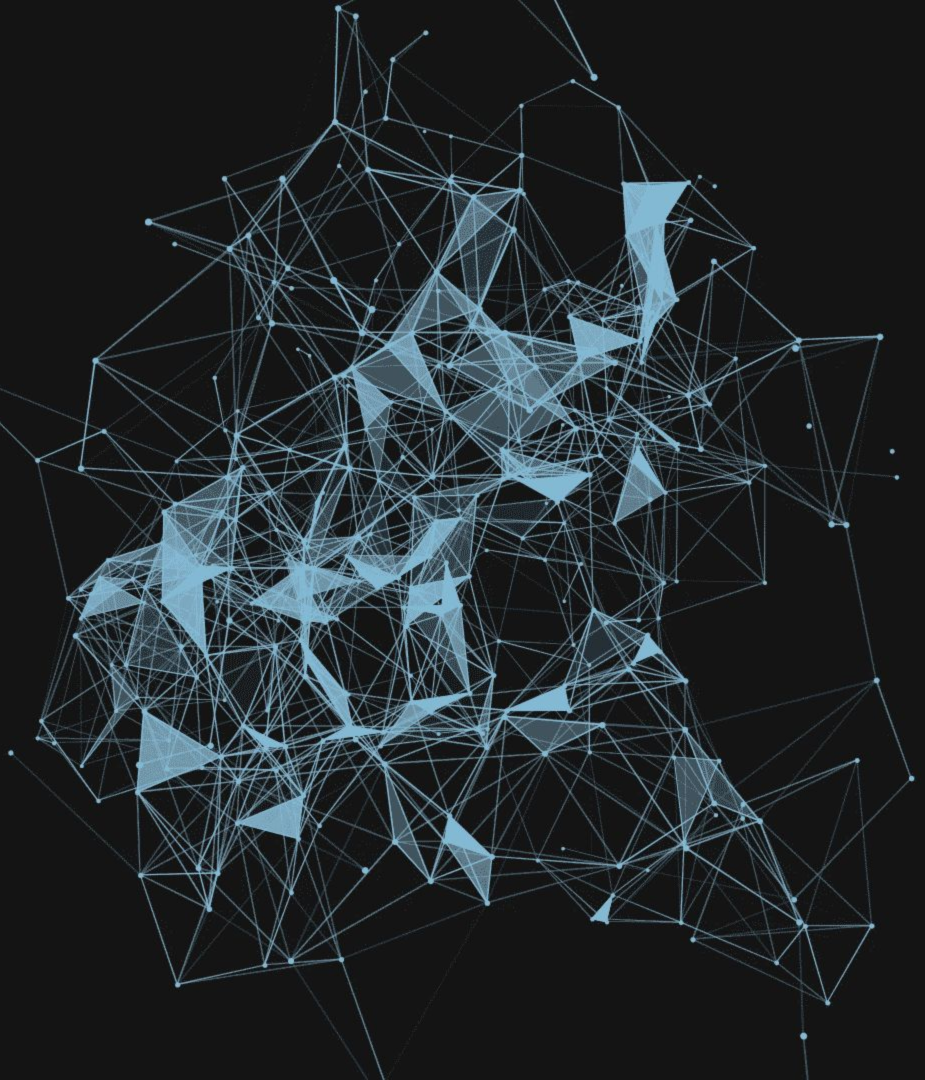
Reinforcement Learning Lesson - 4



Monte Carlo vs Temporal Difference

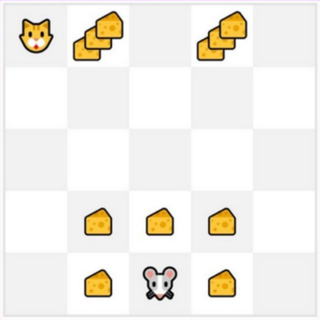
Monte Carlo: learning at the end of the episode

Monte Carlo Approach:

***Monte Carlo:* waits until the end of the episode**, then calculates Gt (return) and uses it as a target for its value or policy.

$$\underbrace{V(S_t)}_{\text{New value of state t}} \leftarrow \underbrace{V(S_t)}_{\substack{\text{Former estimation of value of state t} \\ \text{(= Expected return starting at that state)}}} + \underbrace{\alpha}_{\text{Learning Rate}}[\underbrace{G_t}_{\text{Return at timestep t}} - \underbrace{V(S_t)}_{\substack{\text{Former estimation of value of state t} \\ \text{(= Expected return starting at that state)}}}]$$

Monte Carlo Approach:



At the end of the episode:
- We have a **list of State, Actions, Rewards, and New States**.
- The agent will **sum the total rewards Gt** (to see how well it did).
- **It will then update V(st):**

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

Then **start a new game with this new knowledge.**
By running more and more episodes, **the agent will learn to play better and better.**

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

Monte Carlo vs Temporal Difference

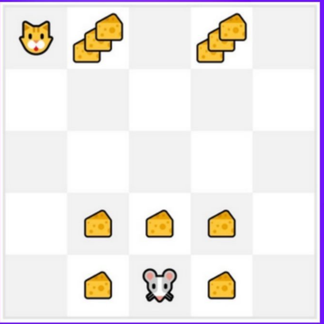
Temporal Difference Learning: learning at each step

TD Learning Approach:

Temporal Difference Learning: learning at **each time step.**

$$\underbrace{V(S_t)}_{\text{New value of state t}} \leftarrow \underbrace{V(S_t)}_{\text{Former estimation of value of state t}} + \underbrace{\alpha}_{\text{Learning Rate}}[\underbrace{\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma V(S_{t+1})}_{\text{Discounted value of next state}}}_{\text{TD Target}} - \underbrace{V(S_t)}_{}]$$

TD Approach:



At the end of one step (State, Action, Reward, Next State):

- We have Rt+1 and St+1
- We update V(St):
  - **We estimate Gt** by adding Rt+1 and the discounted value of next state.
    TD target : Rt+1 + gamma * V(St+1)

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

Now we **continue to interact with this environment** with our updated value function. By running more and more steps, **the agent will learn to play better and better.**

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

Monte Carlo vs Temporal Difference

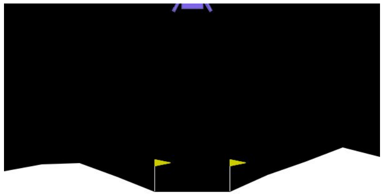
Monte Carlo: $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$

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


AI
GAME
MODE
ON
GAME
MODE
GAME
MODE
GAME
MODE
MODE
ON

Toy Text

Box2D



https://gymnasium.farama.org/environments/toy_text/

Q-Learning

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

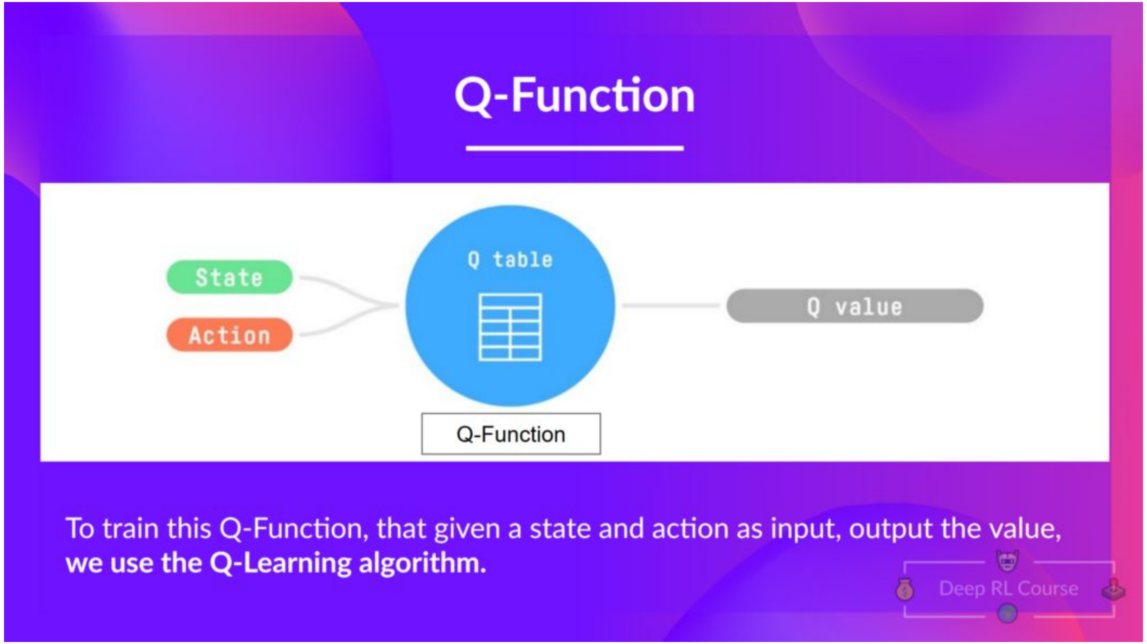
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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

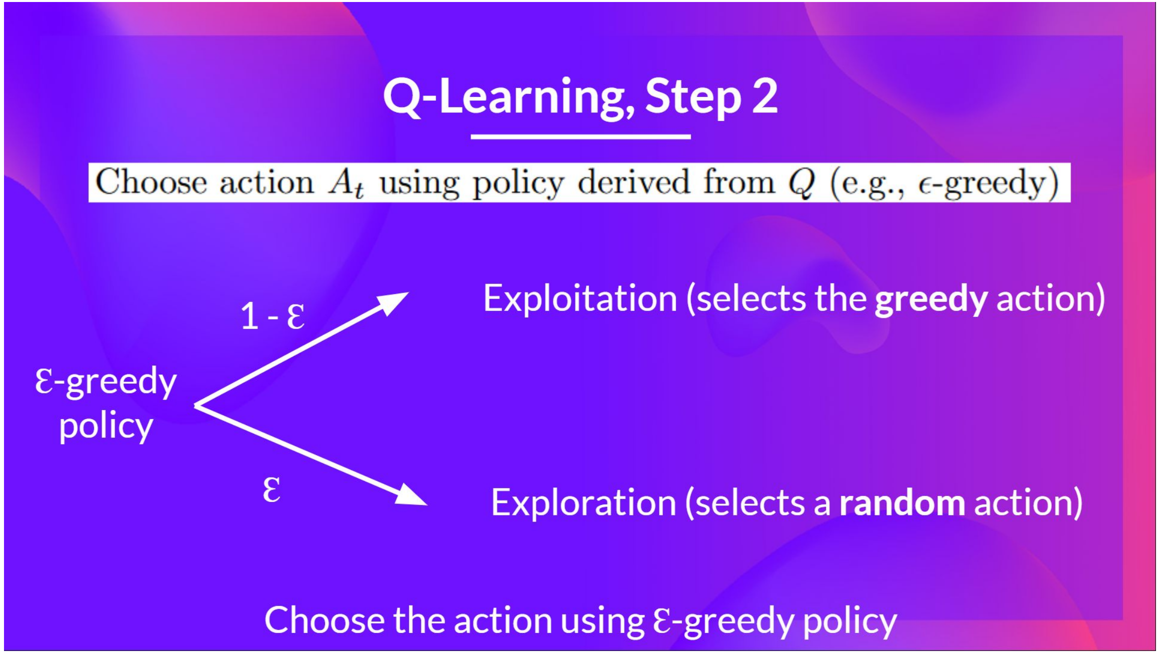
Step 1: We initialize the Q-table

	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

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

Q-Learning

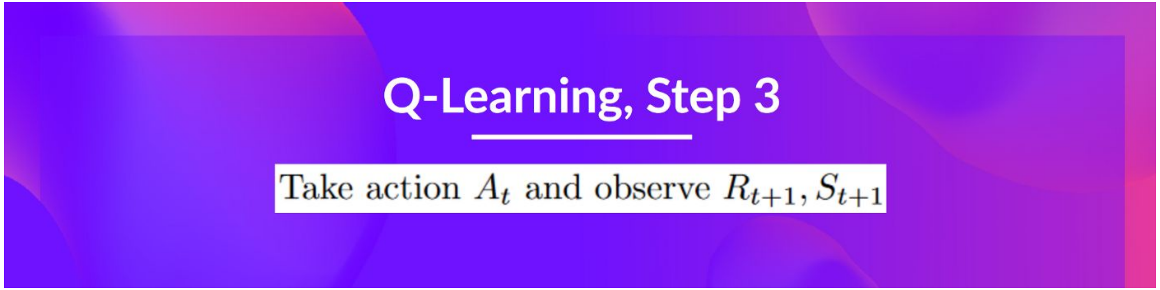
Step 2: Choose an action using the epsilon-greedy strategy



