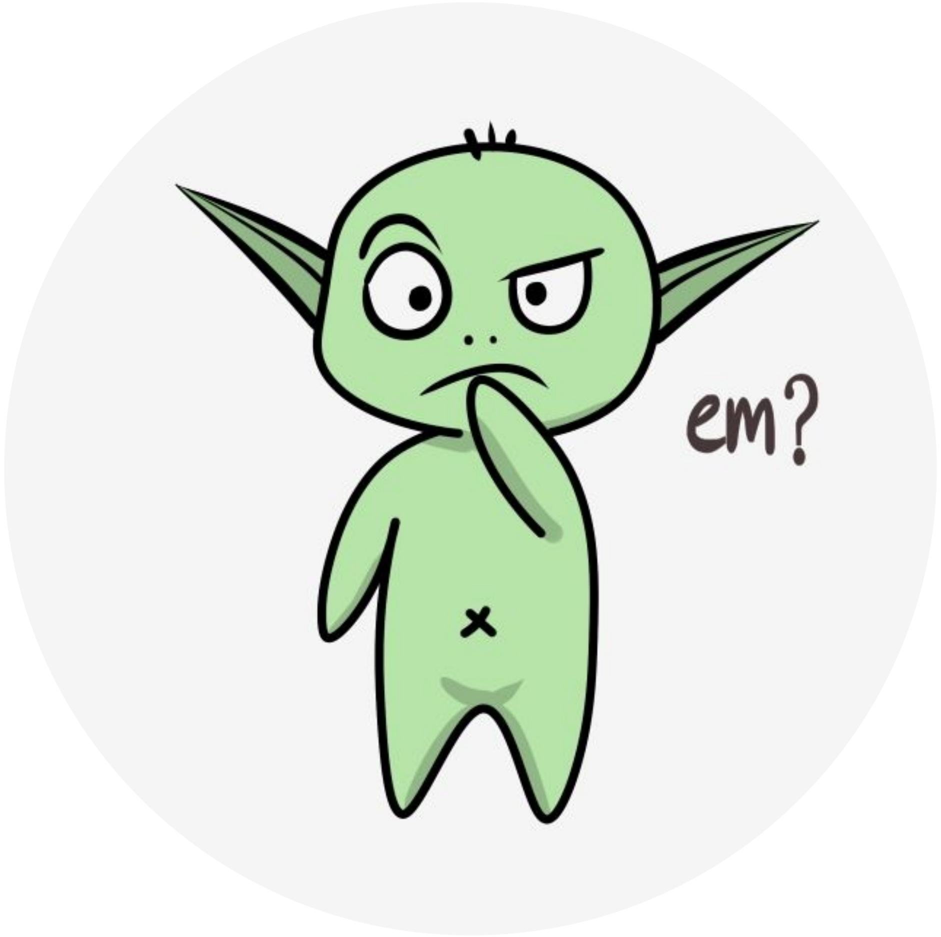
Planning vs. Learning

- The heart of both learning and planning methods is the estimation of value functions by backing-up update operations.
  - Planning uses simulated experience generated by a model.
  - Learning methods use real experience generated by the environment.
- Learning methods require only experience as input, and in many cases they can be applied to simulated experience just as well as to real experience.



Random-sample one-step tabular Q-planning

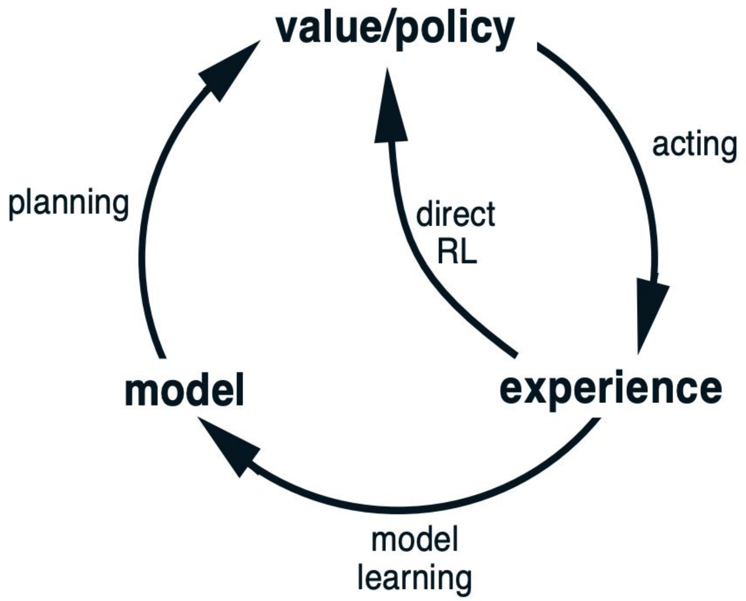
```
Loop forever:
    1. Select a state, $S \in \mathcal{S}$, and an action, $A \in \mathcal{A}(S)$, at random
    2. Send $S, A$ to a sample model, and obtain
           a sample next reward, $R$, and a sample next state, $S'$
    3. Apply one-step tabular Q-learning to $S, A, R, S'$:
           $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
```



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Dyna: Integrated Planning, Acting, and Learning

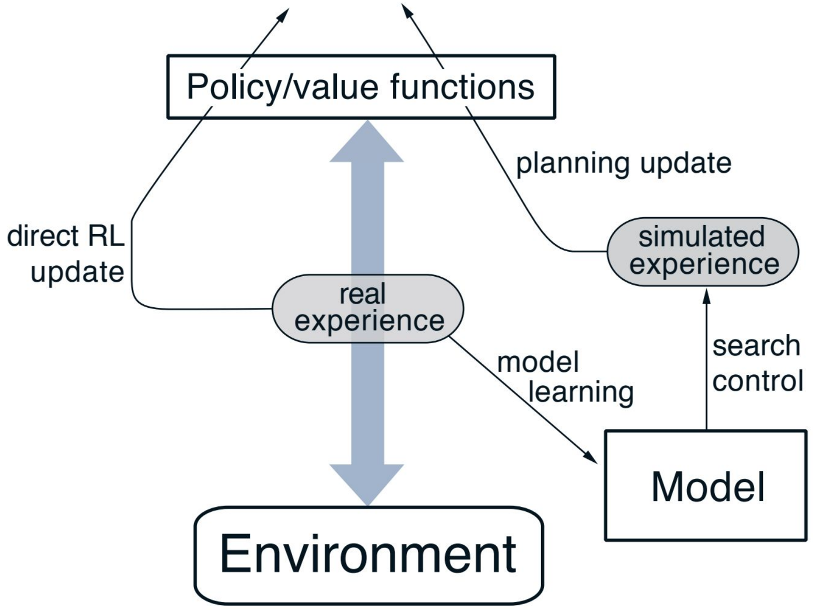
- Online planning, in small, incremental steps.



