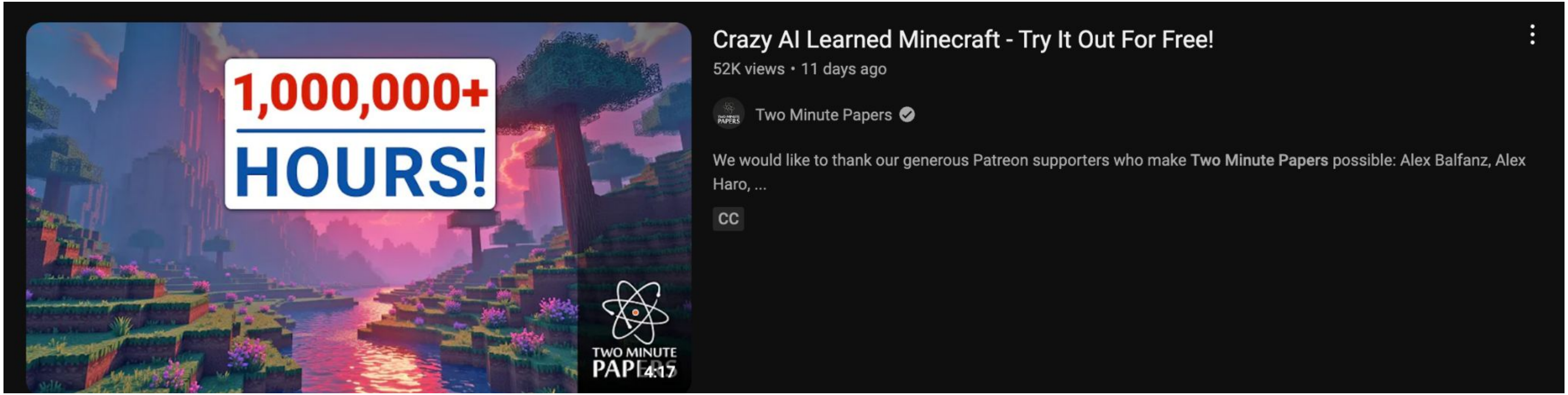
Week 3: TD Control, SARSA, Q-Learning

TD-Control, SARSA, Q-Learning

Go to our Linktree to follow our **socials**

AI News!

- AI Generated Minecraft!!!!!



Yes, you can play now for free

Brief Recap

Bellman Equations

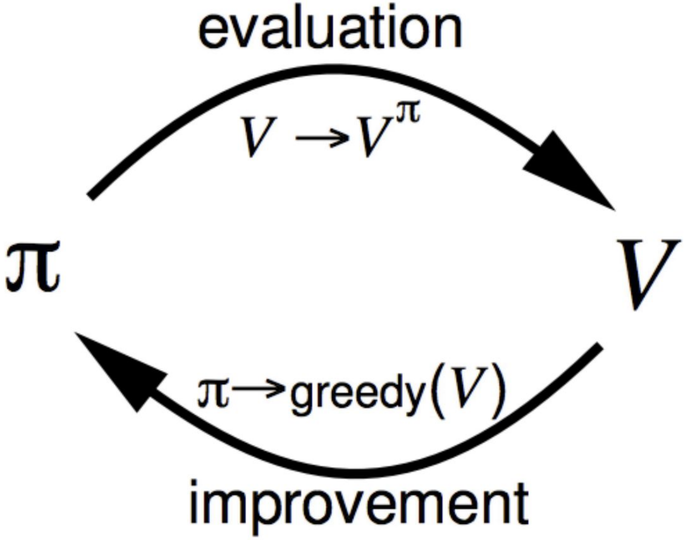
$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_\pi(s')]$$

$$Q_\pi(s, a) = \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s') Q_\pi(s', a')]$$

$$V_*(s) = \max_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_*(s')]$$

$$Q_*(s, a) = \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma \max_{a'} Q_*(s', a')]$$

Policy Iteration



1. Initialize a random policy
2. Policy Evaluation: find the value function for the current policy
3. Policy Improvement: act greedily w.r.t. the found value function
4. Repeat until policy is optimal

$$\pi_1 \rightarrow V_{\pi_1} \rightarrow \pi_2 \rightarrow V_{\pi_2} \rightarrow \pi_3 \rightarrow V_{\pi_3} \rightarrow \ldots \rightarrow \pi_*$$

Optimal Policies from Optimal Values

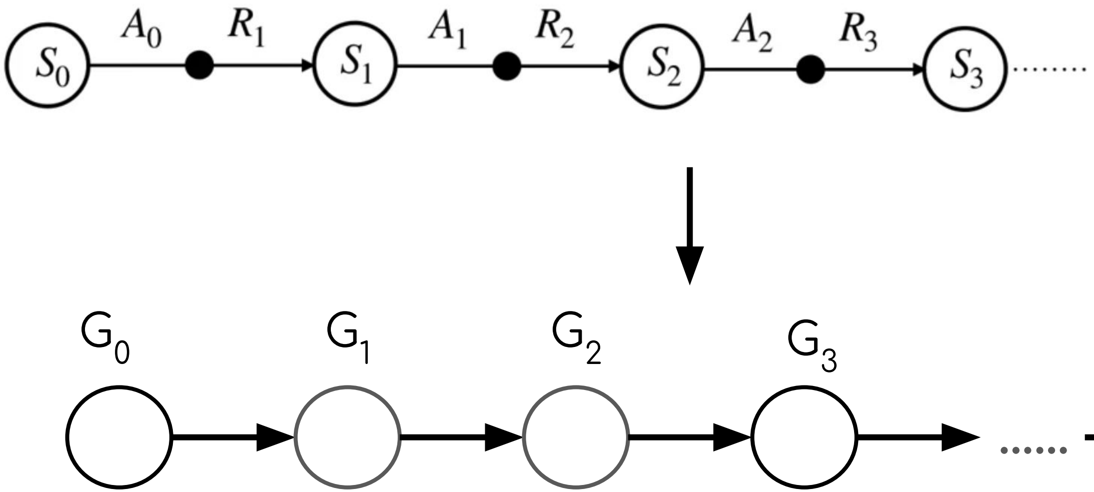
$$\pi_*(s) = \arg\max_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_*(s')]$$

$$\pi_*(s) = \arg\max_a Q_*(s, a)$$

Which one (V, Q) do you want to use for control?

Monte Carlo Methods

Sample many trajectories of experience, obtain the **return** for each state, and take the average of results



Trajectory of experience

Monte Carlo Methods

$$V(S_t) = V(S_t) + \alpha[G_t - V(S_t)]$$

Monte Carlo is a MODEL-FREE algorithm, meaning that it does not need a model of the environment's transition probabilities

State Value vs. State Action Value

Consider this....

Assume this is a completely new environment, and you are given this optimal value function:

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Actions: Up, Down, Left, Right.
Rewards are -1 per timestep

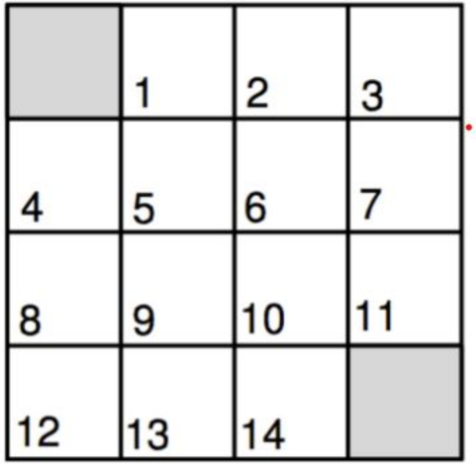
You are not given anything else about the environment, and you want to find the optimal policy given this value function

What is the best action in the highlighted state?

Previously....

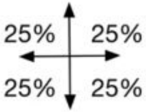
Recall our discussion of policy iteration from last week

Gridworld Example



- Terminal states in gray squares
- Actions: North, South, East, West, and agent cannot move off the grid
- Reward: -1 per timestep
- Episodic task (γ = 1)

*Implied: agent moves to adjacent square



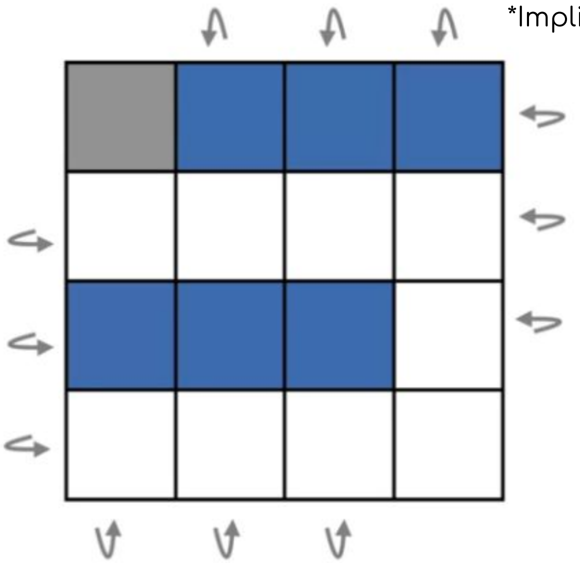
Policy

Previously....

This was the other example that was used

Maze

*Implied: agent moves to adjacent square



- Singular terminal state at gray square
- Actions: North, South, East, West, and agent cannot move off the grid
- Reward: -1 per timestep, but if agent transitions into **blue state, -10 reward**
- Episodic task with discount factor $\gamma = 1$
- Start with <u>random policy</u>

Consider this....

After considering the previous slides, what do you think?:

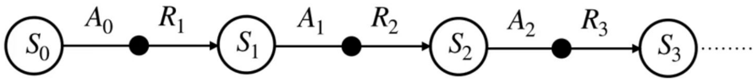
0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

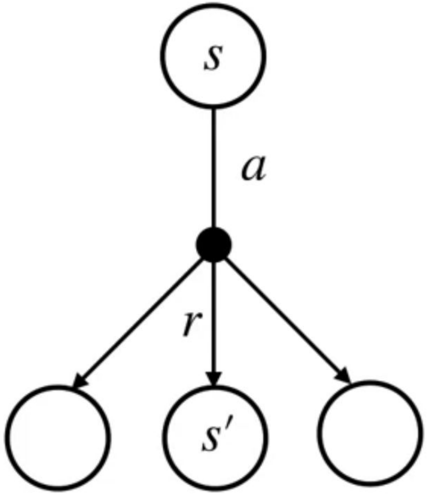
$$\pi_*(s) = \arg\max_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_*(s')]$$

Do you know all the quantities in the above equation?

Reminder: MDP Transition Dynamics

Recall...





$$p(s', r | s, a)$$

Limitation of the State Value Function

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

How do you know that going 'left' takes to the adjacent state? What if it goes to the bottom right corner? Out of the screen? Both?

Without the transition dynamics, we have no idea what next states actions lead to

$$\cancel{\pi_*(s) = \arg\max_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_*(s')]}$$

Why Learn the State-Action Value Function?

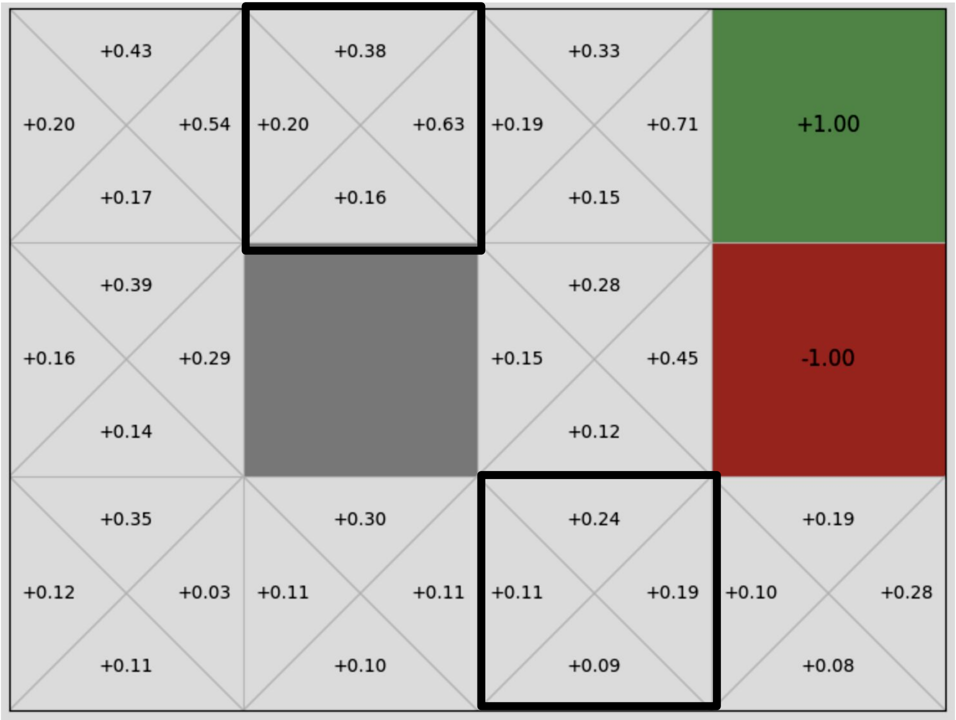
The equation for an optimal policy for the Q-function does not require the dynamics model

$$\pi_*(s) = \arg\max_a Q_*(s, a)$$

The values at all the possible next states are already 'baked' into the Q-function.

This is the preferred way to obtaining optimal policies in value-based RL methods

Example

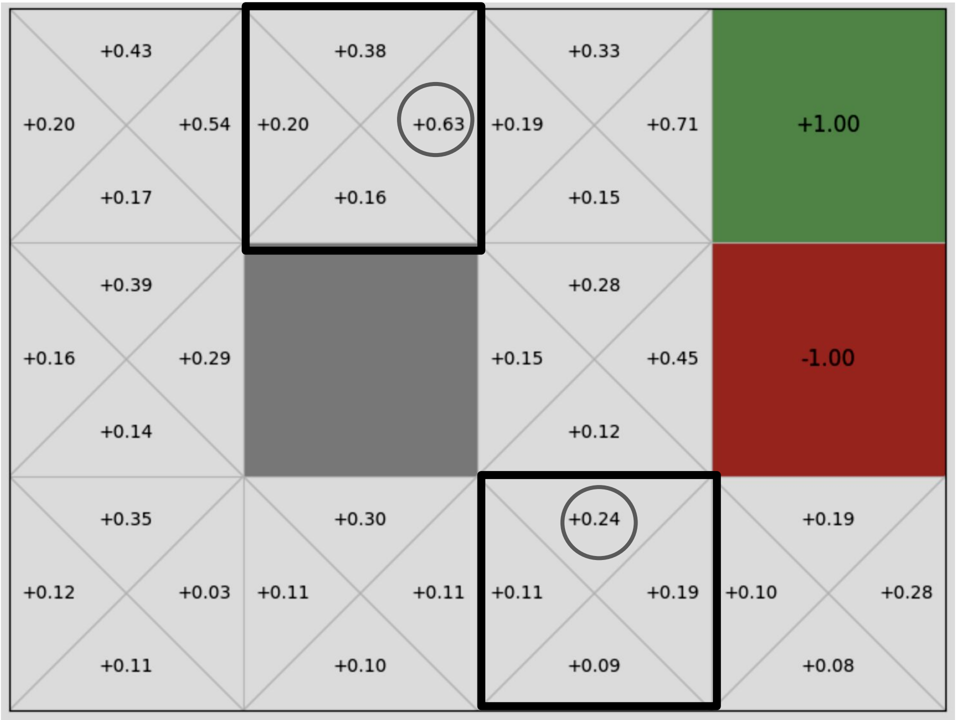


Given the Q-function to the left

<u>Actions</u>: Up, Down, Left, Right.

You are not given anything else about the environment. What are the best actions in the highlighted states?

Example



$$\pi'(s) = \arg\max_a Q(s, a)$$

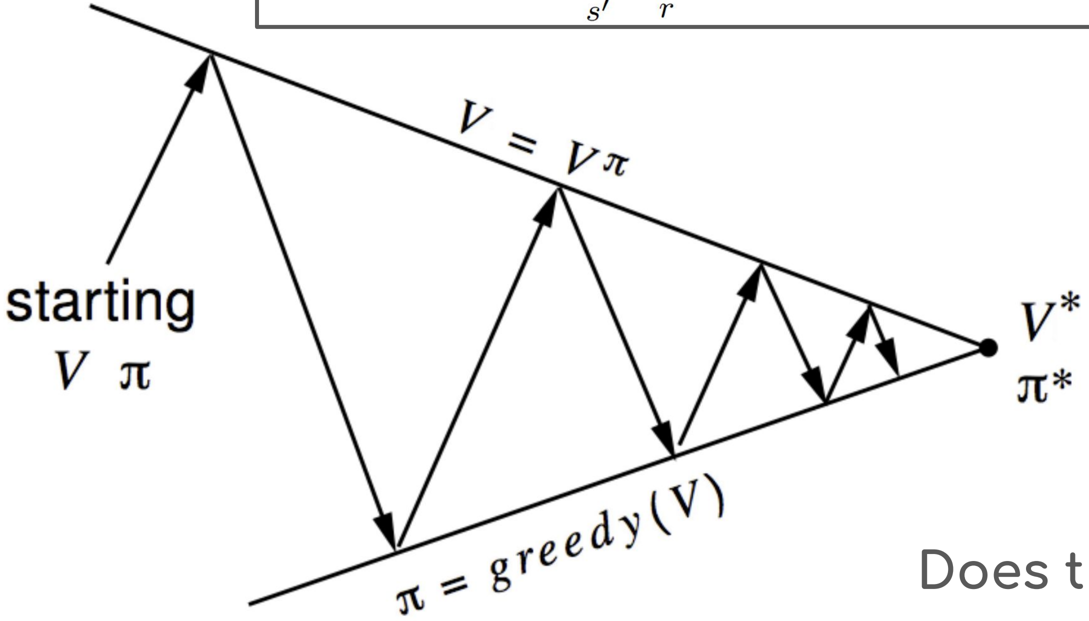
Just pick the largest state-action value!

No need for dynamics model, all future transitions “encoded” in Q value already

Monte Carlo Control

Monte Carlo PI with State Values?

$$\pi'(s) = \arg\max_a \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_\pi(s')]$$

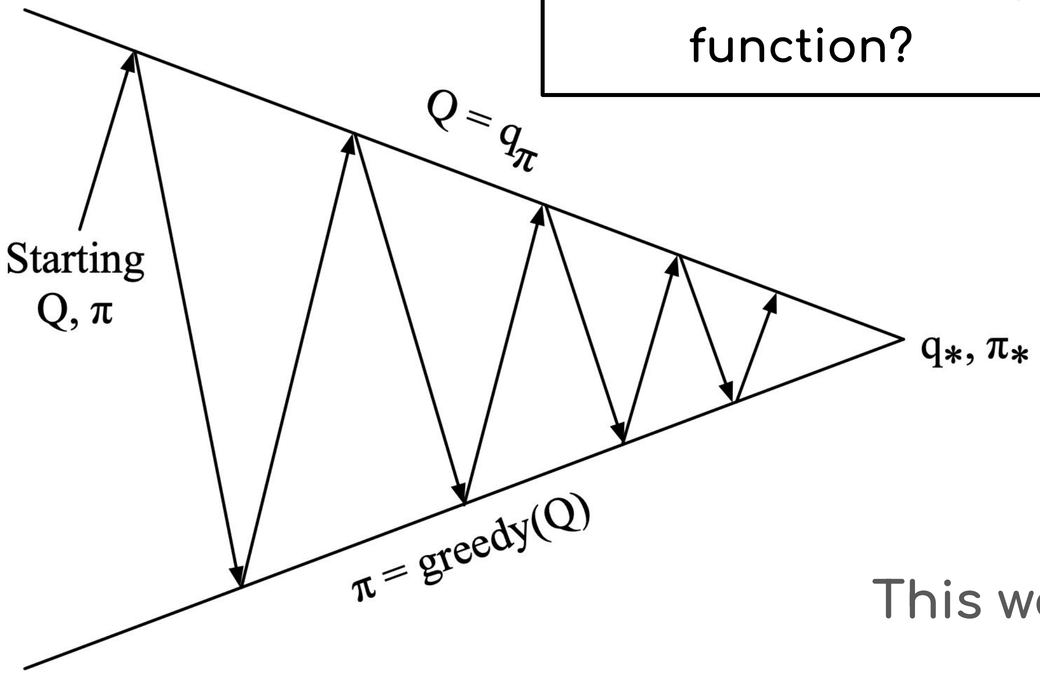


Policy Evaluation:
Monte Carlo

~~**Policy Improvement:**~~
~~Greedy Selection (V)~~

Does this work?

Monte Carlo Control



How to do PE with Q function?

Policy Evaluation:
Monte Carlo

Policy Improvement:
Greedy Selection (Q)

This works!

Monte Carlo Update Rule for Q-Function

Recall: Monte Carlo Update with value function

$$V(S_t) = V(S_t) + \alpha[G_t - V(S_t)]$$

Recall: Definition of Q-function

$$Q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a]$$

The Q-function Monte Carlo update is

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

Temporal Difference (TD) Learning

Bootstrapping Returns

Recall the following expression for the return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

$$= R_{t+1} + \gamma G_{t+1}$$

bootstrapping

Therefore, we have that:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

$$= R_{t+1} + \gamma V_\pi(S_{t+1})$$

bootstrapping

Temporal Difference Update

Recall the following expression for the Monte Carlo update:

$$V(S_t) = V(S_t) + \alpha[G_t - V(S_t)]$$

Replace G_t with the bootstrapped value function expression to obtain the temporal difference (TD) update:

$$\boxed{V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]}$$

where the TD error is $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

Contrast with Dynamic Programming

Recall that in DP, we looked at ALL the possible next states, hence requiring a model:

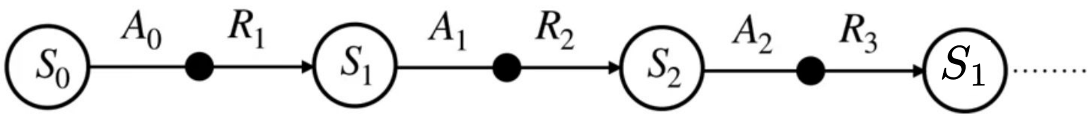
$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma V_\pi(s')]$$

However, in TD, we only need the next state:

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Hence, we do not need a model!

1-Step TD



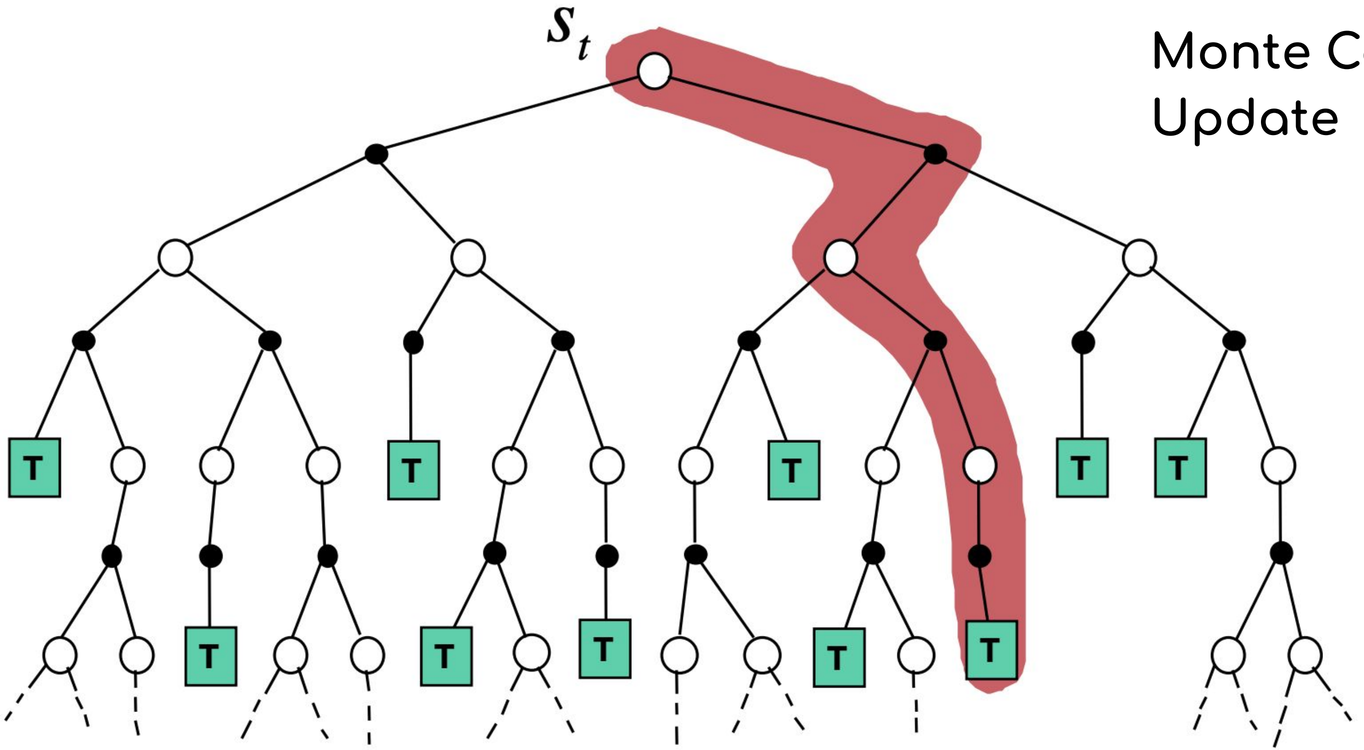
Trajectory of experience

Start in a state S_0, take an action A_0, receive reward and land in next state S_1. When reward and next state are received, update.

