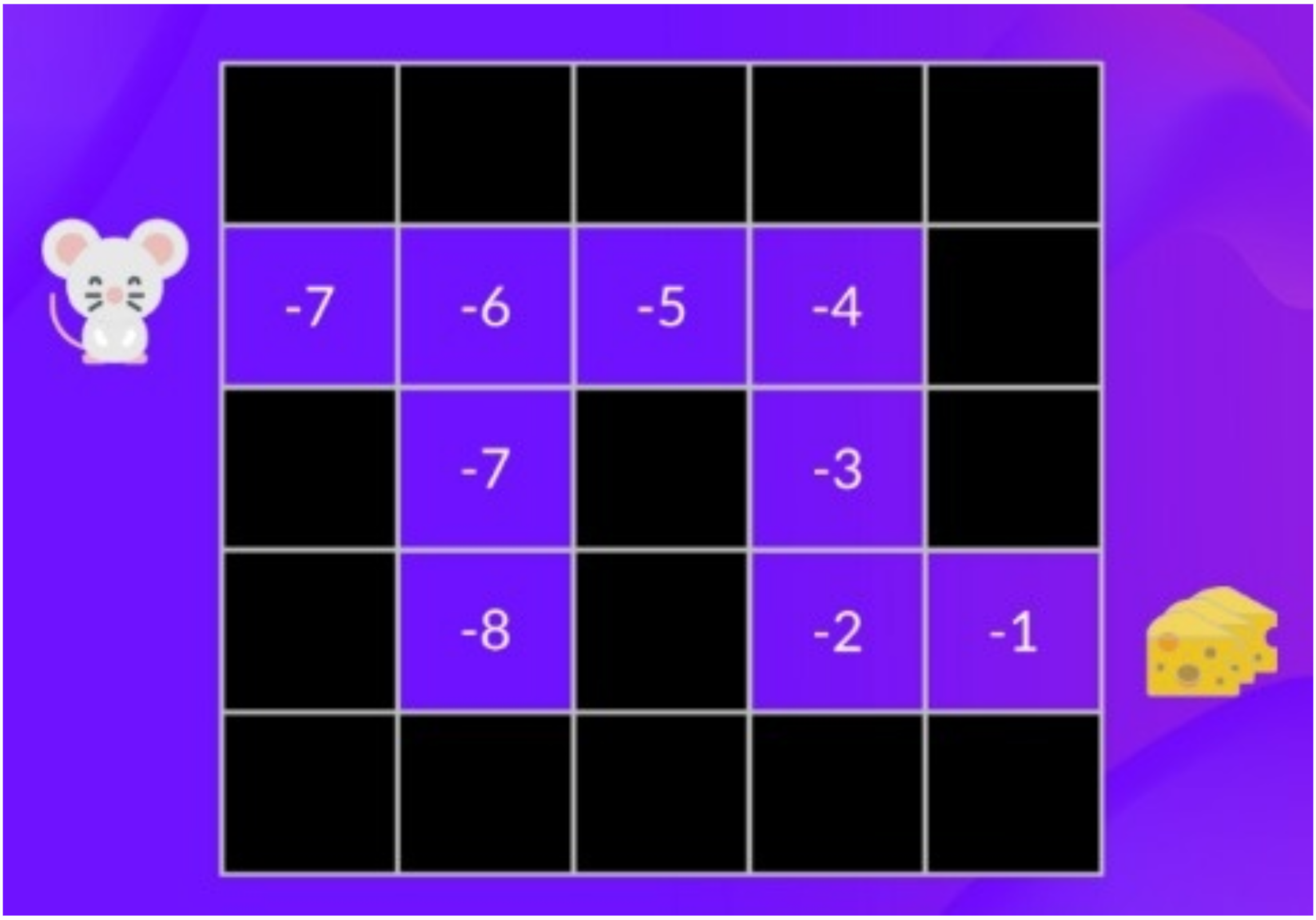
VALUE-BASED APPROACHES

STATE VALUE FUNCTION

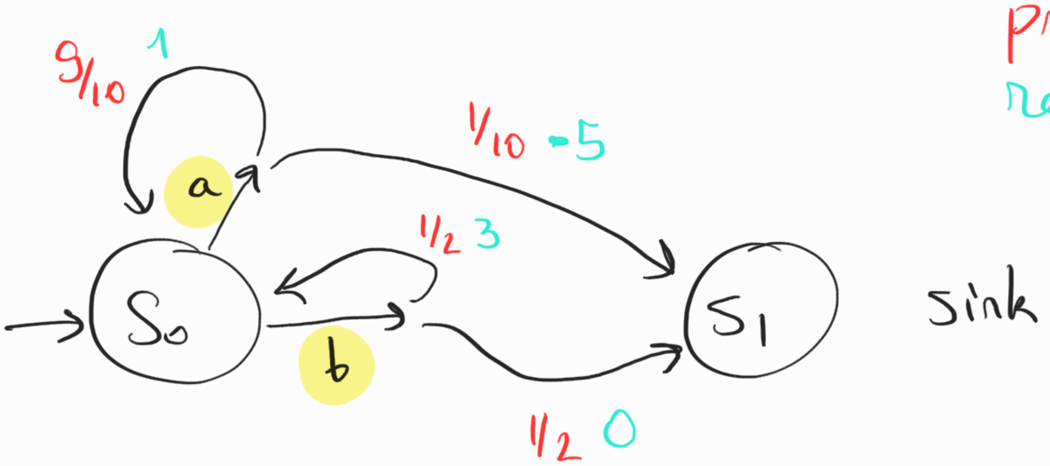
$$V_\pi : S \to \mathbb{R}$$

$$V_\pi(s) = \mathbb{E}\left[G \mid S(0) = s \wedge \pi \right]$$



BELLMAN EQUATIONS

probability
reward



$$\pi(s_0) = a$$

$$\begin{cases} V_\pi(s_0) = \frac{9}{10}\left(1 + \gamma V_\pi(s_0)\right) + \frac{1}{10}\left(-5 + \gamma V_\pi(s_1)\right) \\ V_\pi(s_1) = 0 \end{cases}$$

More generally:

$$V_\pi(s) = \mathbb{E}_{s',R}\left[R + \gamma V_\pi(s') \mid s, a\right]$$

optimal state value function:

$$V_* : S \to \mathbb{R}$$

$$V_*(s) = \sup_{\pi} V_\pi(s)$$

greedy policy

$$\pi(s) = \underset{a \in A}{\operatorname{argmax}} \ \mathbb{E}\left[R + \gamma V(s') \mid s, a\right]$$

STATE-ACTION VALUE FUNCTION

$$Q_\pi : S \times A \longrightarrow \mathbb{R}$$

$$Q_\pi(s,a) = \mathbb{E}\left[G \mid S(0) = s \wedge A(0) = a \wedge \pi \right]$$

optimal state-action value function:

$$Q_* : S \times A \longrightarrow \mathbb{R}$$

$$Q_*(s,a) = \sup_\pi Q_\pi(s,a)$$

greedy policy

$$\pi(s) = \underset{a \in A}{\operatorname{argmax}} \qquad Q(s,a)$$

→ FIRST APPROACH : MONTE CARLO

we play the ε-greedy policy:

$$\pi(s) = \begin{cases} \underset{a \in A}{\operatorname{argmax}}\ Q(s,a) & \text{with probability } 1-\varepsilon \\ \text{Uniform} & \text{with probability } \varepsilon \end{cases}$$

REPEAT

sample a trajectory with ε-greedy policy

compute return G_t

$$V(s) \leftarrow V(s) + \alpha\big(G - V(s)\big)$$