- Indirect methods often make fuller use of a limited amount of experience and thus achieve a better policy with fewer environmental interactions.
- Direct methods are much simpler and are not affected by biases in the design of the model.

Dyna-Q

- Dyna-Q includes all of the processes shown in the diagram—planning, acting, model-learning, and direct RL—all occurring continually (and *simultaneously*).
- Planning method: random-sample one-step tabular Q-planning.
- Direct RL method: one-step tabular Q-learning.
- Model-learning method: table-based and assumes the environment is deterministic.

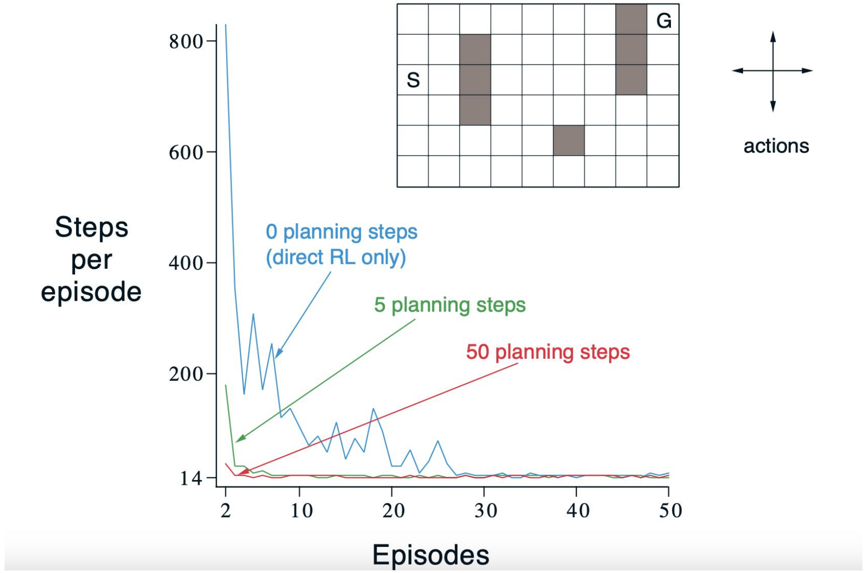


Dyna-Q

Tabular Dyna-Q

```
Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
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) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    (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 previously taken in $S$
            $R, S' \leftarrow Model(S,A)$
            $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
```

Example 8.1 – Dyna Maze



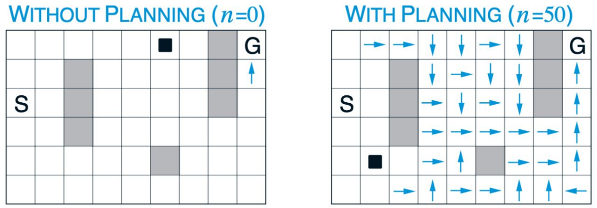
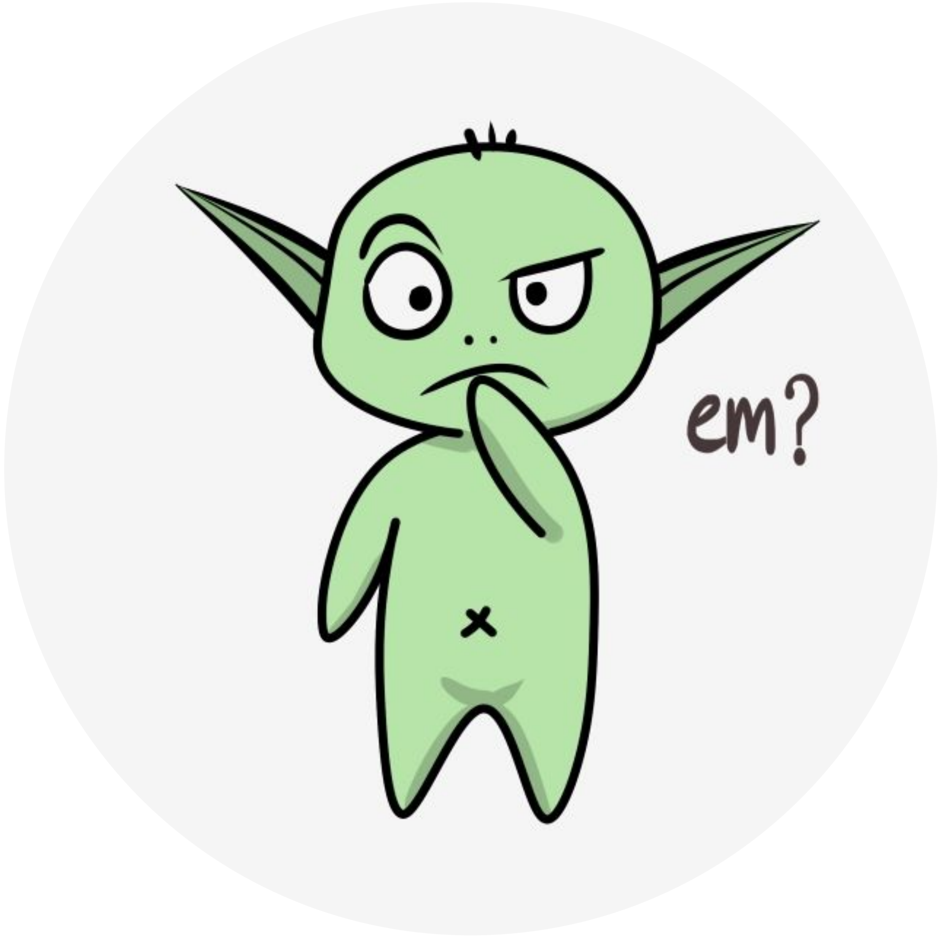