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

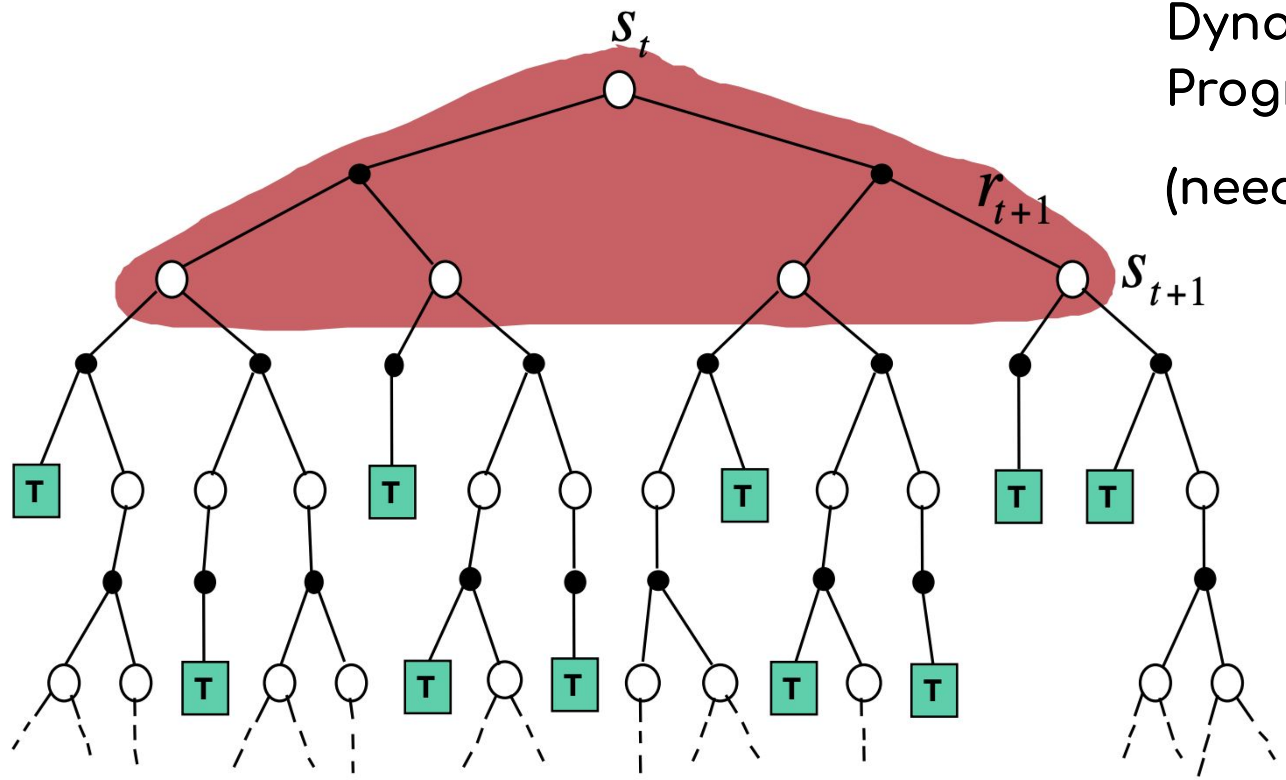
Now, in state S_1, take action A_1, receive reward R_2 and next state S_2, update, etc., etc. until episode ends (terminal state)

Comparing Updates



Monte Carlo
Update

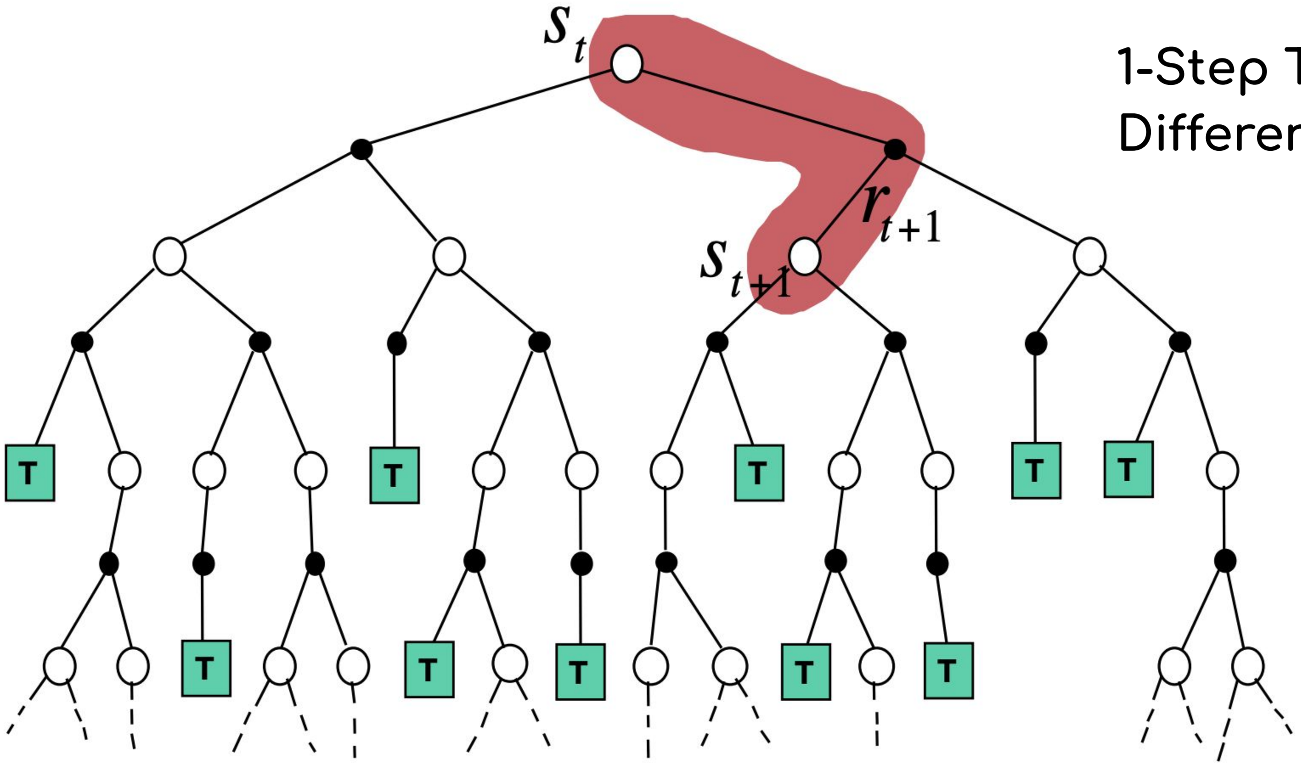
Comparing Updates



Dynamic Programming

(needs dynamics)

Comparing Updates



1-Step Temporal Difference

TD(0) Algorithm

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

So Which Should be Preferred?

Compare the return (Monte Carlo) and the TD target (TD):

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \quad \text{(return)}$$

$$R_{t+1} + \gamma V(S_{t+1}) \quad \text{(TD target)}$$

Which is a better estimate of the value $V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$?

Which takes longer to compute? Which has lower randomness?

Which do you prefer?

TD vs. Monte Carlo

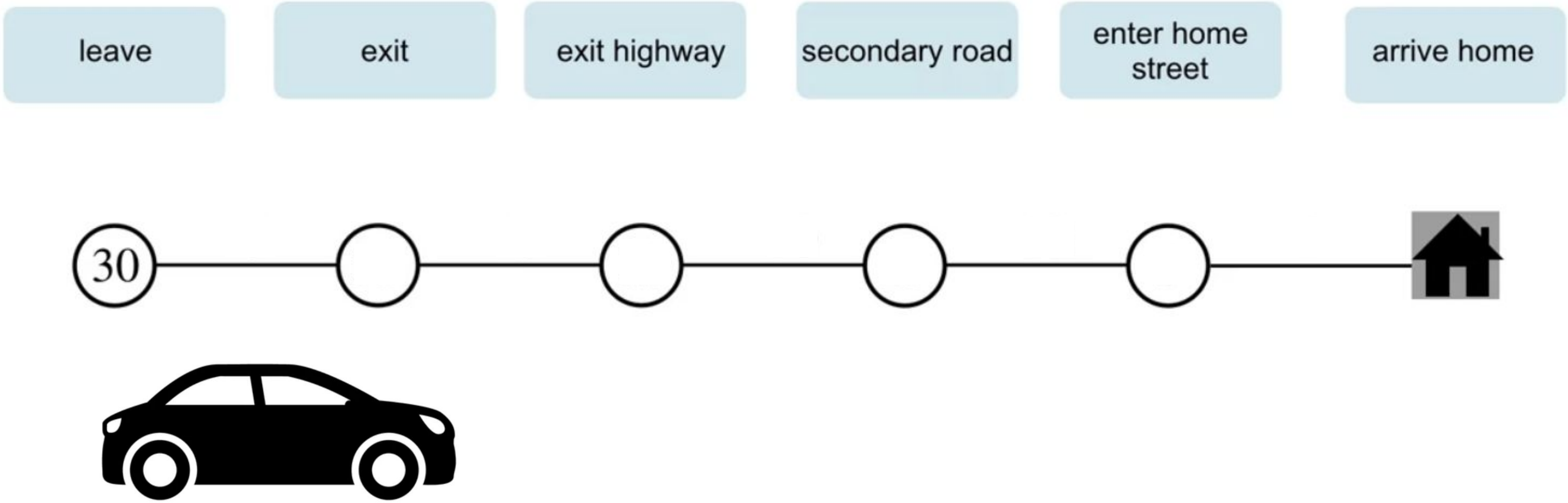
Temporal Difference:

- ❑ Works for both episodic and continuing tasks
- ❑ Updates immediately, does not need to wait until the end of the episode
- ❑ Higher bias (accuracy), lower variance (randomness)

Monte Carlo:

- ❑ <u>ONLY</u> works for episodic tasks
- ❑ Waits until the end of each episode to update
- ❑ Lower bias (accuracy), higher variance (randomness)

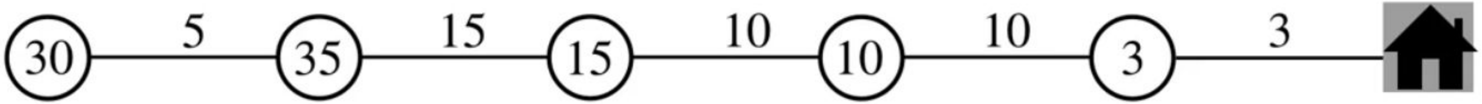
TD vs Monte Carlo Example



0 mins
elapsed

Monte Carlo

leave | exit | exit highway | secondary road | enter home street | arrive home



G_0

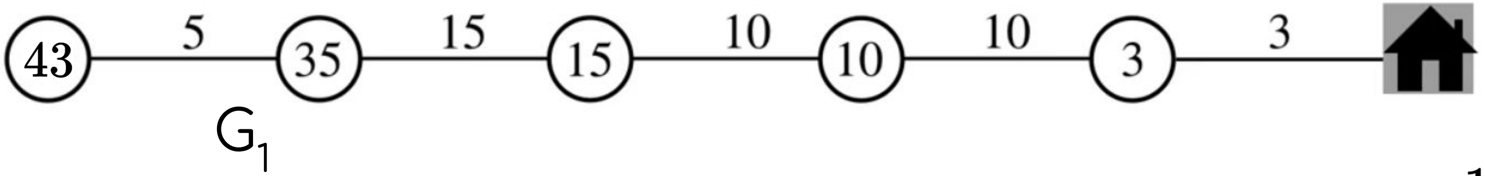
$$V(S_t) = V(S_t) + \alpha[G_t - V(S_t)] \qquad \alpha = 1$$

$$G_0 = 5 + 15 + 10 + 10 + 3 = 43$$

$$V(S_0) = 30 + (43 - 30) = 43$$

Monte Carlo

leave | exit | exit highway | secondary road | enter home street | arrive home



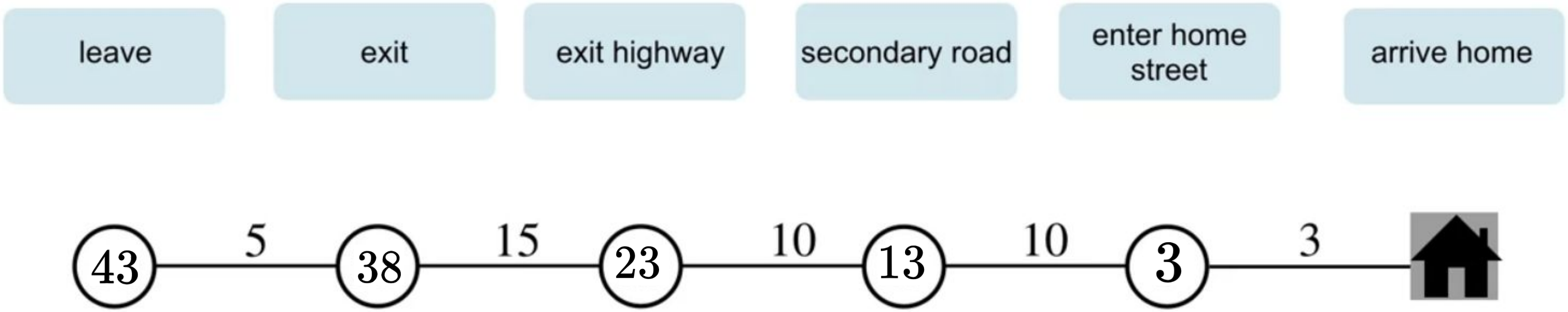
$$V(S_t) = V(S_t) + \alpha[G_t - V(S_t)]$$

$$\alpha = 1$$

$$G_1 = 15 + 10 + 10 + 3 = 38$$

$$V(S_1) = 35 + (38 - 35) = 38$$

Monte Carlo



$$V(S_t) = V(S_t) + \alpha[G_t - V(S_t)]$$

$$\alpha = 1$$

As seen, need to wait until the episode ends to update the value estimates

Temporal Difference

leave | exit | exit highway | secondary road | enter home street | arrive home

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \qquad \alpha, \gamma = 1$$

$$V(S_0) = 30 + [5 + 35 - 30] = 40$$

Can update right away!

Temporal Difference

leave | exit | exit highway | secondary road | enter home street | arrive home

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \qquad \alpha, \gamma = 1$$

$$V(S_1) = 35 + [15 + 15 - 35] = 30$$

In practice, usually TD > MC

WHY?

Temporal Difference Control

TD for Control

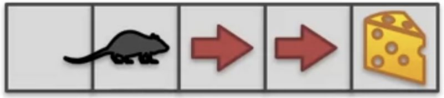
Recall: Control is the act of improving a policy

Recall: We want to learn the state-action value function [$Q_\pi(s,a)$], or the Q-function, to do model-free control

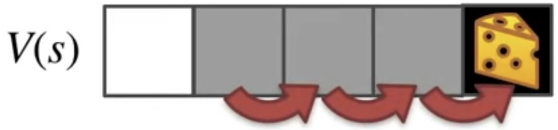
Just like how we used Monte Carlo to find the Q-function, we can also use TD for the same purpose!

State Value to Action Values

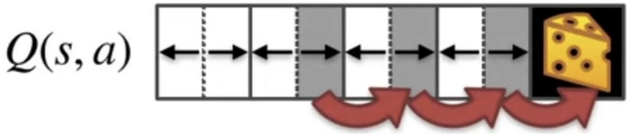
We have mostly been talking about the state-value function so far. However, begin thinking about state-action pair to state-action pair, rather than state to state



State to state



State-action to state-action



State Value to Action Values

Recall the state-value function TD update equation which uses V(s):

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Is there an analogous update rule for using the state-action value function Q(s,a) instead?

Of course!

SARSA

The SARSA Algorithm

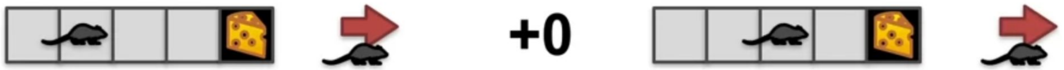
SARSA stands for (S)tate, (A)ction, (R)eward, Next (S)tate, Next (A)ction, the components that go into an update

$$S_t$$

The next action A_{t+1} is sampled from the current policy

The SARSA Update Equation

$S_t \quad A_t \quad R_{t+1} \quad S_{t+1} \quad A_{t+1}$ ~ ε-greedy

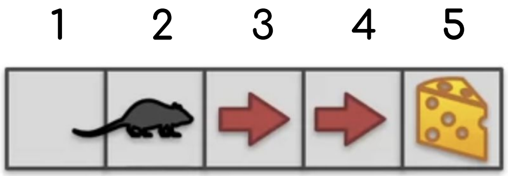


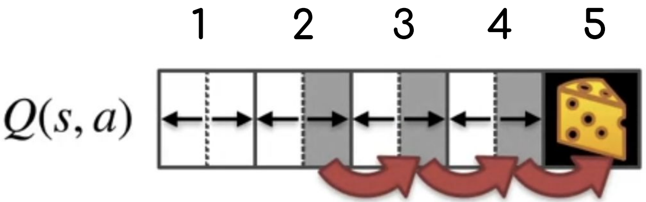
$$\boxed{Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))}$$

Notice how similar this is to the state value update!

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

SARSA Update Example





Uniform Random Policy: Left, Right, α = 1, γ = 1

Starting State-Action Pair	Reward	Next State-Action Pair
(2,→)	+0	(3,→)
Q = 1		Q = 2

Randomly sampled from policy!

$$Q(2, \rightarrow) \leftarrow Q(2, \rightarrow) + (0 + Q(3, \rightarrow) - Q(2, \rightarrow)) = 1 + (0 + 2 - 1) = 2$$

Notebook:
SARSA

SARSA

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

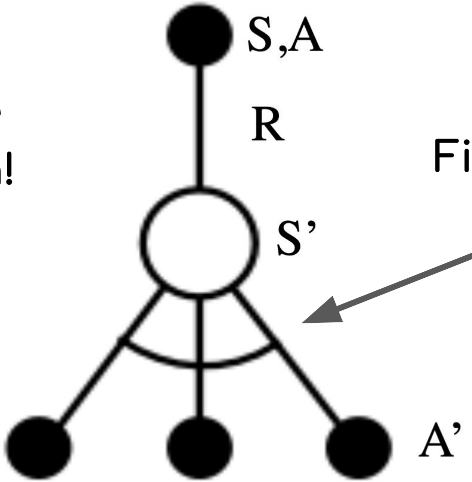
Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-Learning

Q-Learning Update Rule

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

Directly learns the optimal Q function!



Find the maximum Q-value over the next actions

Revisiting the Bellman Equations

SARSA:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

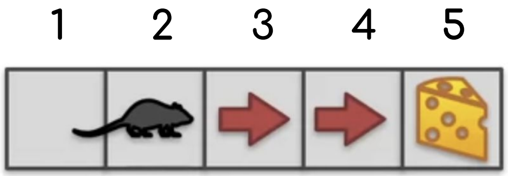
$$Q_\pi(s, a) = \sum_{s'} \sum_{r} p(s', r | s, a)[r + \gamma \sum_{a'} \pi(a'|s') Q_\pi(s', a')]$$

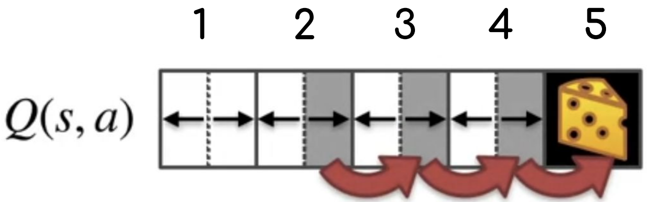
Q-Learning:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$$

$$Q_*(s, a) = \sum_{s'} \sum_{r} p(s', r | s, a)[r + \gamma \max_{a} Q_*(s', a')]$$

Q-Learning Update Example





Uniform Random Policy: Left, Right, α = 1, γ = 1

Starting State-Action Pair	Reward	Next State-Action Pair(s)	
(2,→)	+0	(3,→)	(3,←)
Q = 1		$\boxed{Q = 2}$	Q = 1

$$Q(2, \rightarrow) \leftarrow Q(2, \rightarrow) + (0 + Q(3, \rightarrow) - Q(2, \rightarrow)) = 1 + (0 + 2 - 1) = 2$$

Notebook: Q-Learning

Q-Learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

Expected SARSA

Expected SARSA Formulation

Recall the Bellman equation for Q and SARSA

$$Q_\pi(s,a) = \underline{\sum_{s'}\sum_{r} p(s',r|s,a)}[r + \gamma \underline{\sum_{a'}} \pi(a'|s')Q_\pi(s',a')]$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

Why do we need to sample an action when we already have the policy? Can't we just take the summation (expectation) directly?

Expected SARSA Algorithm

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \boxed{\sum_{a'} \pi(a'|S_{t+1})Q(S_{t+1}, a')} - Q(S_t, A_t))$$

$Q(S_{t+1}, a') =$	0.0	-1.0	2.0	1.0	Values of actions in next state
$\pi(a'\|S_{t+1}) =$	0.1	0.1	0.7	0.1	How likely those actions are

$$\sum_{a'} \pi(a'|S_{t+1})Q(S_{t+1}, a') = (0.1)(0.0) + (0.1)(-1.0) + (0.7)(2.0) + (0.1)(1.0) = 1.4$$

We have the policy, so just calculate the expectation directly!

Stability in Update Target

$Q(S_{t+1}, a') =$	0.0	−1.0	2.0	1.0
$\pi(a' \mid S_{t+1}) =$	0.1	0.1	0.7	0.1

SARSA:

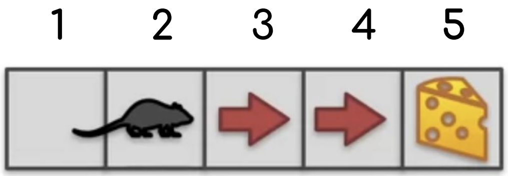
Possible Updates

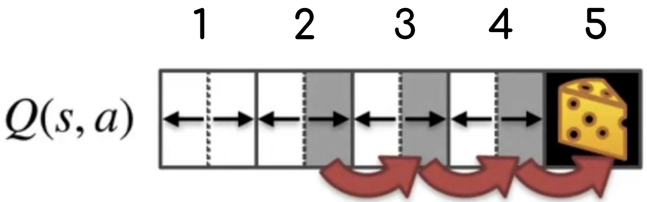
$$1.0 + \gamma(2.0) \quad 1.0 + \gamma(-1.0) \quad 1.0 + \gamma(1.0) \quad 1.0 + \gamma(0.0)$$

$\approx 1.0 + \gamma(1.4)$ (in expectation)

Expected SARSA: $1.0 + \gamma(1.4)$ ← Actual Update

Expected SARSA Update Example





Uniform Random Policy: Left, Right, α = 1, γ = 1

Starting State-Action Pair	Reward	Next State-Action Pair(s)	
(2,→)	+0	(3,→)	(3,←)
Q = 1		Q = 2	Q = 1

$$Q(2,\rightarrow) = Q(2,\rightarrow) + (0 + [(0.5)Q(3,\leftarrow) + (0.5)Q(3,\rightarrow)] - Q(2,\rightarrow)) = 1 + (0.5 + 1 - 1) = 1.5$$

End of Main Lectures - What's Next?

Upcoming Stuff

- There is another session next week
  - Jessica will be sharing her RL research (actual, practical RL)
  - Professor Michael Bowling from the University of Alberta is doing a guest lecture about Deep Q Networks **(extension of Q-Learning)**
- RL TOURNAMENT!!!!
  - Next semester
  - Amazing chance to learn, apply RL

Future Directions

To “solve” a reinforcement learning problem, you can find TWO things

The optimal policy OR the optimal Q-function

We have spoken about methods to find the optimal Q-function in this course (MC with QF, SARSA, Q-Learning, Expected SARSA). Let me call them Q-function (QF) methods

We did not speak at all about methods which do NOT find an optimal Q-function and find only the optimal policy. These are policy gradient (PG) methods

QF and PG methods can be considered a ‘family’ of algorithms, and in each family, there are important RL methods which are used widely today

Future Directions

Below is my personal "grouping" of algorithms in the same family, which you can use for your study. There are all very well known, important algorithms which you can start with

Track 1: PG

Vanilla Policy Gradient

↓

Natural Policy Gradient

↓

Trust Region Policy Optimization

↓

Proximal Policy Optimization

Track 2: QF

Deep Q-Networks

↓

Deep Deterministic Policy Gradient

↙ ↘

Twin Delayed DDPG | Soft Actor Critic

End of Week 3 Questions?

Please Fill Out the Feedback Survey