Chapter 8 - Planning and Learning w/ Tabular Methods:

Jan 5, 2022

The goal is to integrate model-based and model-free methods. **Model-based methods** require a model of the environment and rely on **planning**. **Model-free methods** don't use models and rely on **learning**.

8.1 - Models and Planning:

Models:

A **model** of the environment is anything that an agent can use to predict how the environment will respond to its actions.

↳ **Distribution models** produce a description of all possibilities and their probabilities.
  ↳ Can be used to compute the exact expected outcome

↳ **Sample models** produce <u>just one</u> of the possibilities, sampled according to the probabilities.
  ↳ Can be less computationally expensive.

Planning:

Planning refers to any computational process that takes a model as input and produces/improves a policy for interacting w/ the modelled environment.

model $\xrightarrow{\text{planning}}$ policy

↳ State-space planning searches through the state space for an optimal policy or optimal path to a goal.

↳ Actions cause transitions, compute value functions, basically the approaches learned before.

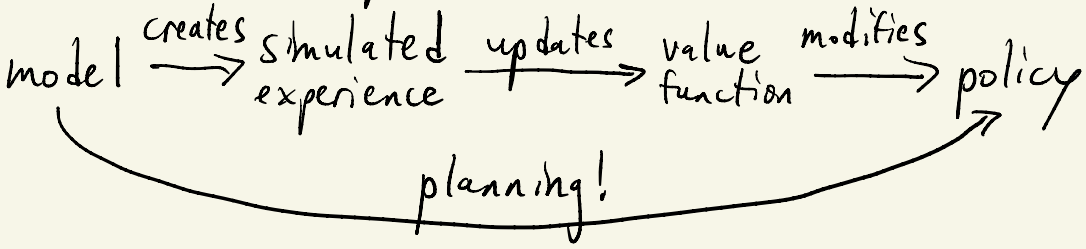
↳ Plan-space planning instead searches through the space of plans.

⛔ Difficult to apply efficiently to RL, ignore this.

State-Space Planning Methods!

The unified view presented in this chapter is that all state-space planning methods share a common structure.

1. They all involve computing value functions as a key intermediate step toward improving the policy.
2. They all compute value functions by updates applied to simulated experience.



Random-Sample One-Step Tabular Q-Planning!

Assumes we have a sample model of the transition dynamics and a strategy for sampling relevent state-action pairs. ✱Note! This only needs simulated experience.

1. Sampling

2. Q-learning update

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

3. Greedy policy improvement

$$\pi(s) = \operatorname{argmax}_a Q(s,a)$$

Pseudocode:

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

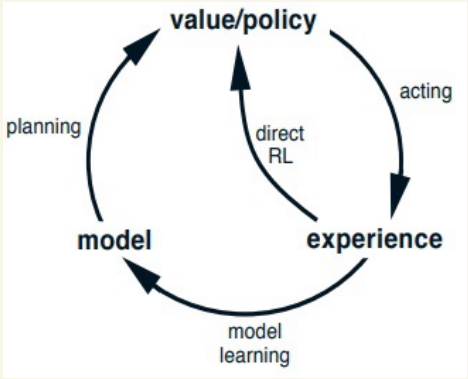
※ This is a simple example of a planning method.

8.2 - Dyna: Integrated Planning, Acting, and Learning:

Within a planning agent, there are at least two roles for real experience.

↳ Model-learning is used to improve the model to make it more accurately match the real environment.

↳ Direct RL improves the value function and policy using previously-discussed methods.