Figure 8.3: Policies found by planning and nonplanning Dyna-Q agents halfway through the second episode. The arrows indicate the greedy action in each state; if no arrow is shown for a state, then all of its action values were equal. The black square indicates the location of the agent. ■

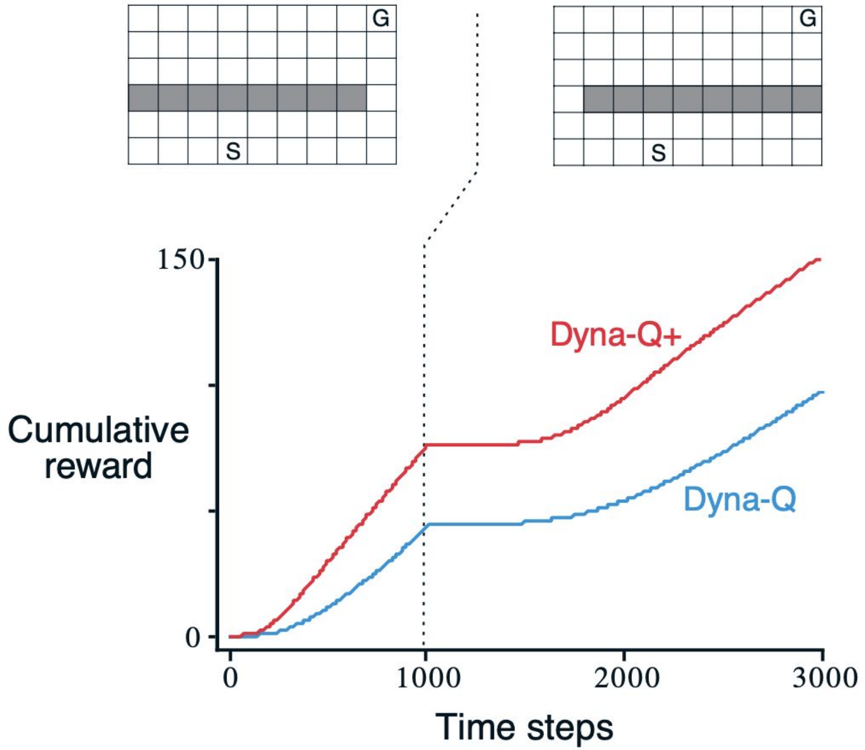


https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

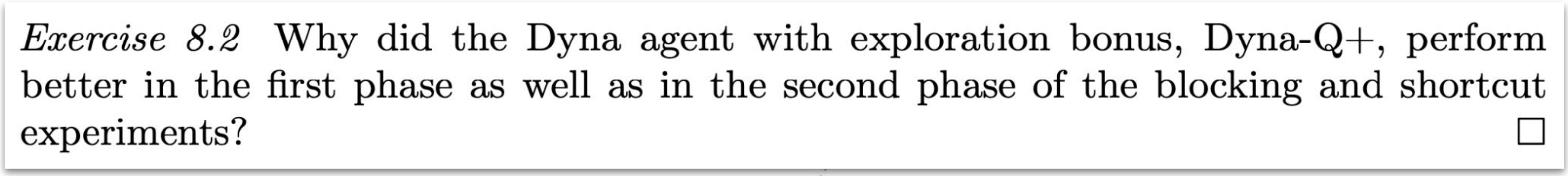
When the Model Is Wrong

- A model can be wrong for all sorts of reasons (e.g., stochastic environment, function approximation, non-stationarity in the environment).
- An incorrect model often leads to suboptimal policies.
- One needs to constantly explore to refine the learned model.
    - Exploration: take actions that improve the model.
    - Exploitation: behaving in the optimal way given the current model.
- Dyna-Q+: Provides “bonus rewards” for long-untried actions.
  Specifically, consider the reward $r + \kappa\sqrt{\tau}$, where τ is the number of time steps since that transition was tried for the last time.