$$\text{NEW} = \text{OLD} + \alpha\,[\text{CURRENT} - \text{OLD}]$$

→ SECOND APPROACH: TEMPORAL DIFFERENCE

Key idea: bootstrapping

REPEAT

Sample a step (s, a, r, s') with ε-greedy policy

$$V(s) \leftarrow V(s) + \alpha \left[r + \gamma V(s') - V(s) \right]$$

$$\text{NEW} = \text{OLD} + \alpha \left[\text{CURRENT} - \text{OLD} \right]$$

Key difference:

- Monte Carlo requires a full trajectory before making an update
  BUT smaller variance
- Temporal Difference updates at each step

Q-LEARNING

- Value-based : $Q : S \times A \to \mathbb{R}$
- Temporal Difference
- Off policy : uses one policy for acting
  and another one for updating

REPEAT

Sample a step (s, a, r, s') with ε-greedy policy

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a) \right]$$

$$\text{NEW} = \text{OLD} + \alpha \left[\text{CURRENT} - \text{OLD} \right]$$

SARSA

- Value-based : $Q : S \times A \to \mathbb{R}$
- Temporal Difference
- On-policy : uses one policy for acting and for updating

REPEAT

sample a step (s, a, r, s') with ε-greedy policy

sample a step (s', a', r', s'') with ε-greedy policy

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right]$$

$$\text{NEW} = \text{OLD} + \alpha \left[\text{CURRENT} - \text{OLD} \right]$$

EXPERIENCE REPLAY

running new samples each time is:

* wasteful
* biassed by the current strategy
* not giving the maximum amount of information

EXPERIENCE REPLAY

The buffer has a fixed size

A buffer stores steps (s, a, r, s')

Two actions:

(1) sample a trajectory and add each step to the buffer (independently!)