Dyna-Q: * Combines Q-learning and Q-planning

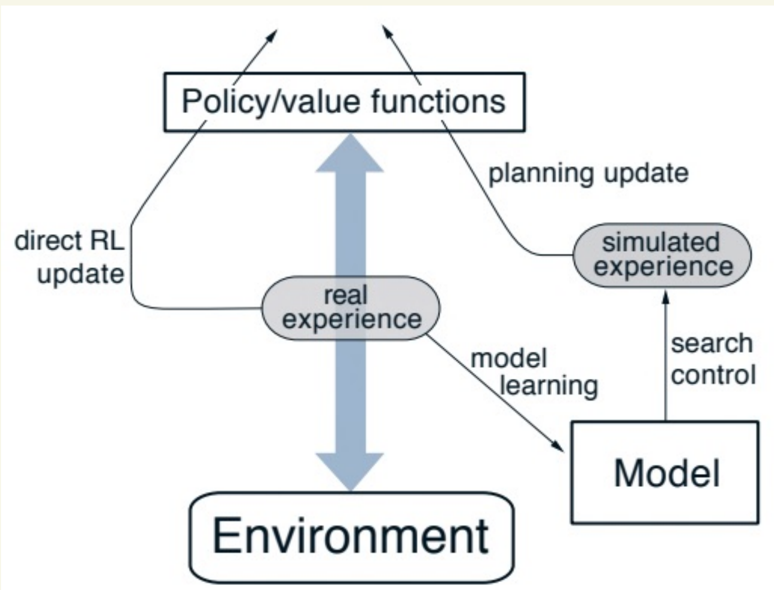
The Dyna-Q algorithm involves all of the processes previously shown (planning, acting, model-learning, direct RL) all occurring continually.

↳ Planning method is random-sample one-step tabular Q-learning.

↳ Direct RL method is one-step tabular Q-learning.

↳ Model-learning is table-based and assumes environment is deterministic; deterministic transitions.

General Dyna Architecture:



* Acting, model-learning, and direct-RL require little computation → planning is most computationally expensive.

Tabular Dyna-Q Pseudocode!

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

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

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

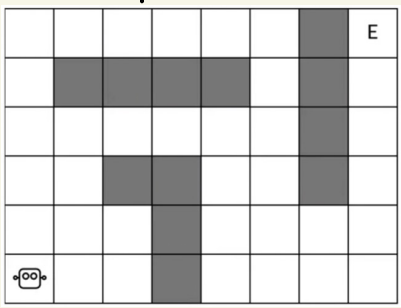
Q-learning (direct-RL)

Model learning

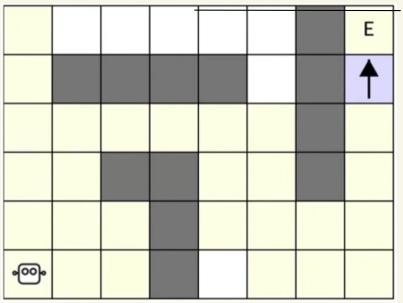
Many planning updates per transition

Q-planning

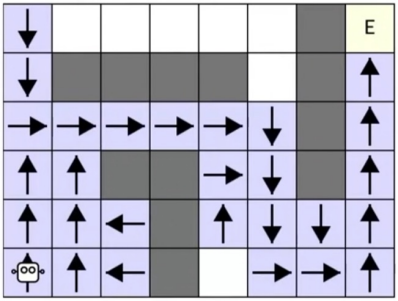
Example!



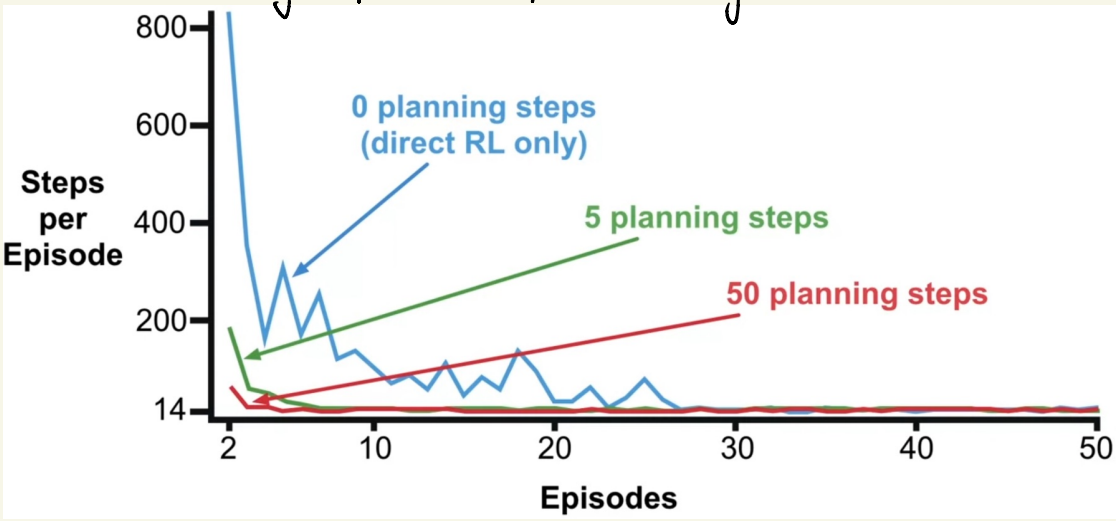
One direct-RL update using Q-learning

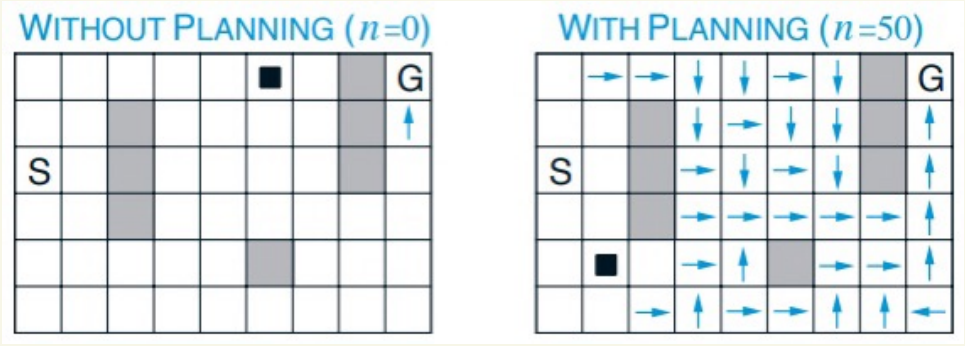


N planning updates using Q-planning



More Planning Speeds Up Learning!

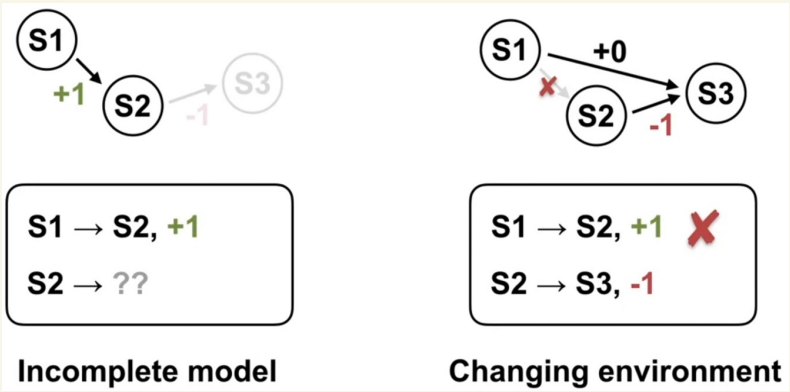




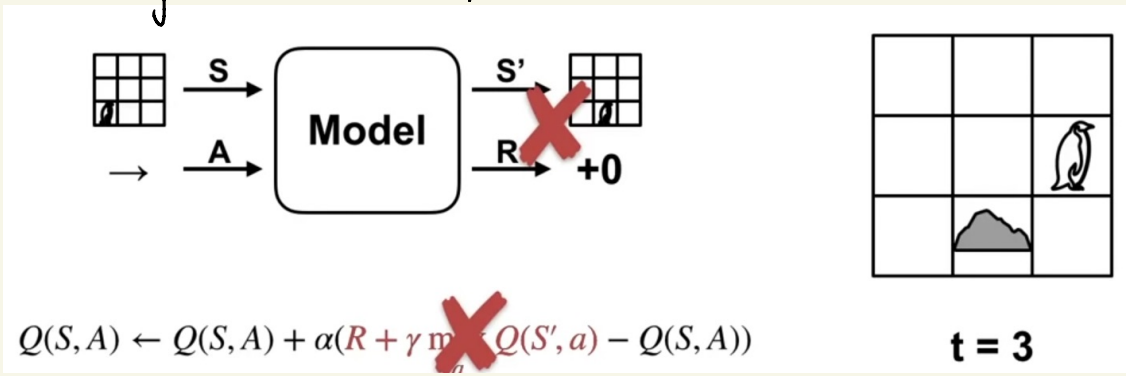
8.3 - When the Model is Wrong!

Models are inaccurate when the transitions they store are different from transitions that happen in the environment.

↳ Happens when model is incomplete (hasn't tried most actions) or if the environment changes.



Planning w/ an Incomplete Model:



↳ Environment changes, model becomes outdated. Planning w/ that model will make agent's policy worse w.r.t. the environment.

Dyna w/ an Incomplete Model!

Dyna-Q can plan w/ an incomplete model by only sampling state-action pairs that have been previously visited.

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

For Changing Environments!

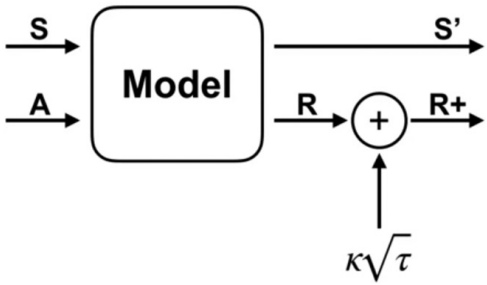
To encourage the agent to visit previously-explored states, add a bonus reward.

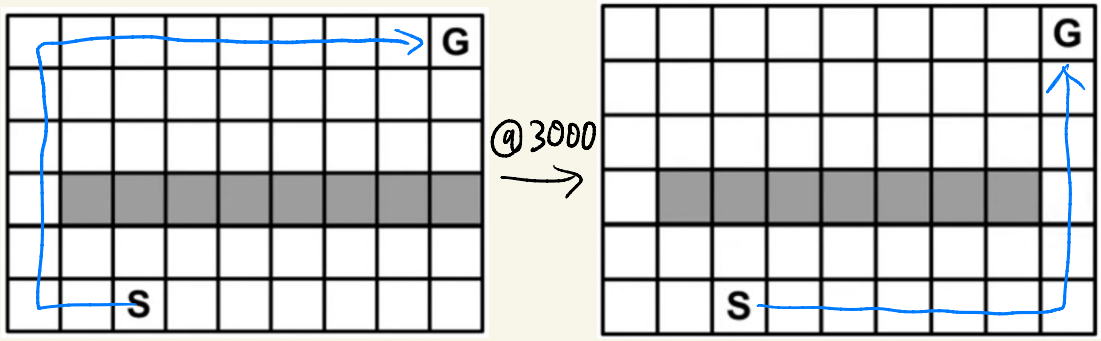
$$\text{new reward} = r + K\sqrt{\tau}$$ ⁂ Exploration bonus

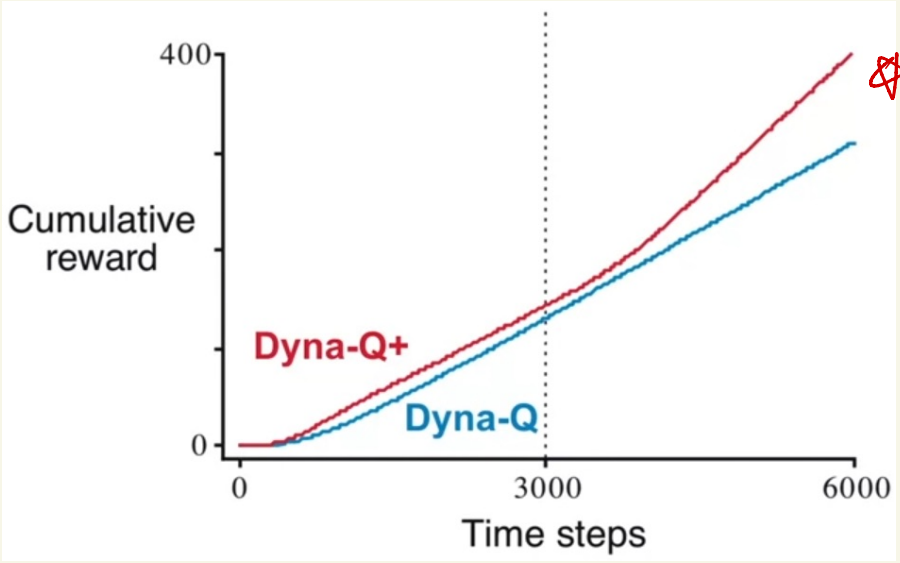
r: Actual reward
τ: Time steps since transition was last tried
K: Small constant

Dyna-Q+ Algorithm!

Adding this reward bonus to Dyna-Q's planning updates results in the Dyna-Q+ algorithm







*Dyna-Q+ finds the shortcut!

↳ Both can find the shortcut, but Dyna-Q+ is more likely due to modified reward.