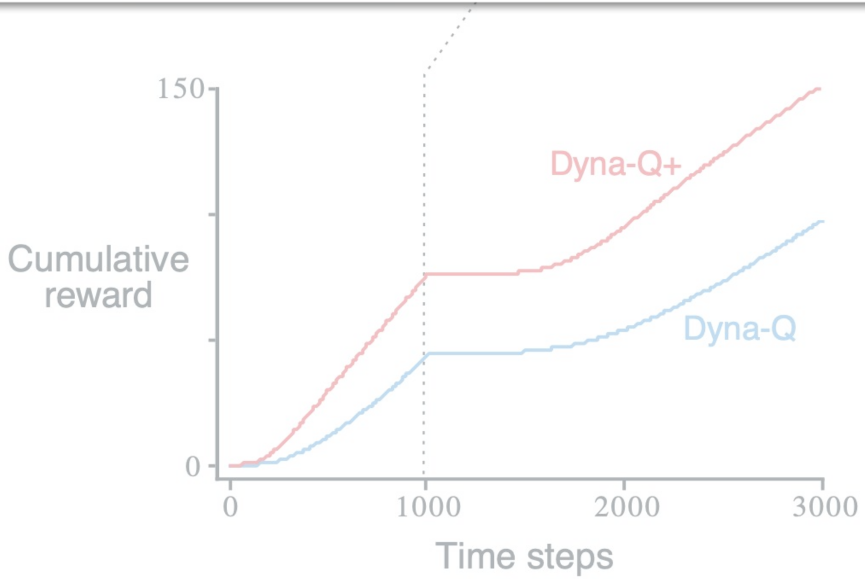
Dyna-Q+ Sometimes Works ¯_(ツ)_/¯



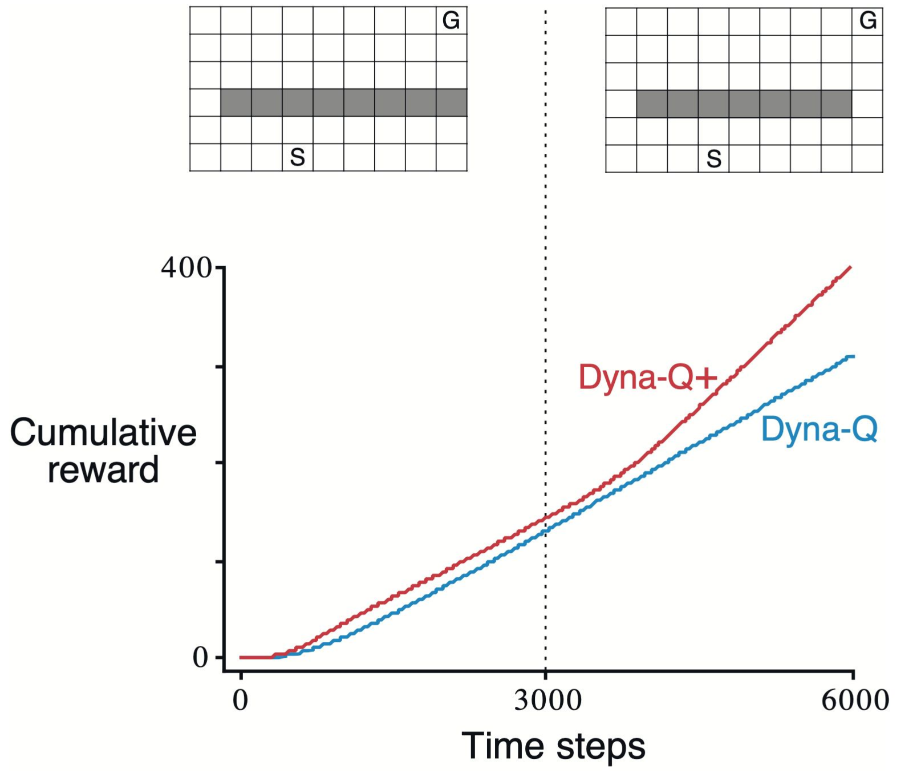
Dyna-Q+ Sometimes Works ¯_(ツ)_/¯



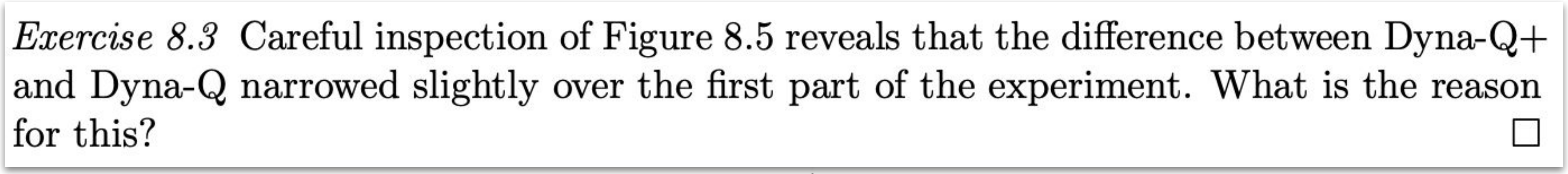
Exercise 8.2 Why did the Dyna agent with exploration bonus, Dyna-Q+, perform better in the first phase as well as in the second phase of the blocking and shortcut experiments? □