(2) sample from the buffer to update

PRIORITISED EXPERIENCE REPLAY

- when we add (s, a, r, s') we compute the bias

$$B = r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a)$$

to obtain a distribution we apply a softmax:

$$\exp(B) \Big/ \sum_{B'} \exp(B')$$

- we sample from the buffer with this distribution

DOUBLE Q-LEARNING

Issue:

$$Q(s,a) = Q(s,a) + \alpha \left(r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a) \right)$$

$\rightarrow$ tends to overshoot

THE MAXIMISATION BIAS

$X_1, \dots, X_n$ random variables

Goal: estimate $\max_i \mathbb{E}[X_i]$

single estimator:

for each i: Sample X_i a number of times

→ choose $\operatorname{argmax}_i \hat{\mathbb{E}}[X_i] = *$

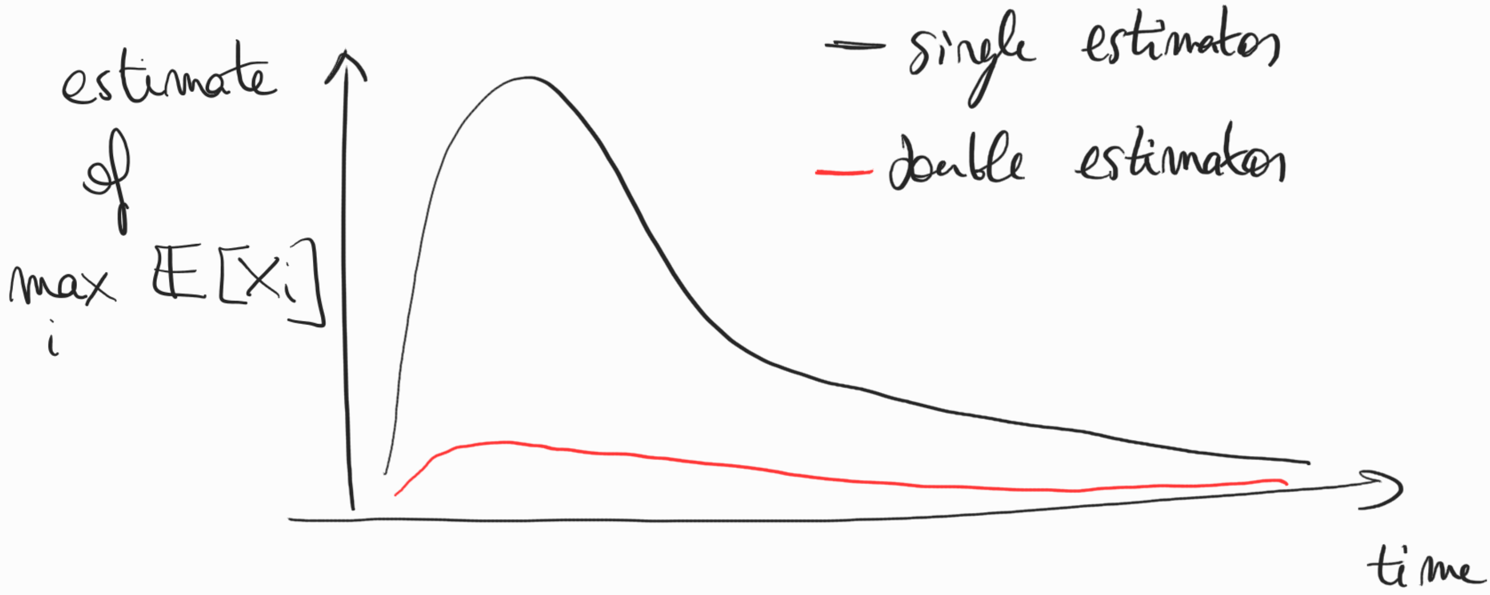
→ evaluate by $\hat{\mathbb{E}}[X_*]$

double estimator:

for each i: Sample X_i a number of times, twice

→ choose $\operatorname{argmax}_i \hat{\mathbb{E}}[X_i] = *$ with first set

→ evaluate by $\hat{\mathbb{E}}[X_*]$ with second set


single estimator
double estimator
estimate of $\max_i \mathbb{E}[X_i]$
time

DOUBLE Q-LEARNING ALGORITHM

with probability 1/2 :

- $a' = \underset{a'}{\operatorname{argmax}}\, Q_2(s', a')$
- $Q_1(s,a) \leftarrow Q_1(s,a) + \alpha\left(r + \gamma\, Q_1(s', a') - Q_1(s,a)\right)$

else :

symmetrically