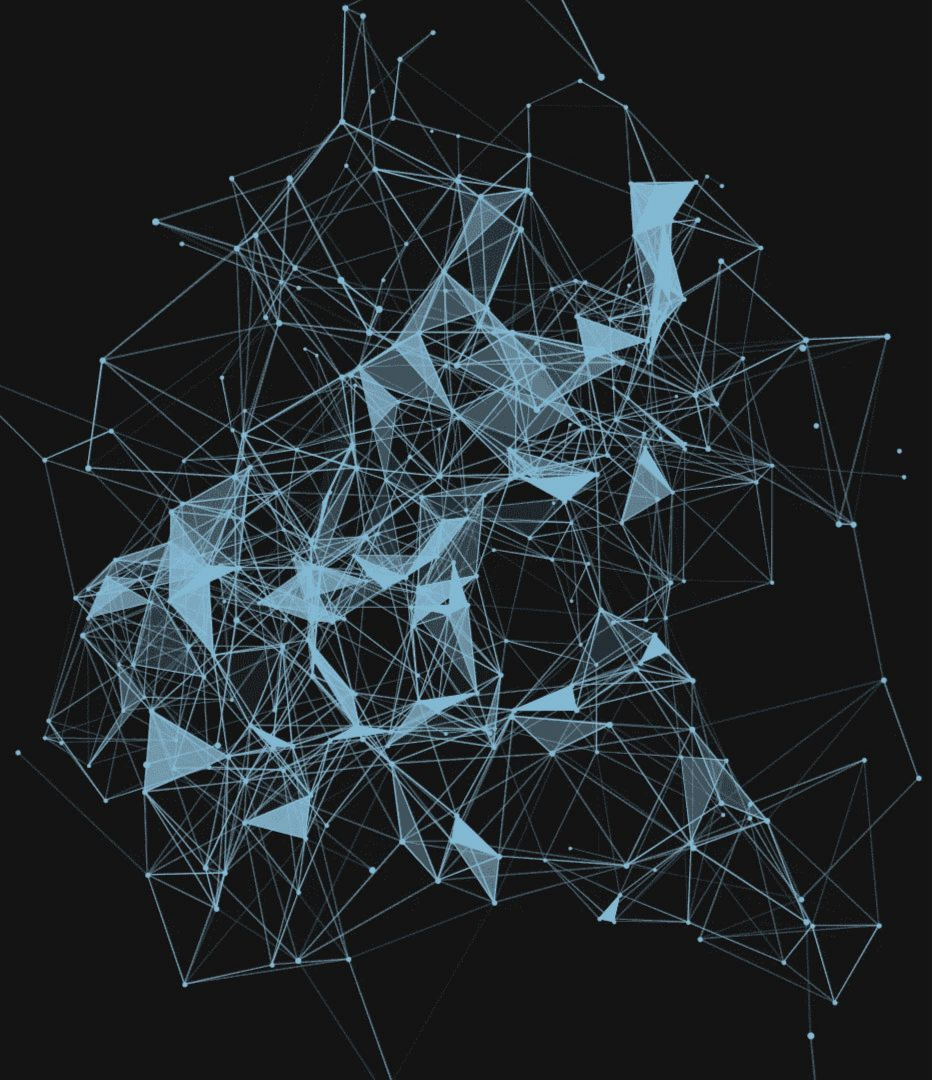
Reinforcement Learning Lesson - 3

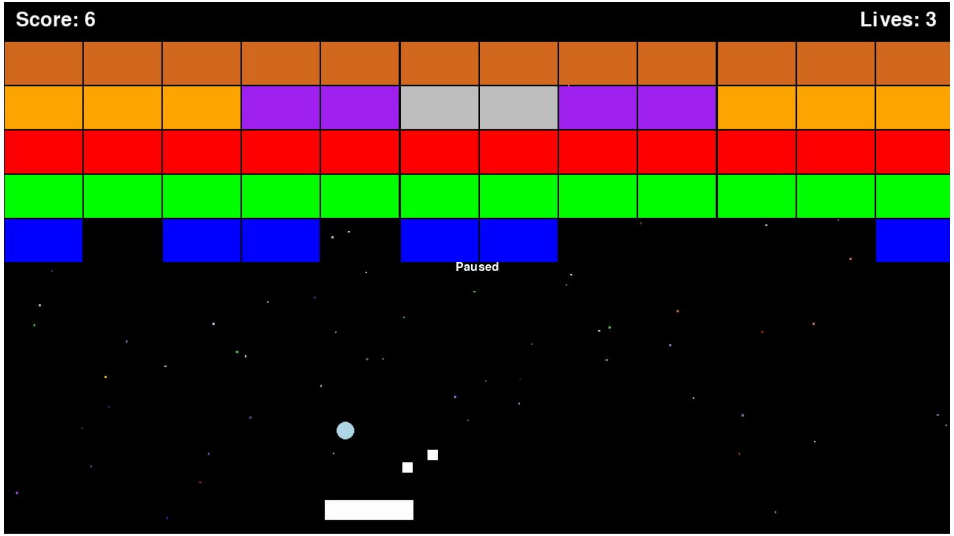



RENEFENENEEND
LEARNING
REWART
FOTCION
COVINE
RERENENTONT
LEARNING
QUIZ
TIME
REWERNICE
FUNCION
CUVE
QUIZ TIME
REWARN
FOTCION
REWARD
FUNCION
CUVE
REREENSION
LEARNING
QZ-2
TIME

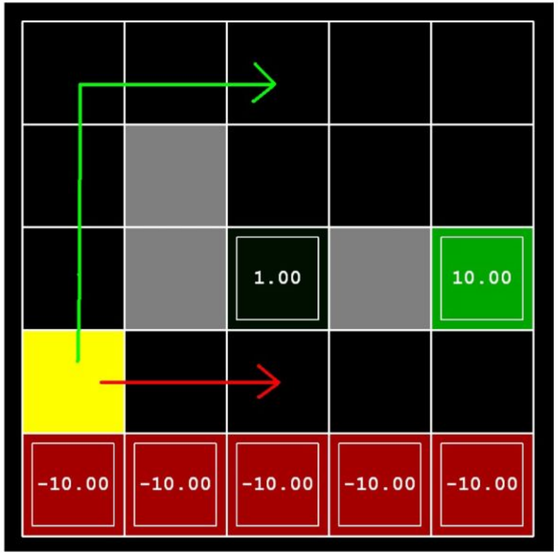
Breakout

Game Scenario:

Imagine you are playing Breakout. Your paddle is positioned near the bottom of the screen, and a ball is bouncing towards it. The ball has already destroyed a few bricks on the top right side of the wall. You have 3 lives left, and the ball is about to hit the paddle.



Exercise 1: Effect of Discount and Noise



(a) Prefer the close exit (+1), risking the cliff (-10)

(b) Prefer the close exit (+1), but avoiding the cliff (-10)

(c) Prefer the distant exit (+10), risking the cliff (-10)

(d) Prefer the distant exit (+10), avoiding the cliff (-10)

(1) $\gamma = 0.1$, noise = 0.5

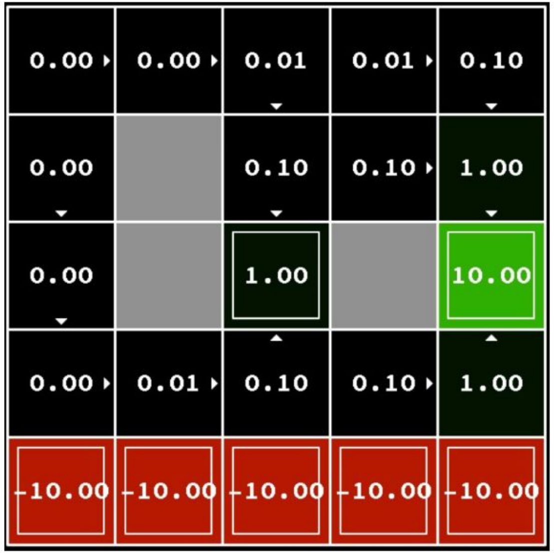
(2) $\gamma = 0.99$, noise = 0

(3) $\gamma = 0.99$, noise = 0.5

(4) $\gamma = 0.1$, noise = 0

Credit: Pieter Abbeel foundation of deep RL

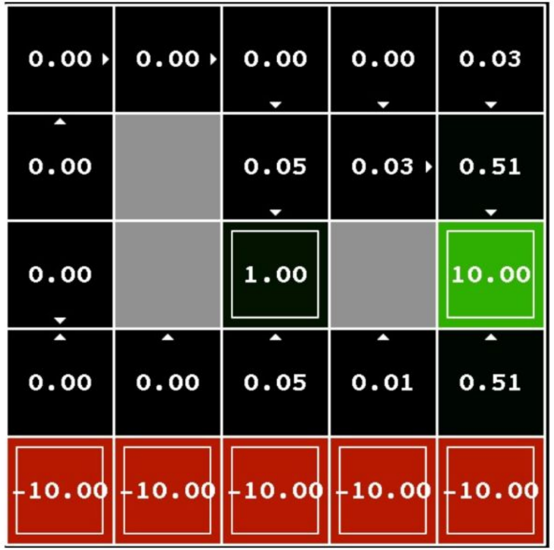
Exercise 1 Solution



(a) Prefer close exit (+1), risking the cliff (-10) --- (4) $\gamma = 0.1$, noise = 0

Credit: Pieter Abbeel
foundation of deep RL

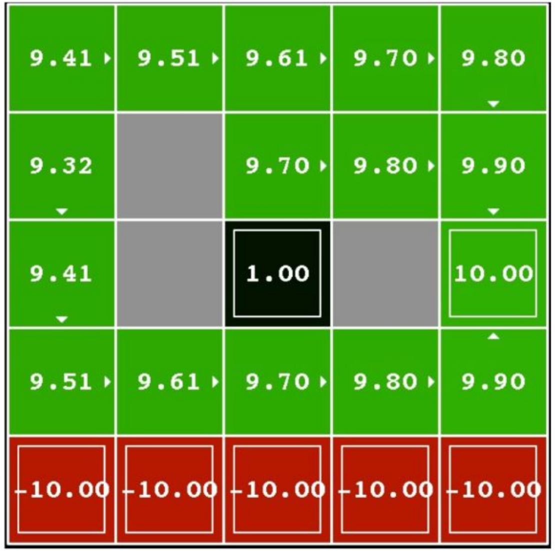
Exercise 1 Solution



(b) Prefer close exit (+1), avoiding the cliff (-10) --- (1) $\gamma = 0.1$, noise = 0.5

Credit: Pieter Abbeel foundation of deep RL

Exercise 1 Solution



(c) Prefer distant exit (+10), risking the cliff (-10) --- (2) $\gamma = 0.99$, noise = 0

Credit: Pieter Abbeel foundation of deep RL

Exercise 1 Solution

(d) Prefer distant exit (+10), avoid the cliff (-10) --- (3) $\gamma = 0.99$, noise = 0.5

Credit: Pieter Abbeel
foundation of deep RL

Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

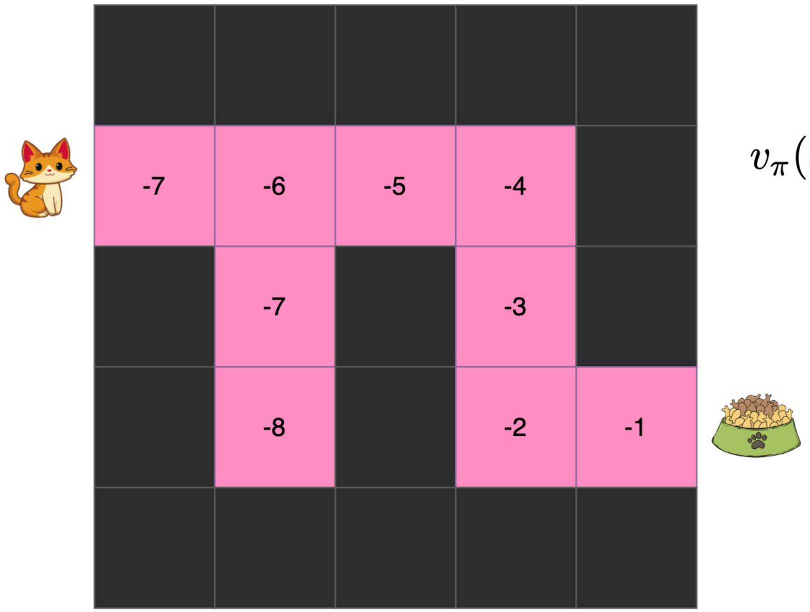
2. Policy Evaluation
   Loop:
   - $\Delta \leftarrow 0$
   - Loop for each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - *old-action* $\leftarrow \pi(s)$
   - $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
   - If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

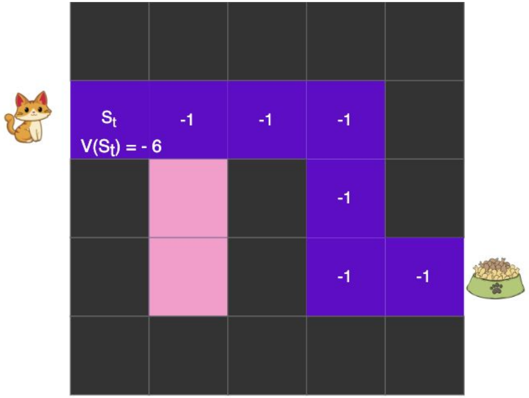
Bellman Equation

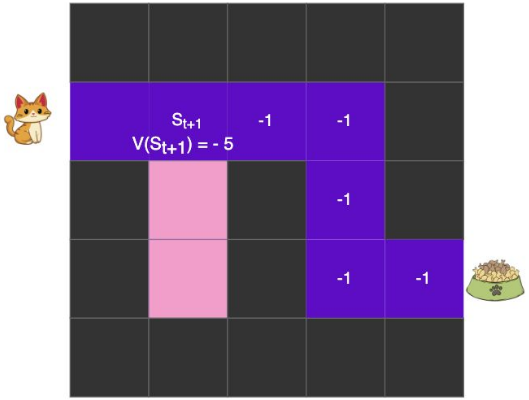


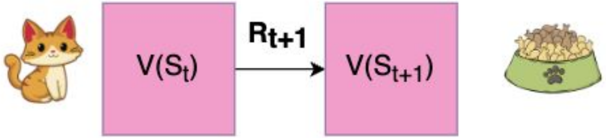
$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \,|\, s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big]
\end{aligned}
$$

Bellman Equation

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_t = s\right] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a)\Big[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']\Big] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \boxed{\gamma v_\pi(s')}\Big],
\end{aligned}
$$







$V(S_t) = R_{t+1} + \gamma * V(S_{t+1})$

$V(S_t) = -1 + 1 * (-5) = -6$

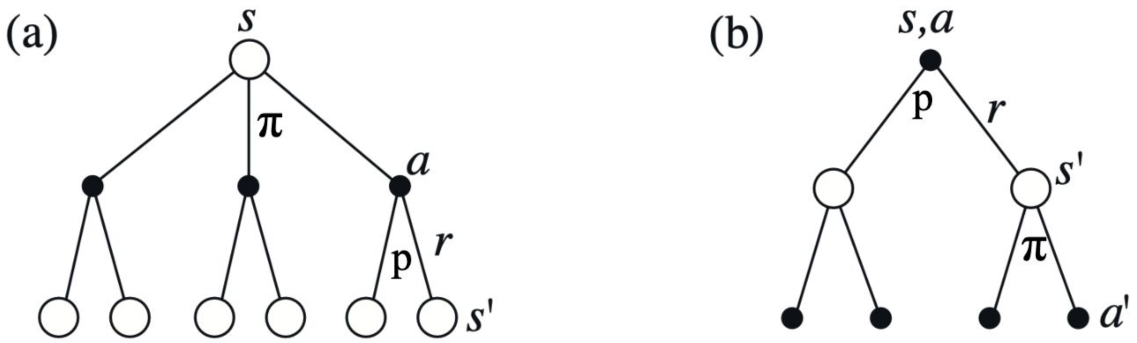
Action-Value function for policy π

Similarly, we define the value of taking action a in state s under a policy π, denoted $q_\pi(s, a)$, as the expected return starting from s, taking the action a, and thereafter following policy π:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]. \tag{3.13}$$

Backup Diagram

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



○ → a state

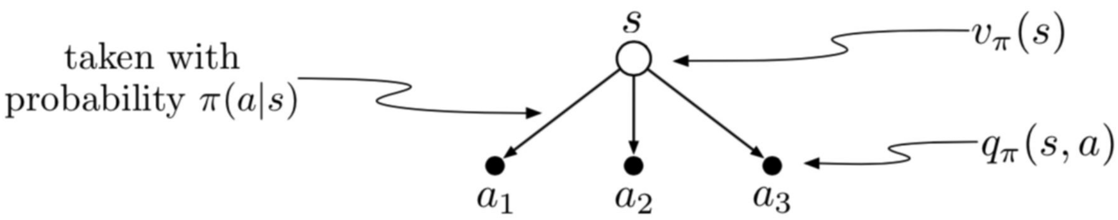
● → state-action pair

Figure 3.4: Backup diagrams for (a) v_π and (b) q_π.

What is the Bellman equation for action values for qπ?

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

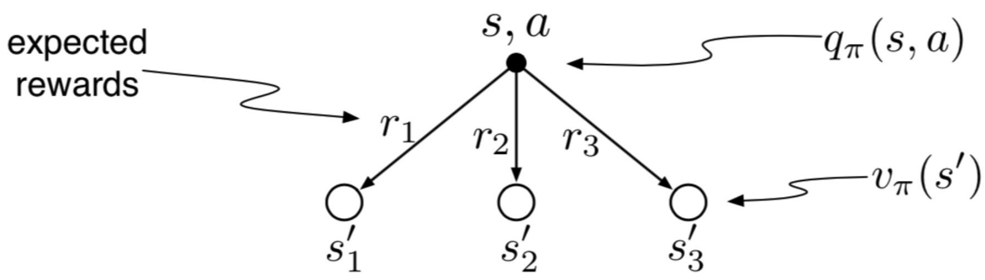
Backup Diagram



Backup diagram for example 3.18

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s, a)$$

Backup Diagram



Backup diagram for example 3.19

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

Optimal Value Functions

Optimal state-value function,

$$v_*(s) \doteq \max_\pi v_\pi(s),$$

for all $s \in \mathcal{S}$.

Optimal action-value function,

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a),$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$.

Optimal Value Functions

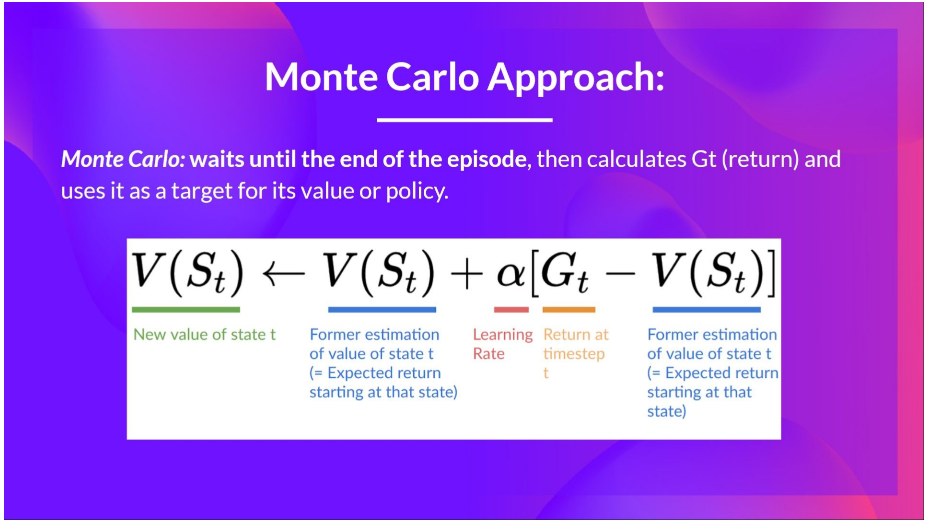
Figure 3.7: Backup diagrams for (a) v_* and (b) q_*

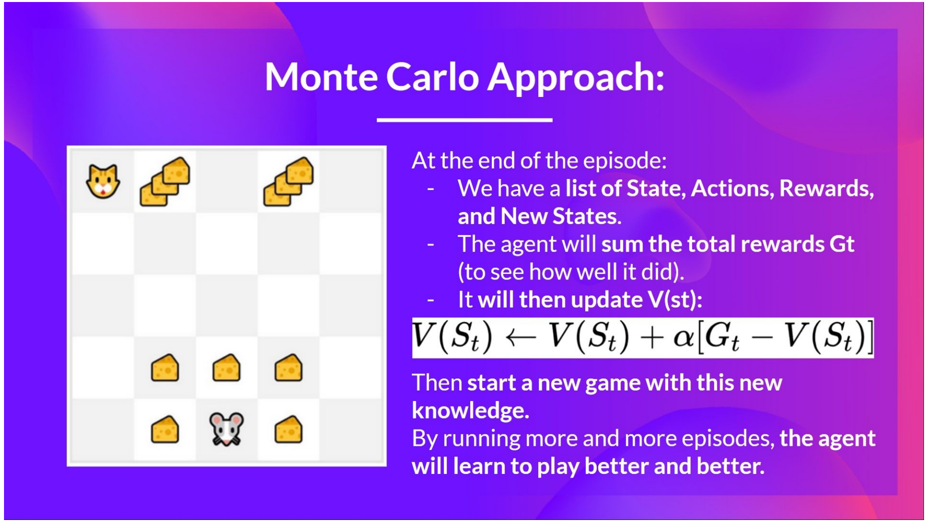
$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r | s, a) \big[r + \gamma v_*(s') \big].$$

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

Monte Carlo vs Temporal Difference

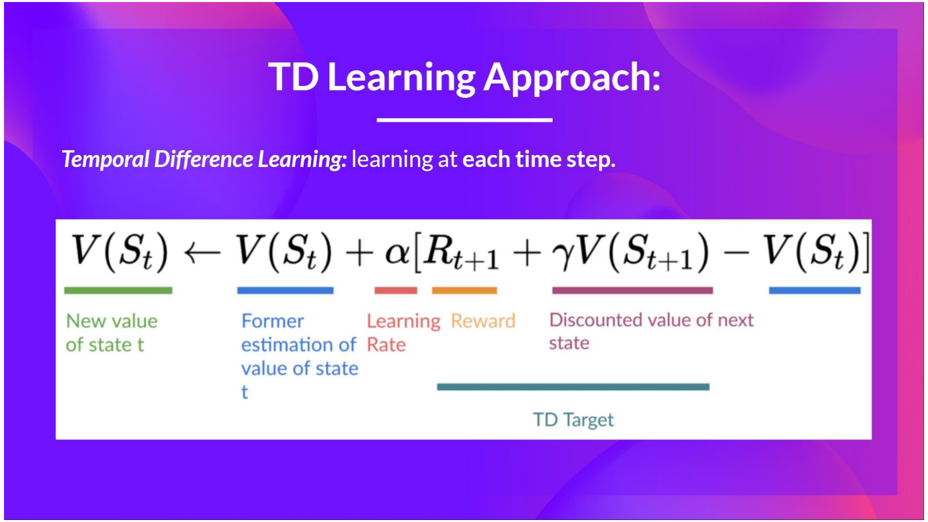
Monte Carlo: learning at the end of the episode

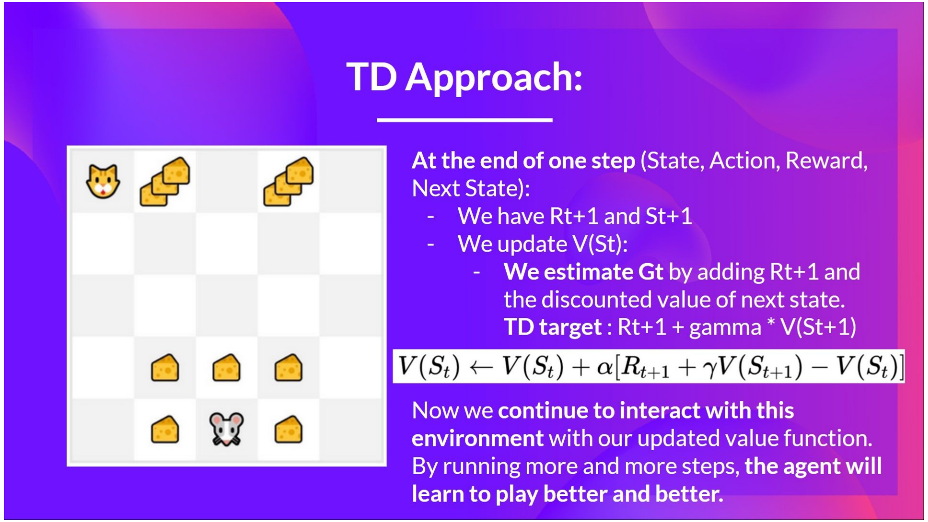




Monte Carlo vs Temporal Difference

Temporal Difference Learning: learning at each step





$$NewEstimate \leftarrow OldEstimate \; + \; StepSize \left[Target - OldEstimate \right].$$

Monte Carlo vs Temporal Difference

Monte Carlo: $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$

TD Learning: $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$

Q-Learning

Q-Learning is an off-policy value-based method that uses a TD approach to train its action-value function:

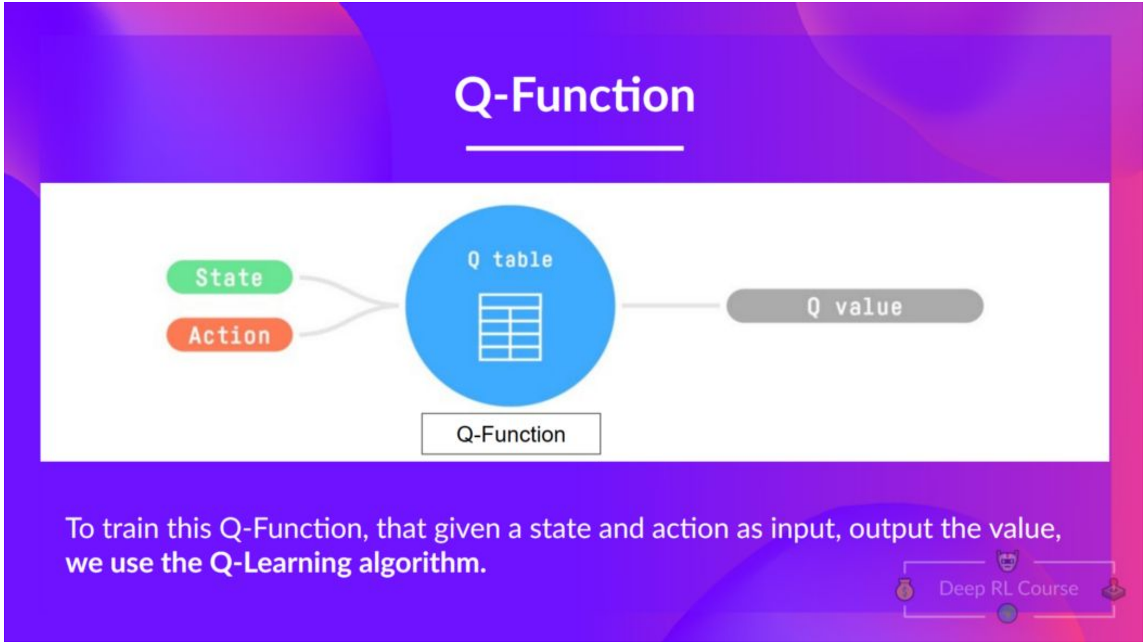
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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

Q-Learning

Q-Learning is an off-policy value-based method that uses a TD approach to train its action-value function:



https://huggingface.co/learn/deep-rl-course/en/unit2/q-learning

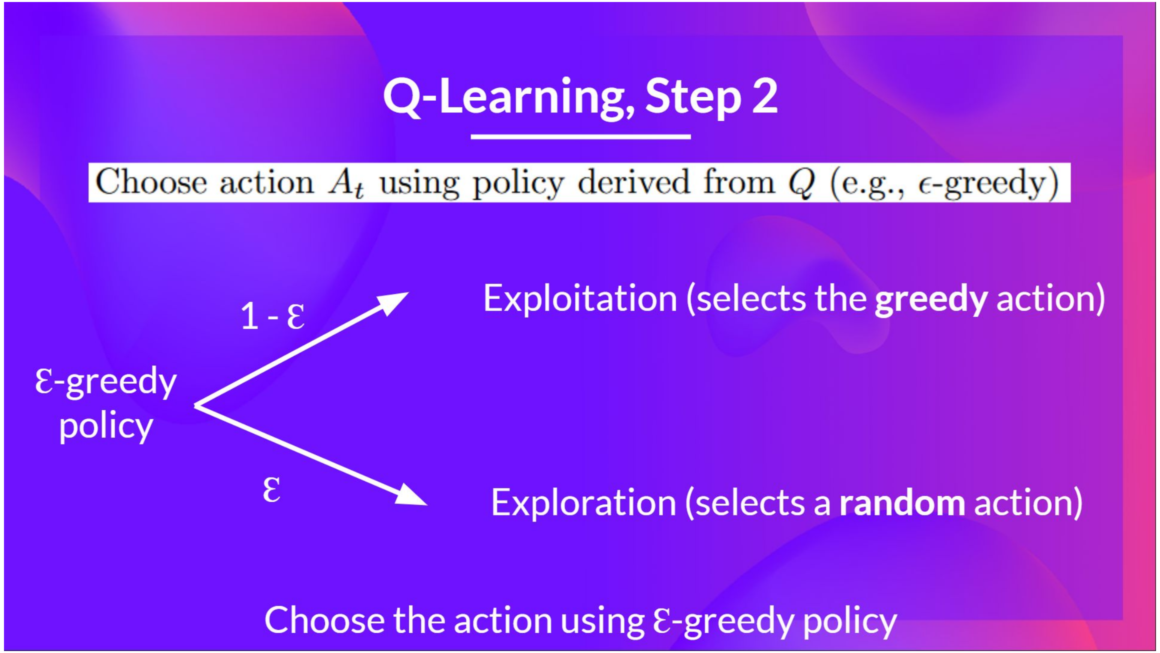
Q-Learning

Step 1: We initialize the Q-table

https://huggingface.co/learn/deep-rl-course/en/unit2/q-learning

Q-Learning

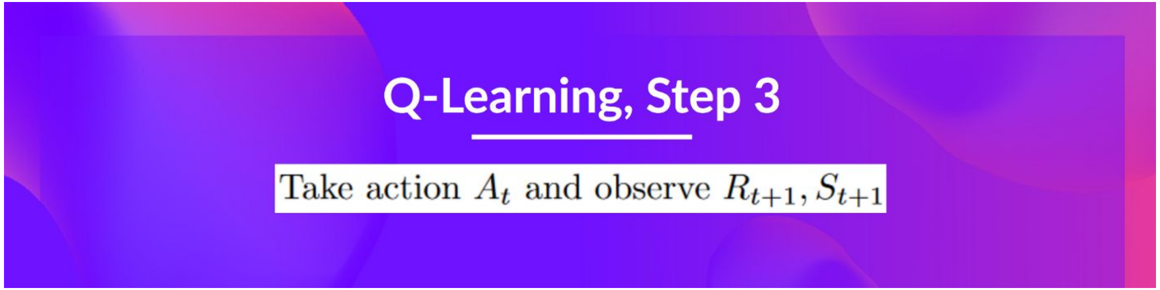
Step 2: Choose an action using the epsilon-greedy strategy



https://huggingface.co/learn/deep-rl-course/en/unit2/q-learning

Q-Learning

Step 3: Perform action At, get reward Rt+1 and next state St+



https://huggingface.co/learn/deep-rl-course/en/unit2/q-learning

Q-Learning

Step 4: Update Q(St, At)

$$\underbrace{Q(S_t, A_t)}_{\text{New Q-value estimation}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{Former Q-value estimation}} + \underbrace{\alpha}_{\text{Learning Rate}} \underbrace{[\underbrace{\underbrace{R_{t+1}}_{\text{Immediate Reward}} + \underbrace{\gamma max_a Q(S_{t+1}, a)}_{\text{Discounted Estimate optimal Q-value of next state}}}_{\text{TD Target}} - \underbrace{Q(S_t, A_t)}_{\text{Former Q-value estimation}}]}_{\text{TD Error}}$$

https://huggingface.co/learn/deep-rl-course/en/unit2/q-learning