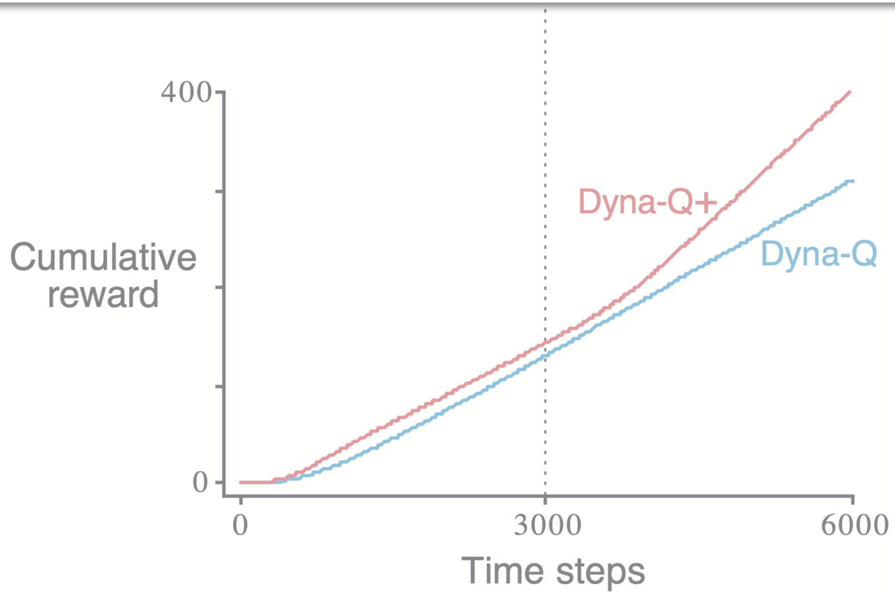
Sometimes it is Much Harder

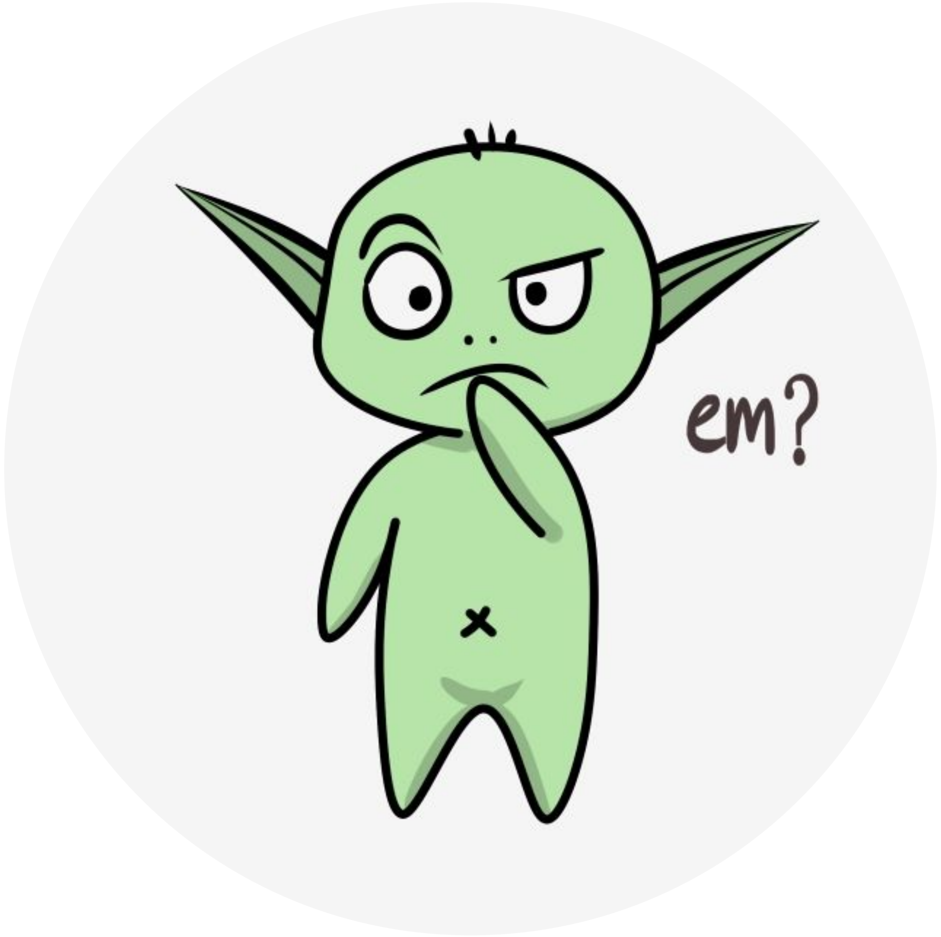


Sometimes it is Much Harder



Exercise 8.3 Careful inspection of Figure 8.5 reveals that the difference between Dyna-Q+ and Dyna-Q narrowed slightly over the first part of the experiment. What is the reason for this? □

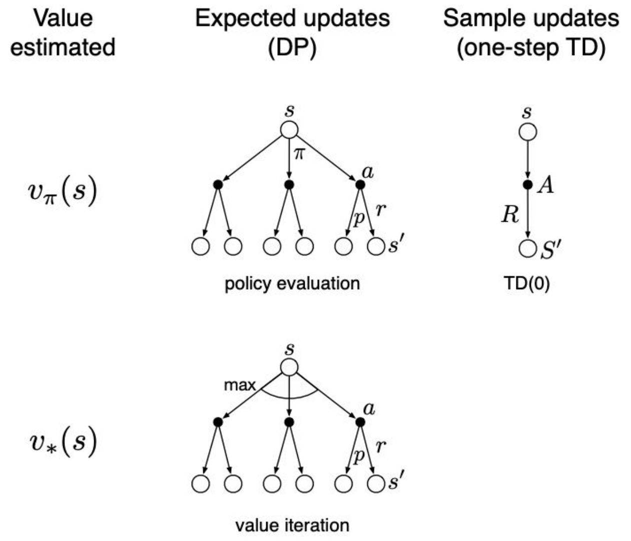


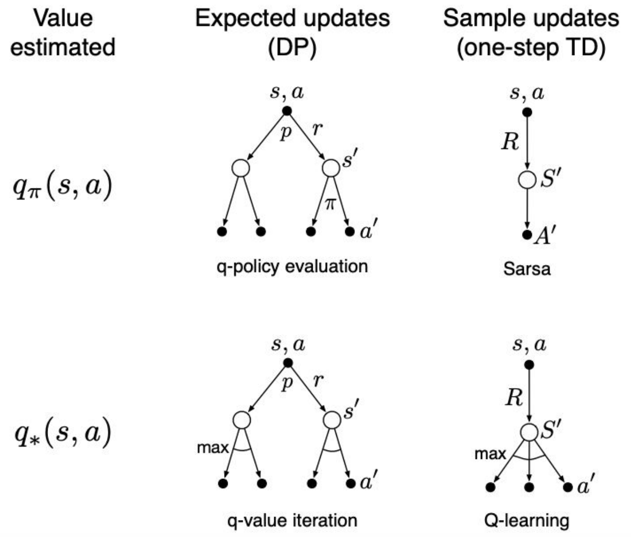


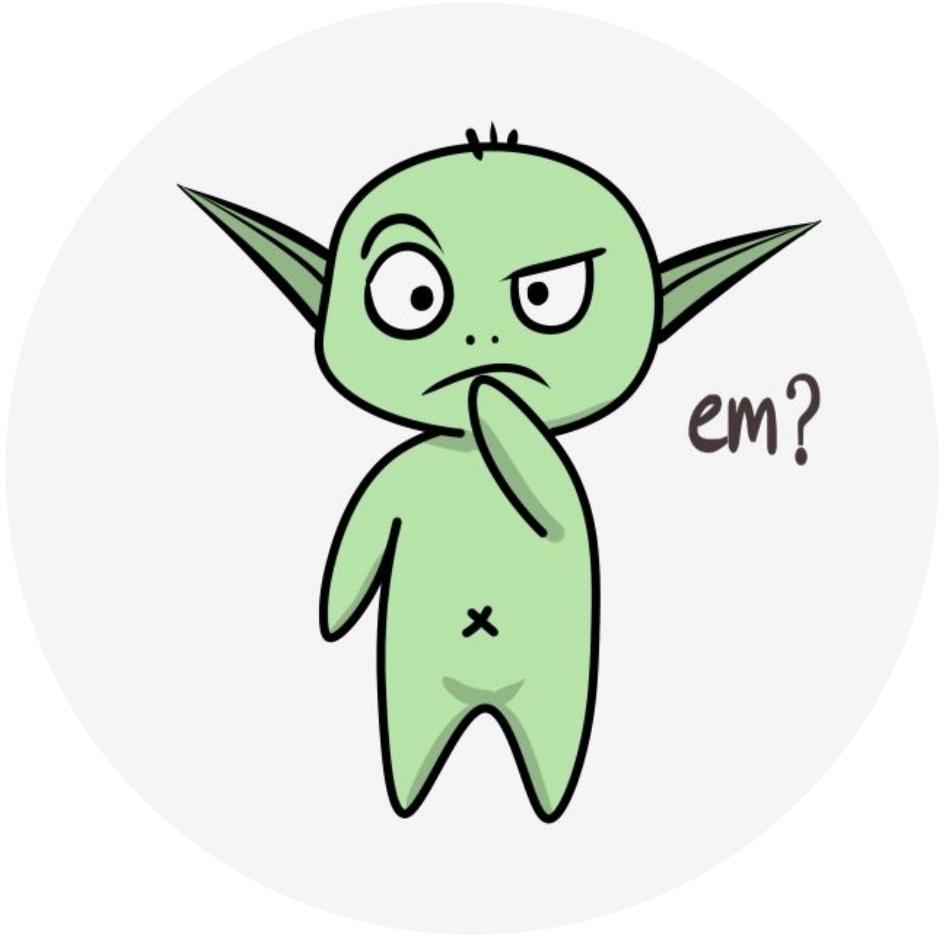
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Expected vs. Sample Updates