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

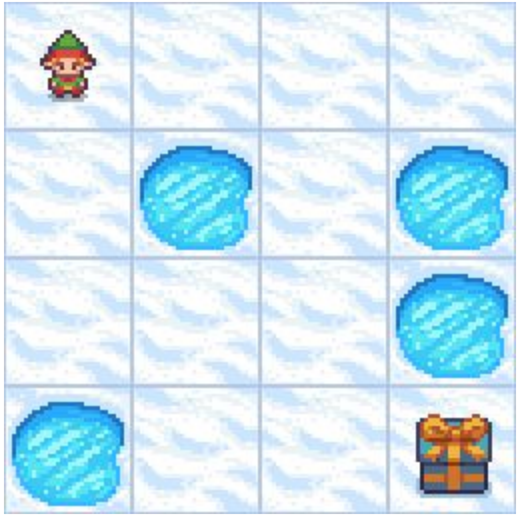
Q-Learning

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

Frozen Lake



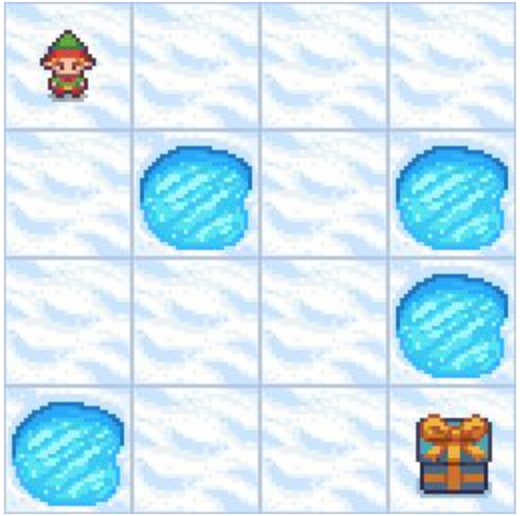
Action Space	Discrete (4) 0: Move left 1: Move down 2: Move right 3: Move up
Observation Space	Discrete (16)
Reward	Reach goal: +1 Reach hole: 0 Reach frozen: 0
Termination	Move into a hole. Reach the goal.

Frozen Lake

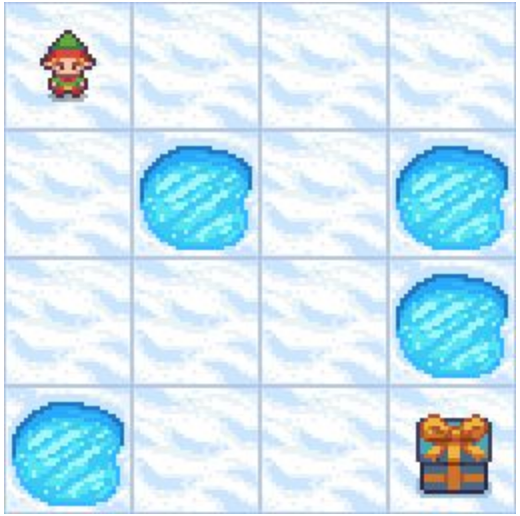
What patterns do you observe in the agent's path to the goal?

How often do you think the agent fall into holes or get stuck in loops?

What do you think we can do to improve agent behaviour?



Frozen Lake (Reward Shaping)



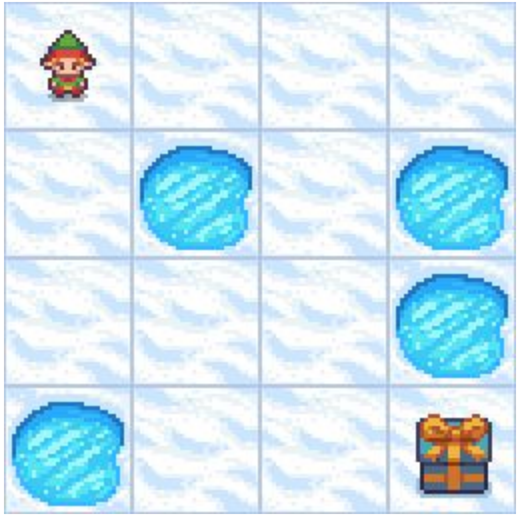
Action Space	Discrete (4) 0: Move left 1: Move down 2: Move right 3: Move up
Observation Space	Discrete (16)
Reward	Reach goal: +10 Reach hole: -5 Reach frozen: -0.1
Termination	Move into a hole. Reach the goal.

Frozen Lake (Reward Shaping)

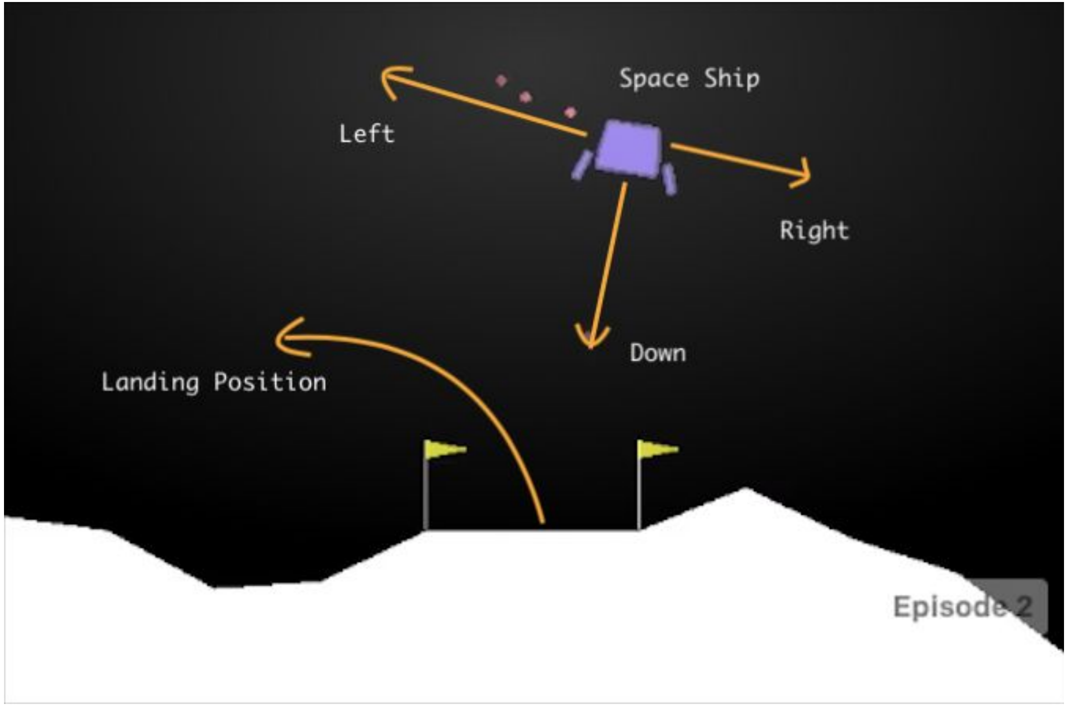
How does changing the reward structure affect the agent's behavior?

Does the agent learn faster with additional rewards for avoiding holes or getting closer to the goal?

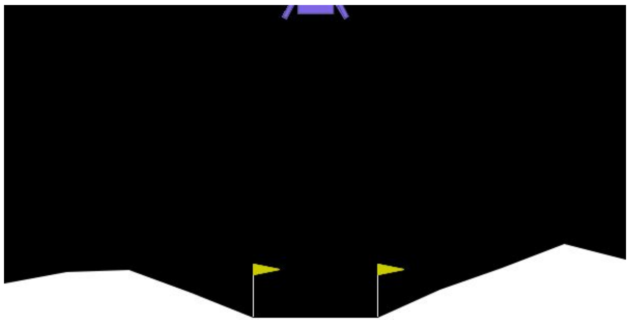
Aside from reward structure, what other parameters are important for agent behaviour?



Lunar Lander



Lunar Lander



Action Space	**Successful Landing:** +100 to +140. **Crash**: -100 points. **Main Engine:** -0.3 points per frame. **Side Engines:** -0.03 points per frame. **Legs Touching Ground:** +10 points per leg.