- There are three dimensions in the updates one can do:
  - Should we use state values or action values?
  - Should we estimate the value for the optimal policy or for an arbitrary given policy?
  - Should we use expected or sample updates?







https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

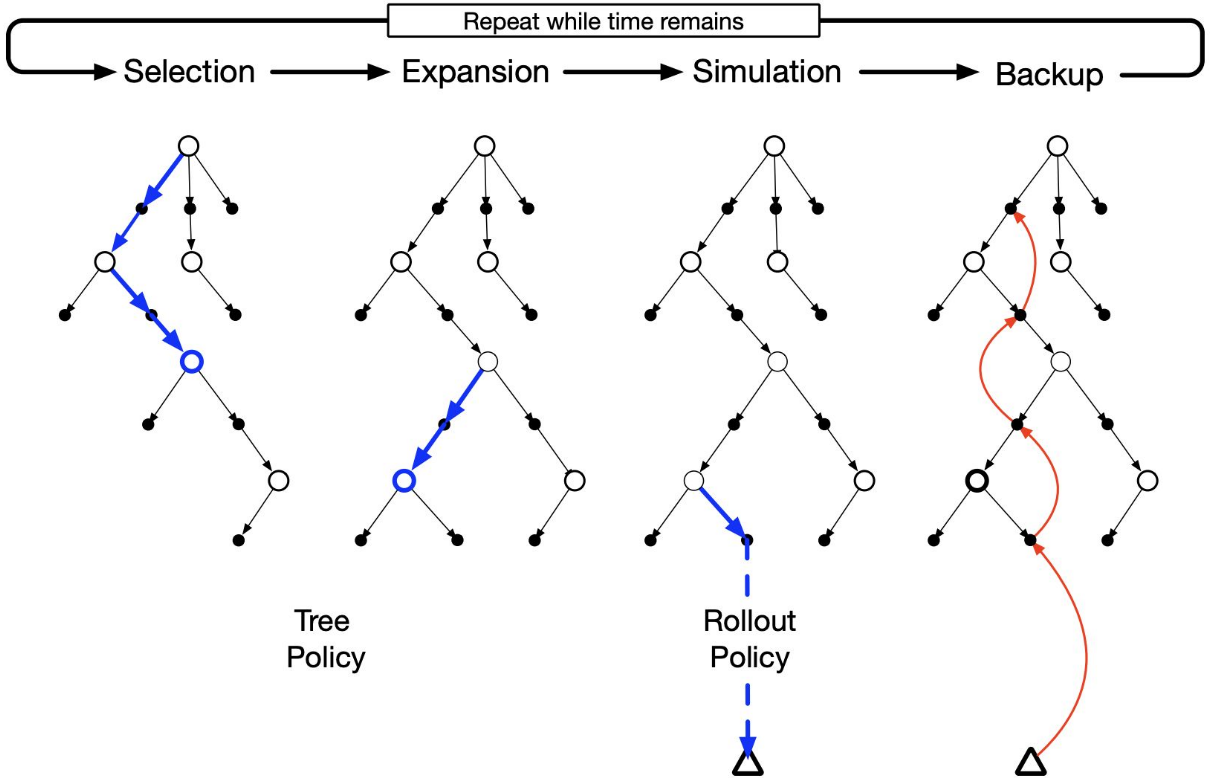
Rollout Algorithms

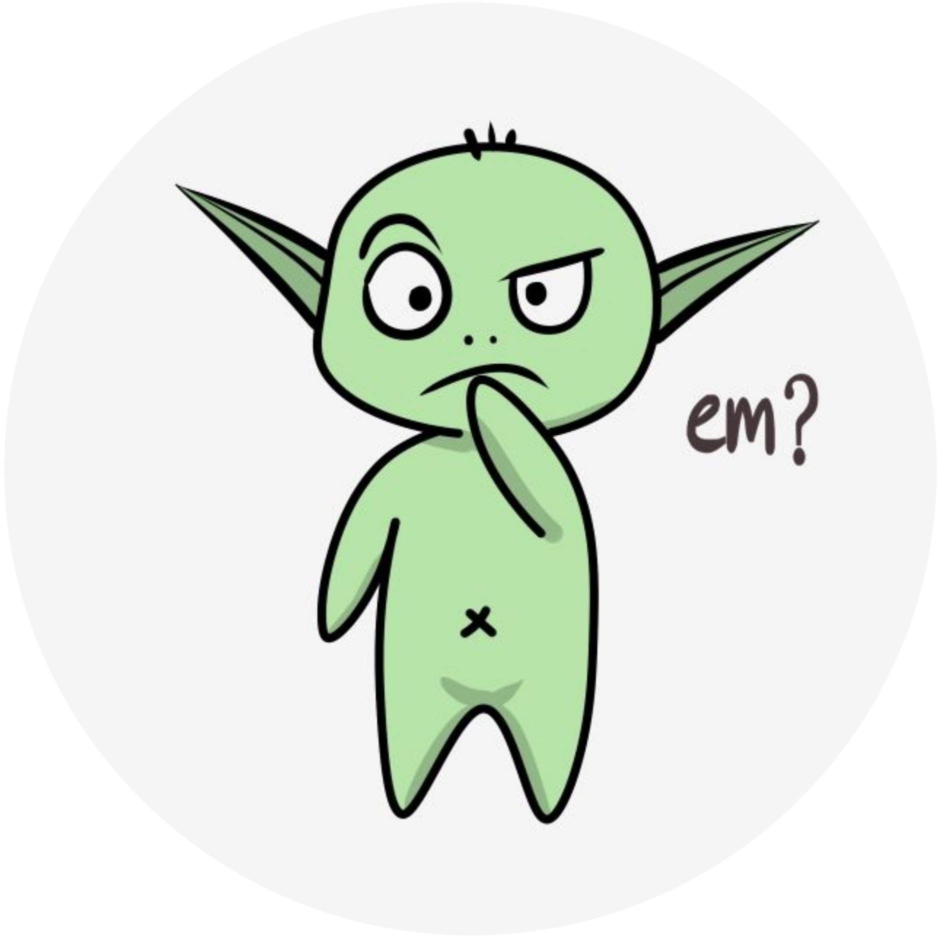
- Rollout algorithms are decision-time planning algorithms based on MC control applied to simulated trajectories that all begin at the current environment state.
    - They estimate action values for a given policy by averaging the returns of many simulated trajectories that start with each possible action and then follow the given policy.
- Unlike the Monte Carlo control algorithms previously described, the goal of a rollout algorithm is not to estimate a complete optimal action-value function, q_*, or a complete action-value function, q_π, for a given policy π.
    - They produce Monte Carlo estimates of action values only for each current state and for a given policy usually called the rollout policy.
- They are not learning algorithms *per se*, but they do leverage the RL toolkit.

Monte Carlo Tree Search (MCTS)

- MCTS is a great example of a rollout, decision-time planning algorithm.
  - But enhanced by the addition of a means for accumulating value estimates obtained from the MC simulations in order to successively direct simulations toward more highly-rewarding trajectories.
- The core idea of MCTS is to successively focus multiple simulations starting at the current state by extending the initial portions of trajectories that have received high evaluations from earlier simulations.
  - Monte Carlo value estimates are maintained only for the subset of state–action pairs that are most likely to be reached in a few steps, which form a tree rooted at the current state.

Monte Carlo Tree Search (MCTS)



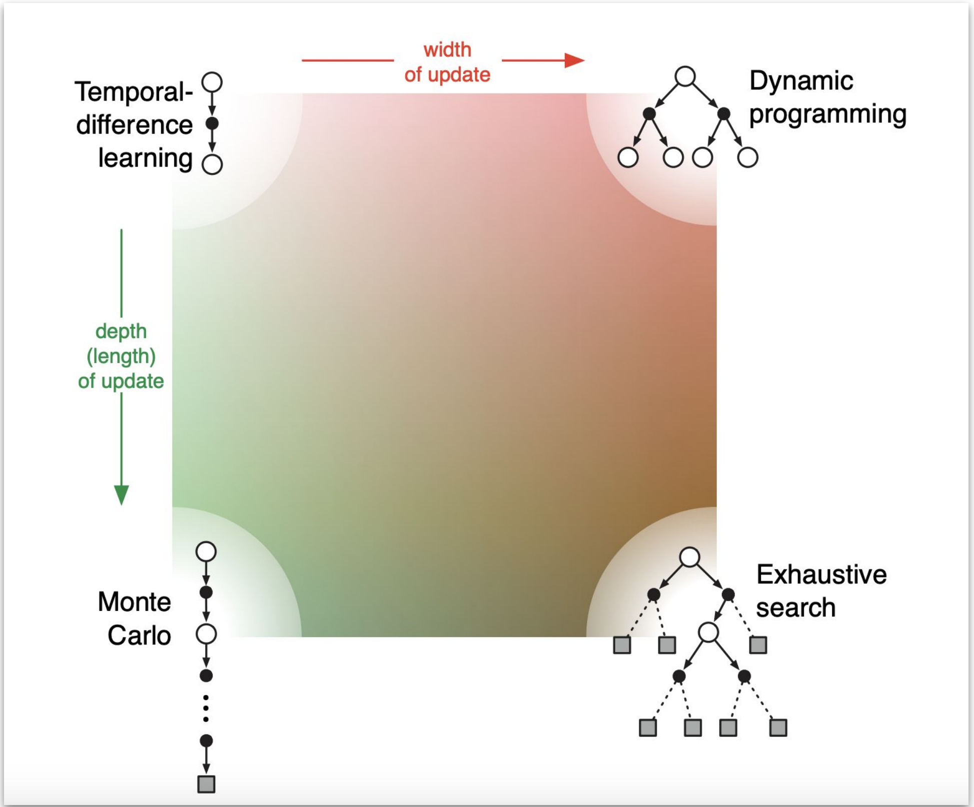


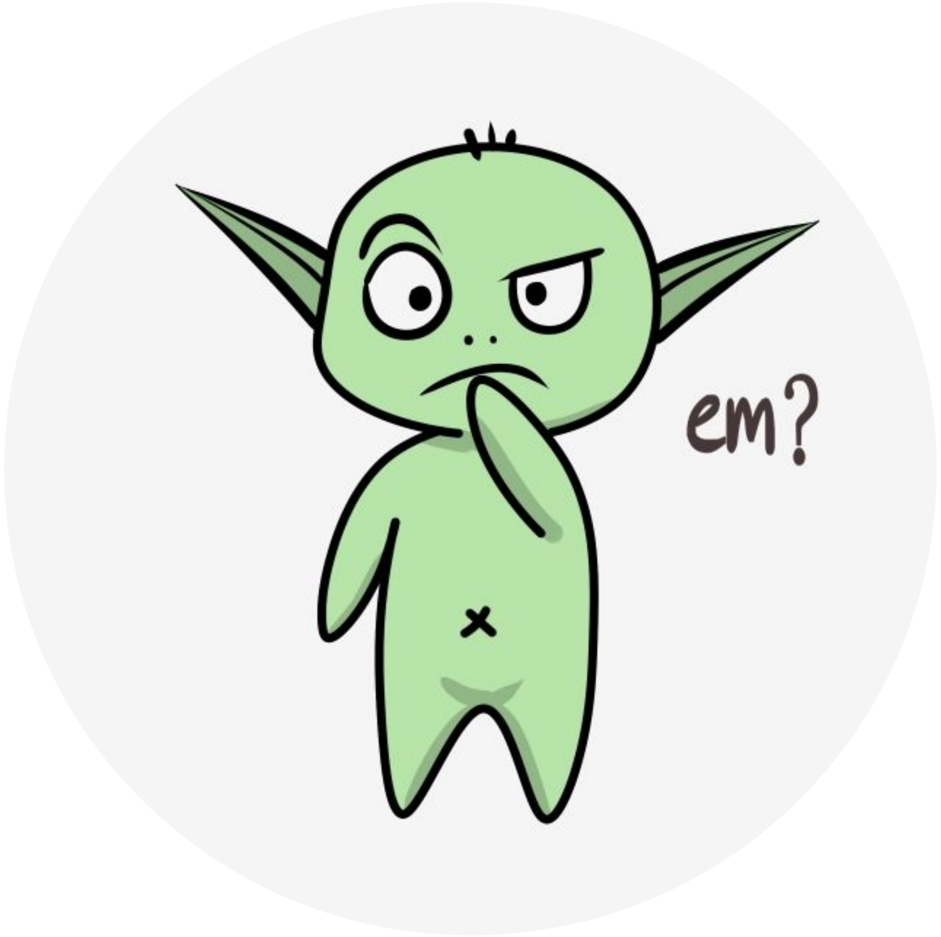
https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Wrapping Up

- We have finished Part I of the textbook, Tabular Solution Methods.
- Reinforcement learning can be seen as being more than a collection of individual methods, but a coherent set of ideas cutting across methods.
  - They all seek to estimate value functions.
  - They all operate by backing up values along actual or possible state trajectories.
  - They all follow the general strategy of generalized policy iteration (GPI).

Wrapping Up


width
of update
Temporal-
difference
learning
Dynamic
programming
depth
(length)
of update
Monte
Carlo
Exhaustive
search



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Exercise

Consider an MDP with three states $\mathscr{S} = \{1, 2, 3\}$, where each state has two possible actions $\mathscr{A} = \{1, 2\}$ and a discount rate $\gamma = 0.5$. Suppose estimates of Q(S, A) are initialized to 0 and you observed the following episode according to an unknown behaviour policy where S_3 is the terminal state.

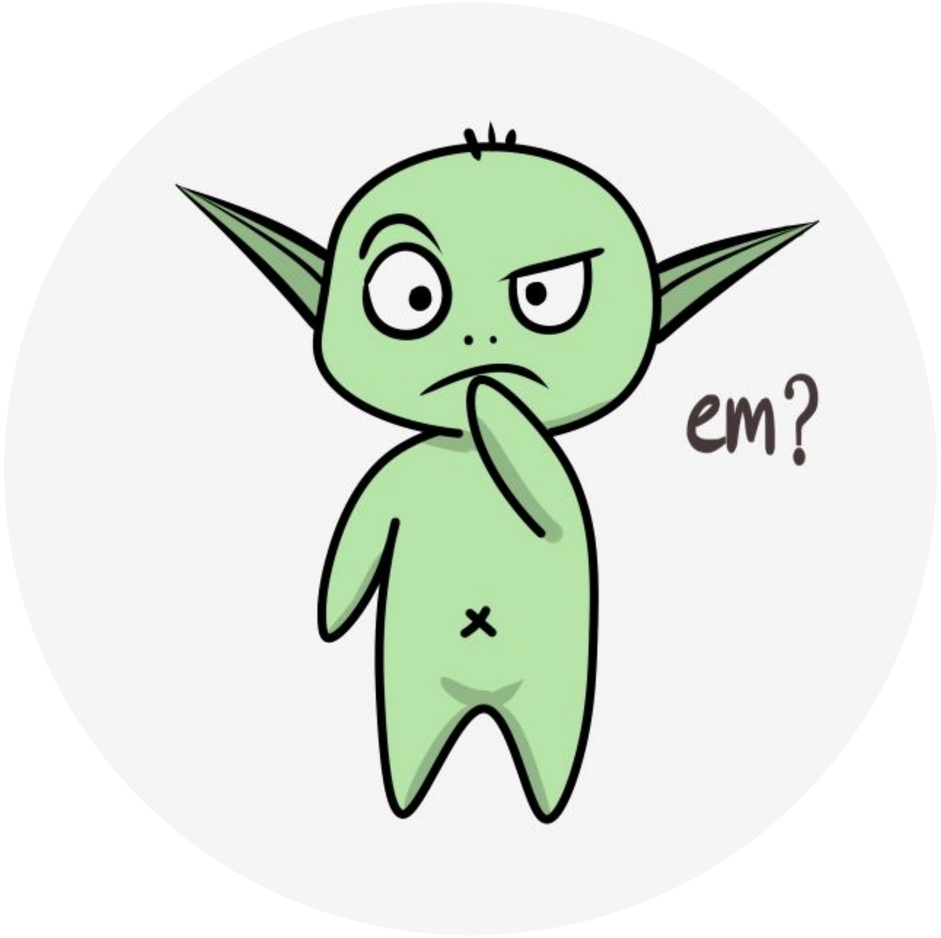
$$S_0 = 1, A_0 = 1, R_1 = -7, S_1 = 3, A_1 = 2, R_2 = 5, S_2 = 1, A_2 = 1, R_3 = 10$$

(a) Suppose you used Q-learning with the above trajectory to estimate Q(S, A), what are your new estimates for Q(S = 1, A = 1) using $\alpha = 0.1$?

(b) What is one possible model for this environment? Is the model stochastic or deterministic?

(c) Suppose in the planning loop, after search control, we would like to update Q(S = 1, A = 1) with Q-planning. What are the possible outputs of Model(S = 1, A = 1)?

(d) If your model outputs $R = R_3$ and $S' = S_3$, what is Q(S = 1, A = 1) after one Q-planning update? Use the estimates of Q(S, A) from before.



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht