

RL @ PicsArt

Day 1, part 2

Value-based methods



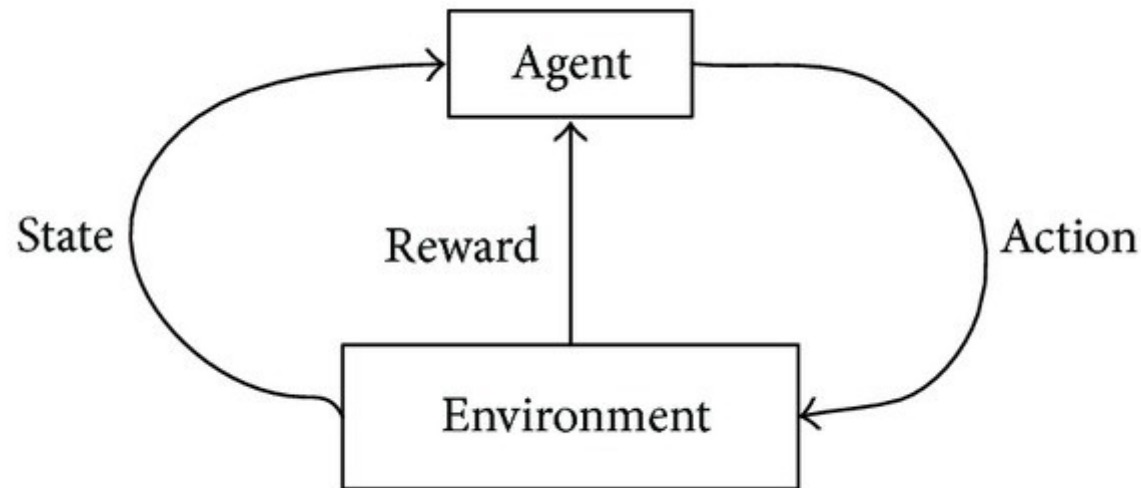
Yandex
Data Factory

LAMBDA 



**British Hedgehog
Preservation Society**

Recap: Discounted reward MDP

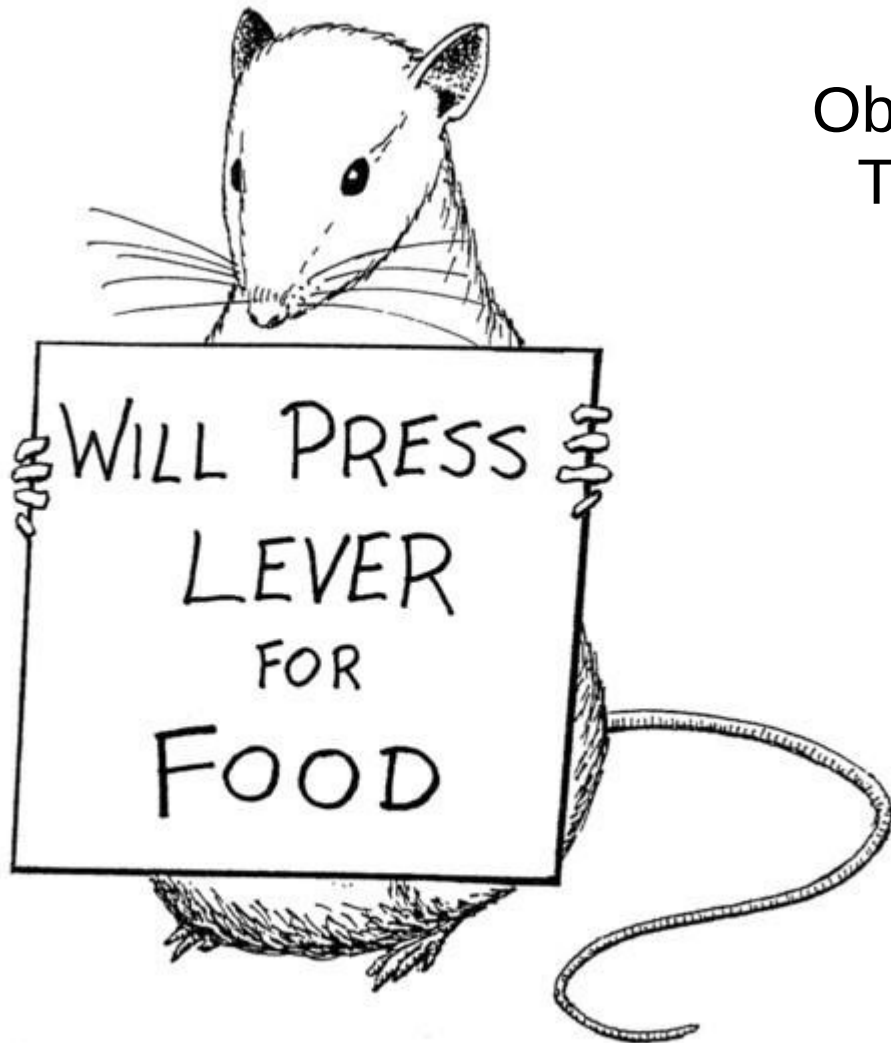


Classic MDP(Markov Decision Process)

Agent interacts with environment

- Environment states: $s \in S$
- Agent actions: $a \in A$
- State transition: $P(s_{t+1}|s_t, a_t)$
- Reward: $r_t = r(s_t, a_t)$

Recap: Discounted reward MDP



Objective:
Total action value

$$G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \dots + \gamma^n \cdot r_{t+n}$$

$$G_t = \sum_i \gamma^i \cdot r_{t+i} \quad \gamma \in (0,1) \text{ const}$$

$\gamma \sim$ patience

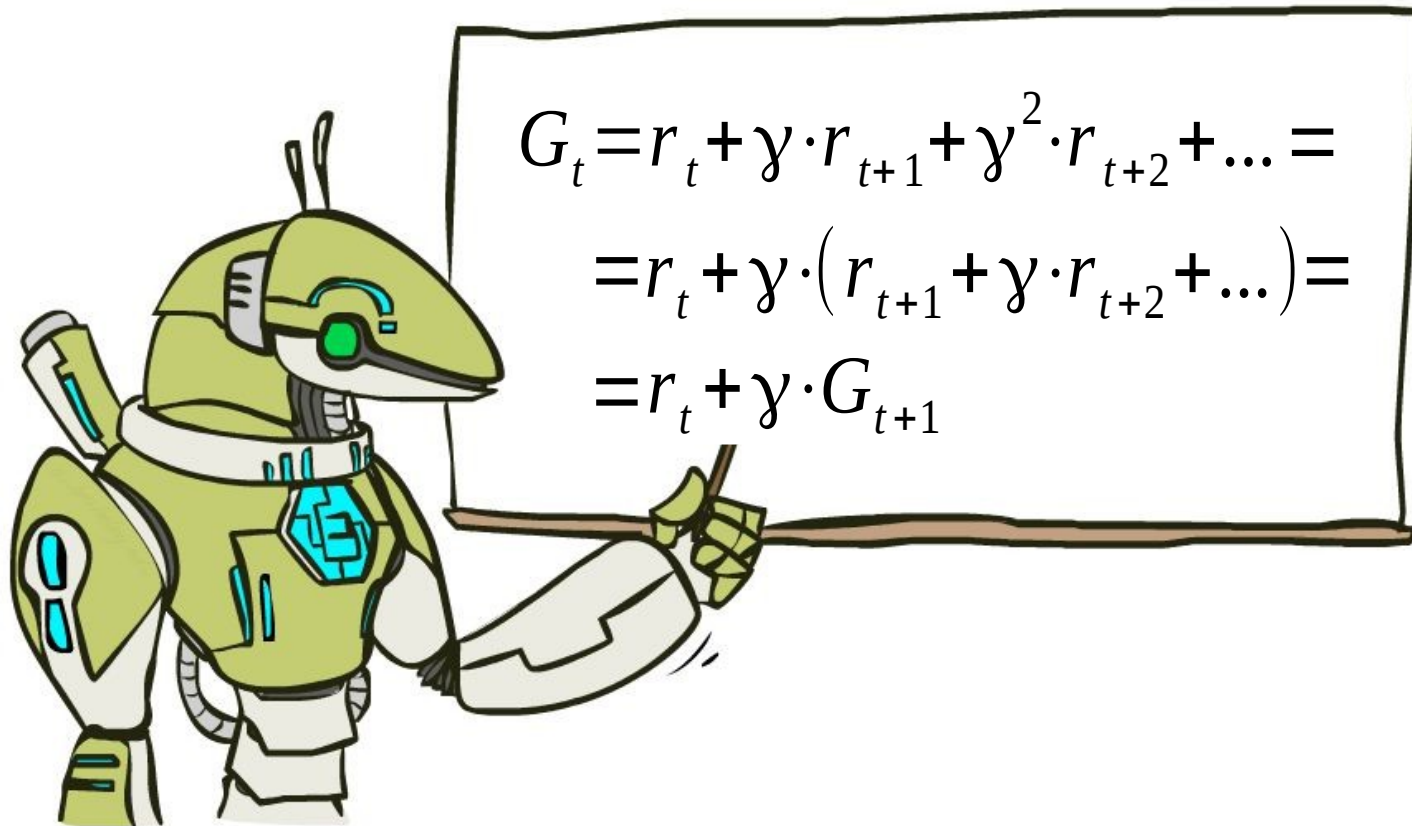
Cake tomorrow is γ as good as now

Reinforcement learning:

- Find policy that maximizes the expected reward

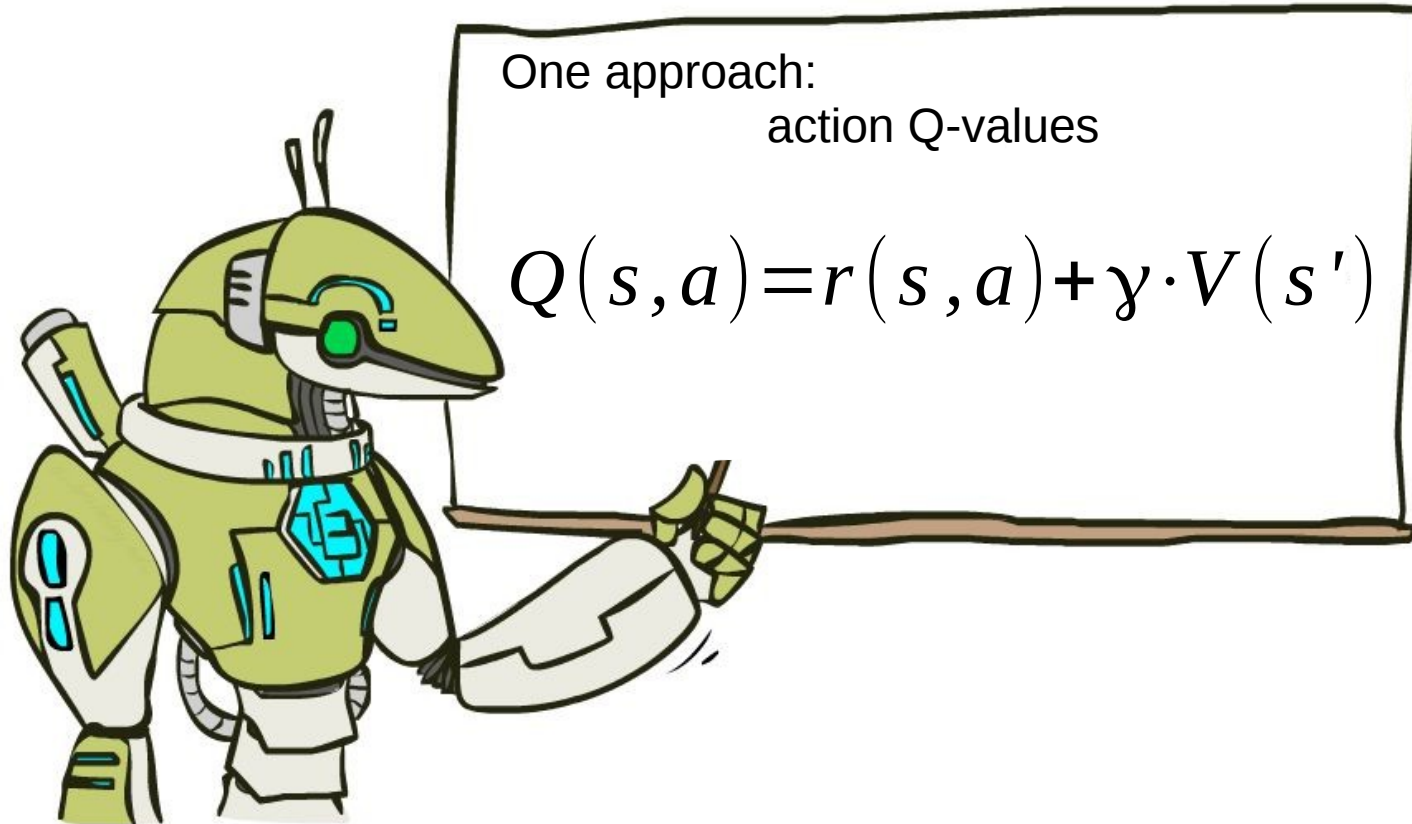
$$\pi = P(a|s) : E[G] \rightarrow \max$$

Recap: Optimal policy



We rewrite G with sheer power of math!

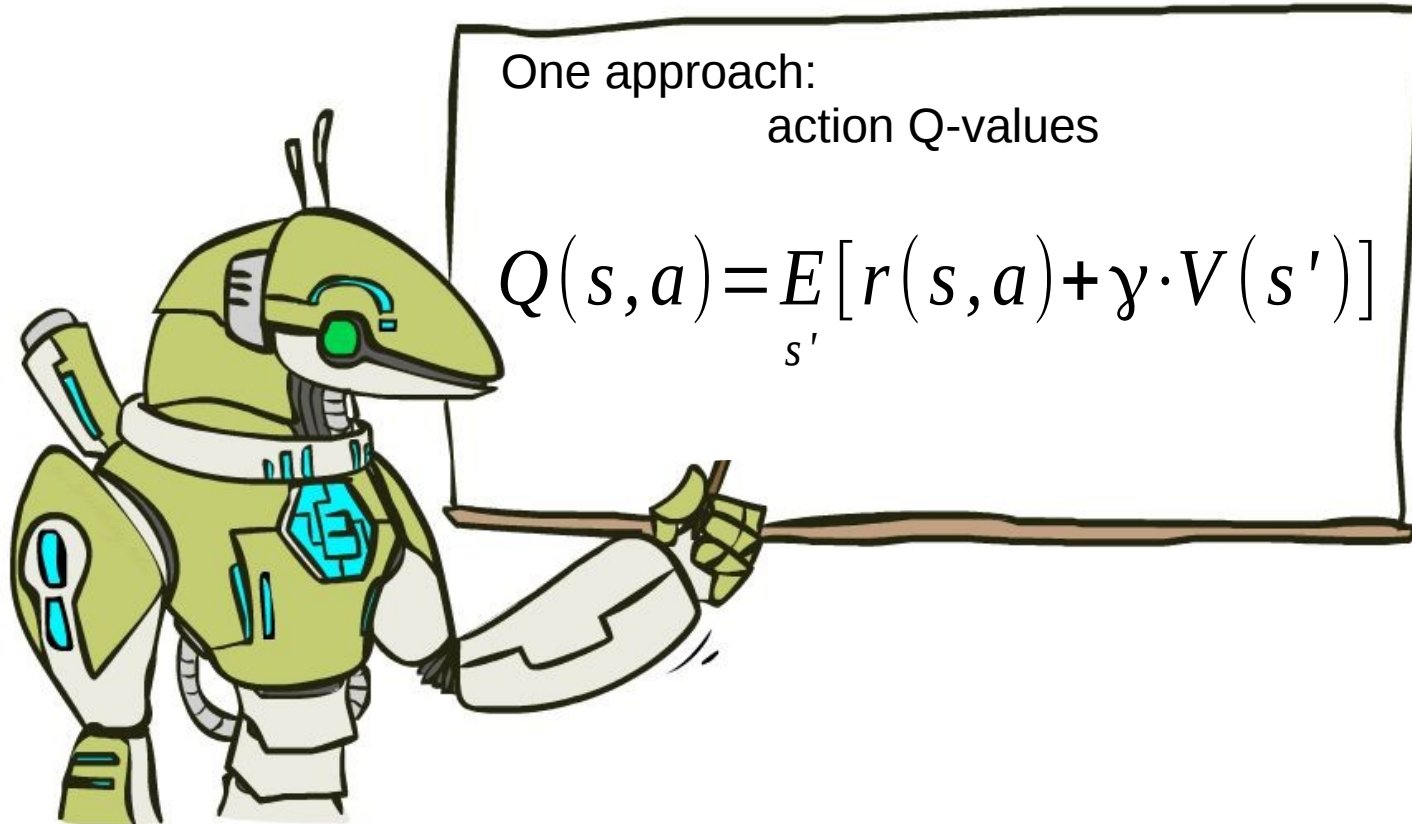
Recap: Q-learning



Action value $Q(s, a)$ is the expected total reward **G** agent gets from state **s** by taking action **a** and following policy π from next state.

Trivia: how do we get policy $\pi(a|s)$ given $Q(s, a)$?

Recap: Q-learning



Action value $Q(s, a)$ is the expected total reward **G** agent gets from state **s** by taking action **a** and following policy **π** from next state.

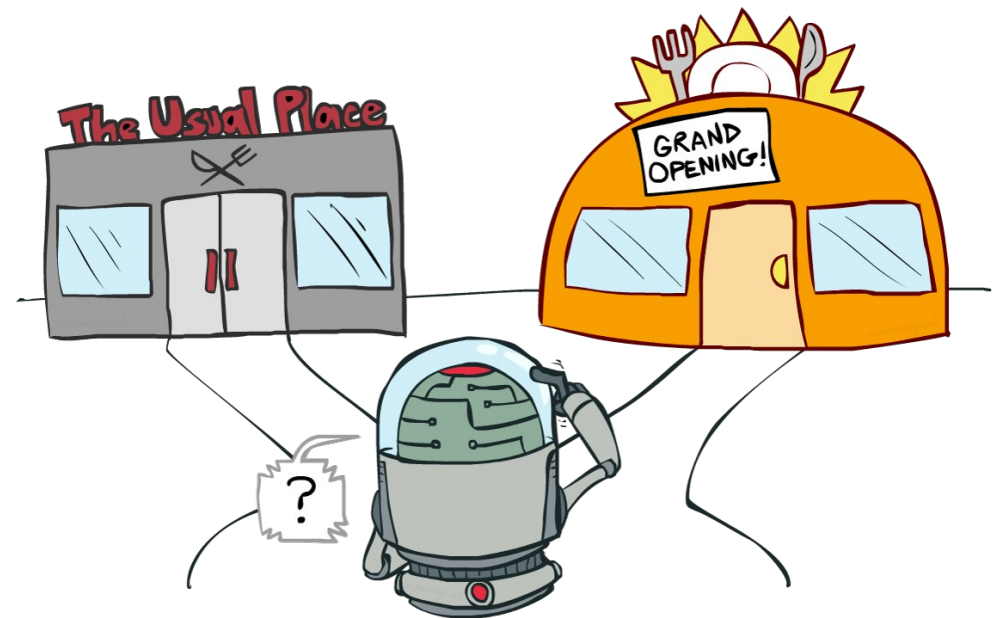
$$\pi(s) : \operatorname{argmax}_a Q(s, a)$$

Recap: Exploration Vs Exploitation

Balance between using what you learned and trying to find something even better

ϵ -greedy

- With probability ϵ take random action;
- Otherwise take optimal action.



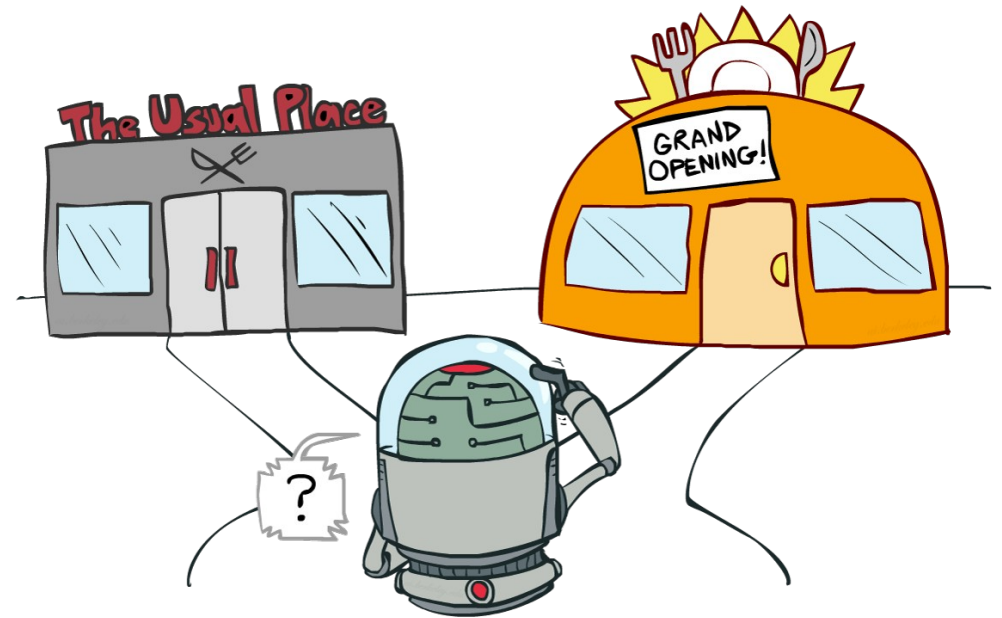
Trivia: how to define $\pi(a|s)$ now?

Exploration Vs Exploitation

Balance between using what you learned and trying to find something even better

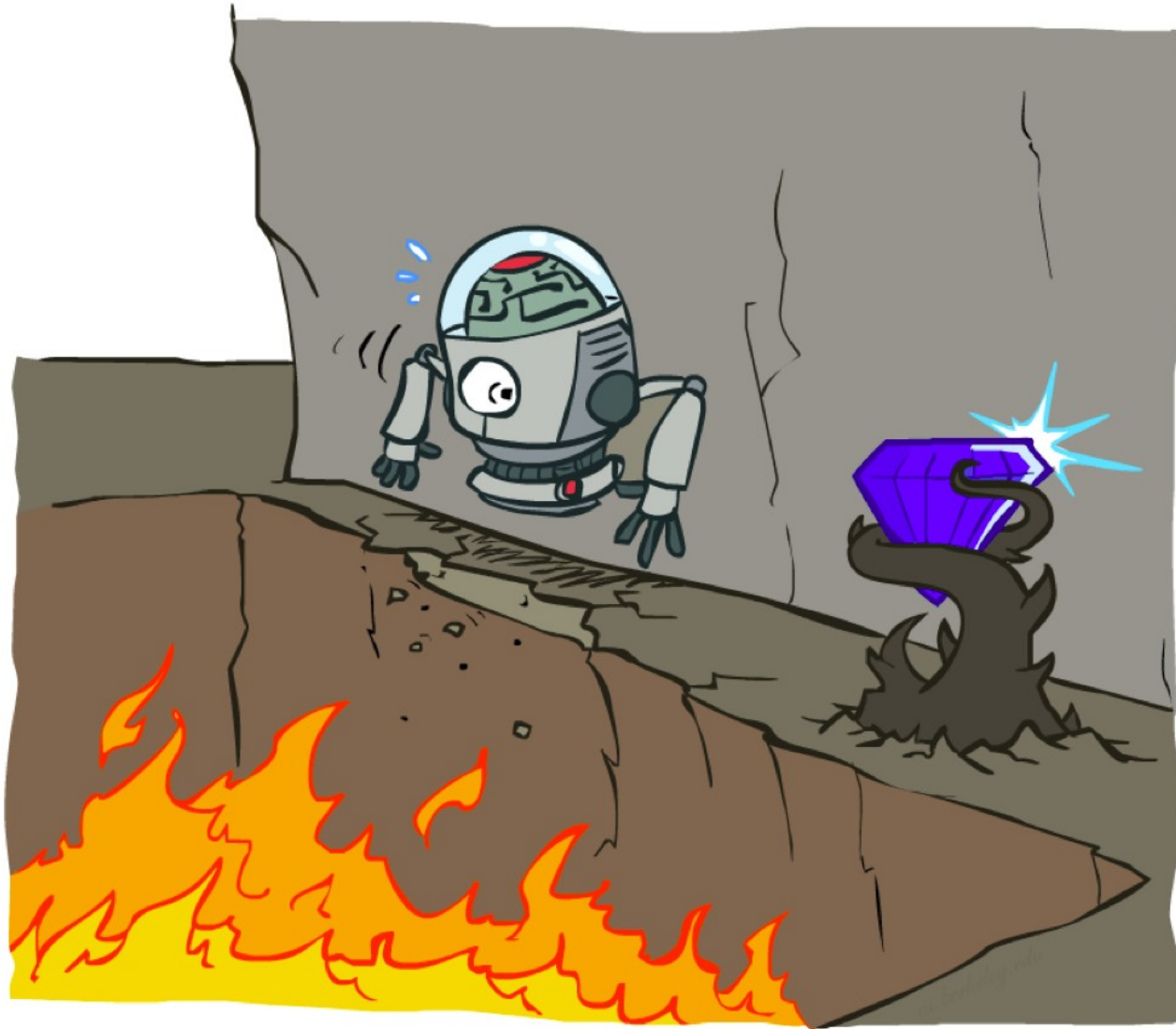
ϵ -greedy

- With probability ϵ take random action;
- Otherwise take optimal action.



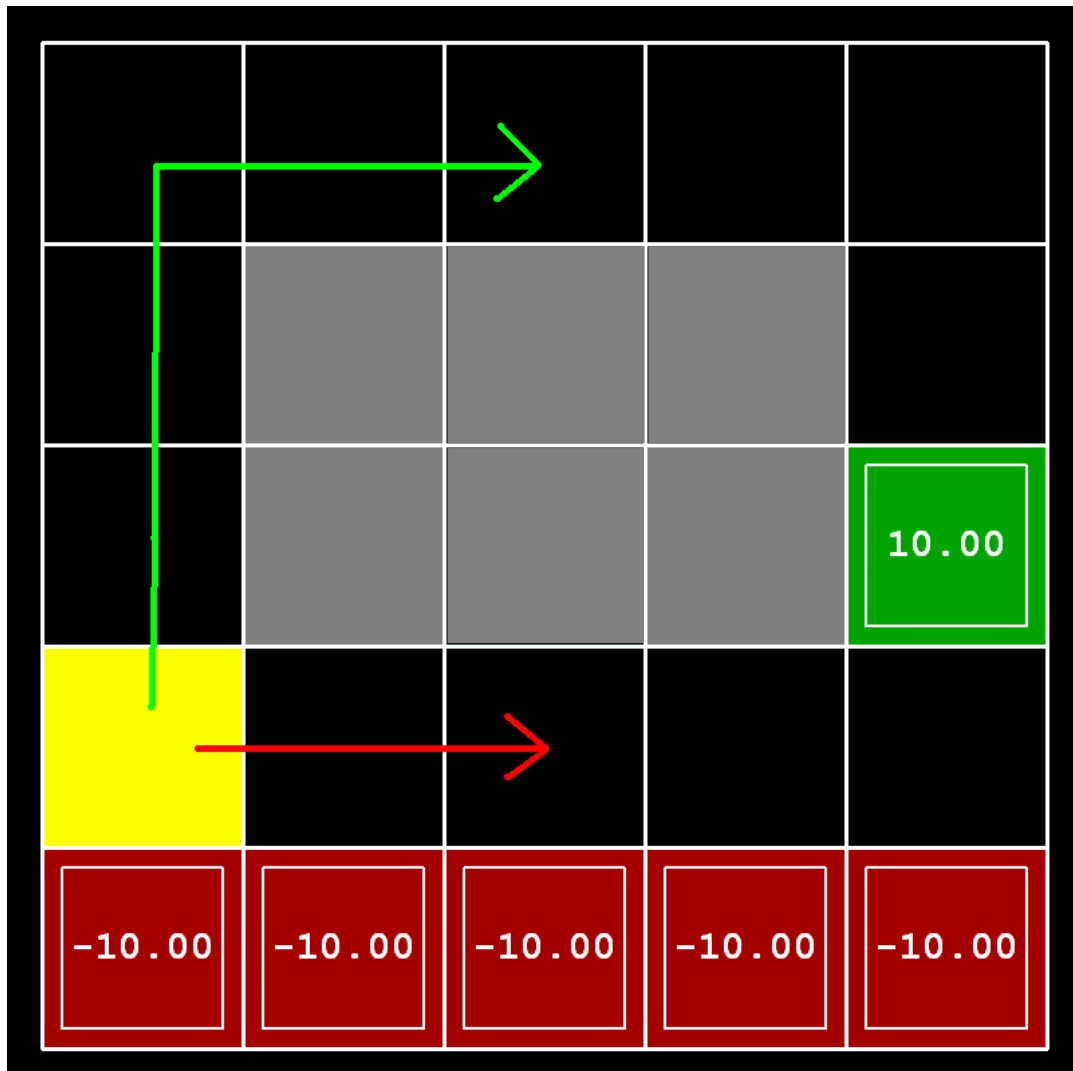
$$\pi(a|s) : (1 - \epsilon)[a = \operatorname{argmax}_a Q(s, a)] + \frac{\epsilon}{|A|}$$

Cliff world



Picture from Berkeley CS188x

Cliff world



Conditions

- Q-learning

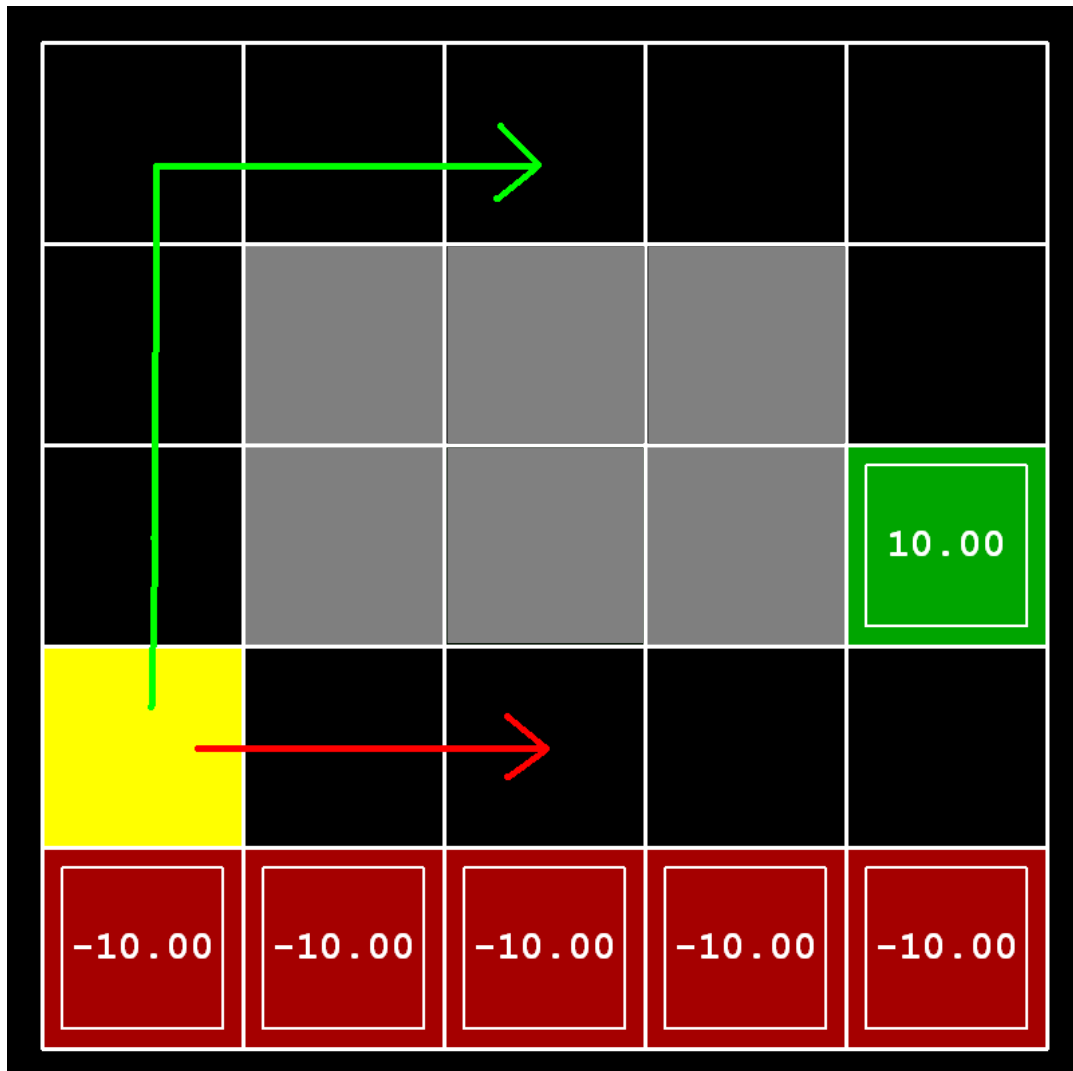
$$\gamma = 0.99 \quad \epsilon = 0.1$$

- no slipping

Trivia:

What will q-learning learn?

Cliff world



Conditions

- Q-learning

$$\gamma = 0.99 \quad \epsilon = 0.1$$

- no slipping

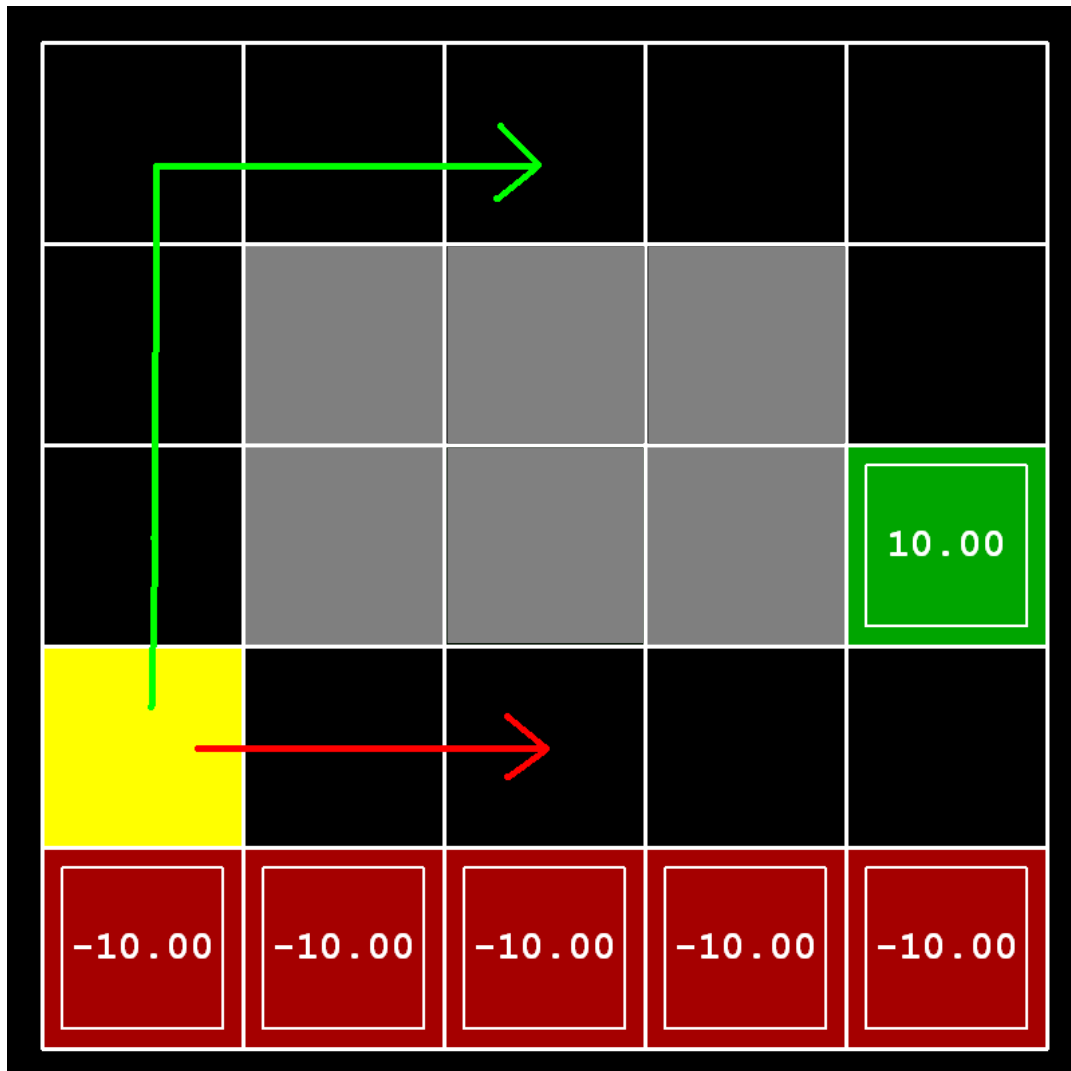
Trivia:

What will q-learning learn?

follow the short path

Will it maximize reward?

Cliff world



Conditions

- Q-learning

$$\gamma = 0.99 \quad \epsilon = 0.1$$

- no slipping

Trivia:

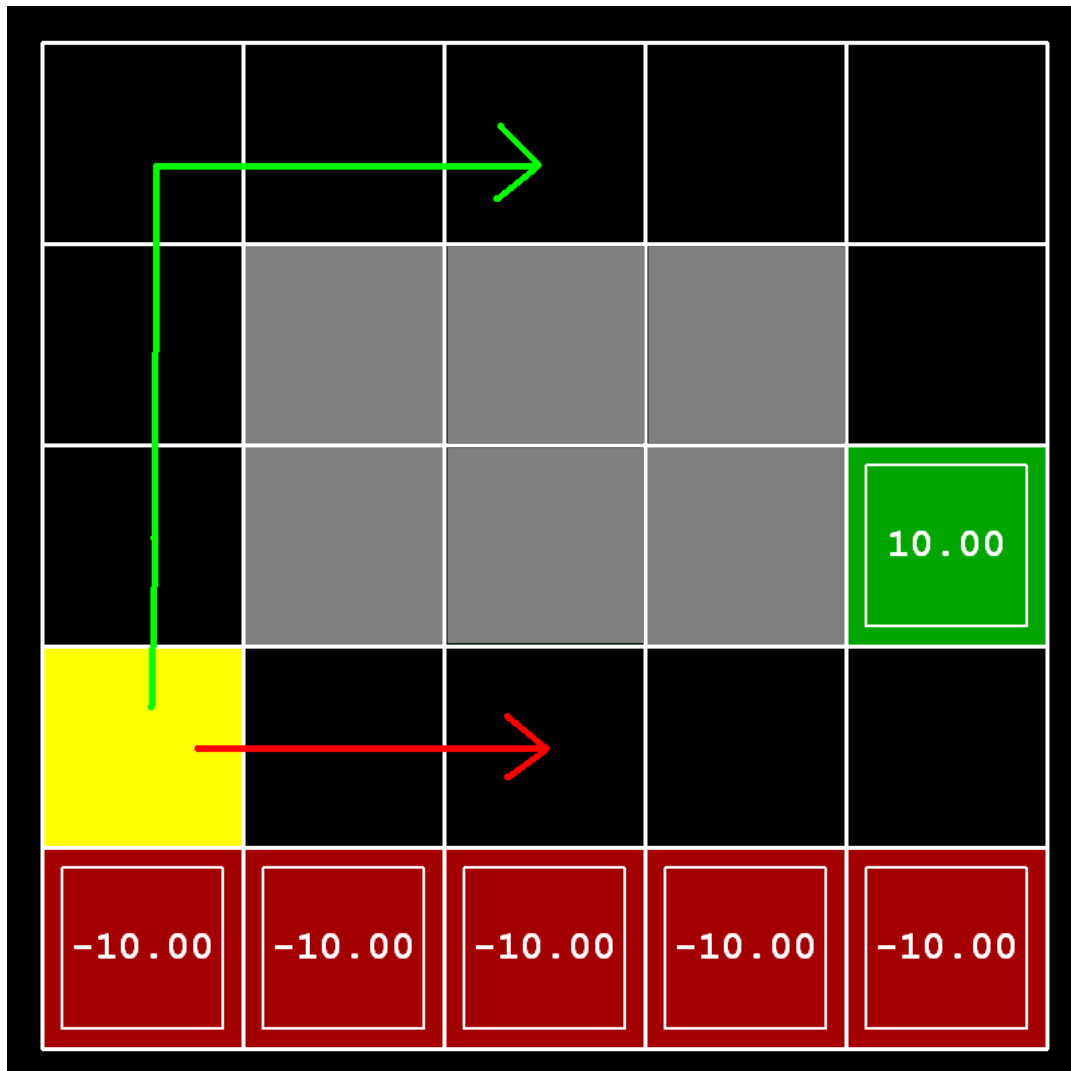
What will q-learning learn?

follow the short path

Will it maximize reward?

**no, robot will fall due to
epsilon-greedy “exploration”**

Cliff world



Conditions

- Q-learning

$$\gamma = 0.99 \quad \epsilon = 0.1$$

- no slipping

**Decisions must account
for actual policy!**

e.g. ϵ -greedy policy

Generalized value iteration

Update rule (from Bellman eq.)

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

 “better $Q(s,a)$ ”

Q-learning VS SARSA

Update rule (from Bellman eq.)

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

Q-learning

“better $Q(s, a)$ ”



$$\hat{Q}(s, a) = r(s, a) + \gamma \cdot \max_{a'} Q(s', a')$$

Q-learning VS SARSA

Update rule (from Bellman eq.)

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

Q-learning

“better $Q(s, a)$ ”



$$\hat{Q}(s, a) = r(s, a) + \gamma \cdot \max_{a'} Q(s', a')$$

SARSA

$$\hat{Q}(s, a) = r(s, a) + \gamma \cdot E_{a' \sim \pi(a'|s')} Q(s', a')$$

Q-learning VS SARSA

Update rule (from Bellman eq.)

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

Q-learning

“better $Q(s, a)$ ”



$$\hat{Q}(s, a) = r(s, a) + \gamma \cdot \max_{a'} \underbrace{Q(s', a')}$$

SARSA

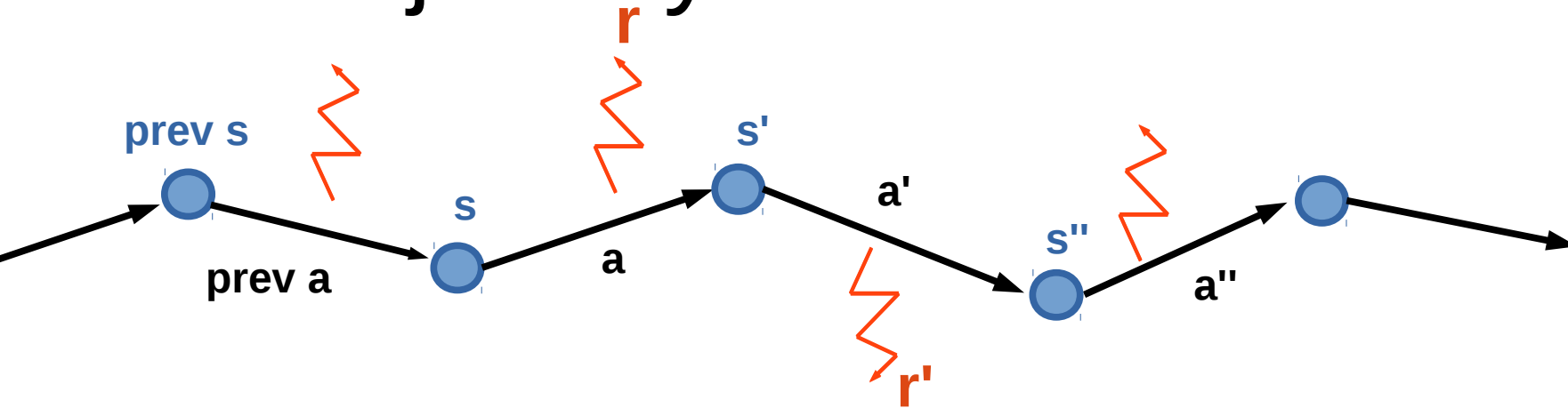
Expected from

$s' \sim P(s'|s, a)$



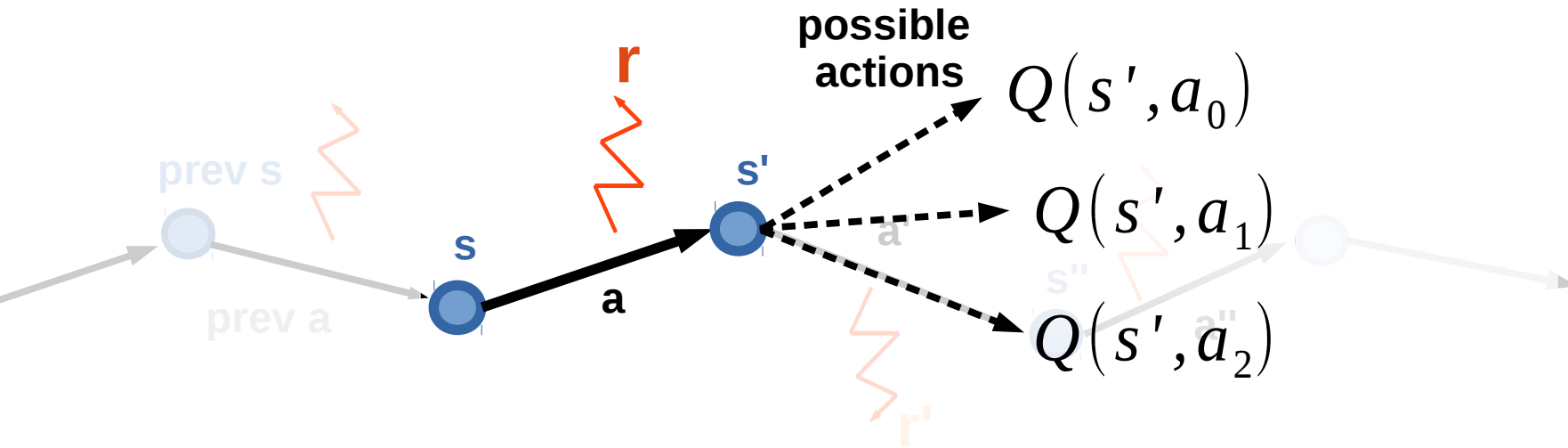
$$\hat{Q}(s, a) = r(s, a) + \gamma \cdot \underset{a' \sim \pi(a'|s')}{E} \underbrace{Q(s', a')}$$

MDP trajectory



- sample sequence of
 - states (s)
 - actions (a)
 - rewards (r)
- Can be infinite, we can't wait that long

Recap: Q-learning



$$\forall s \in S, \forall a \in A, Q(s, a) \leftarrow 0$$

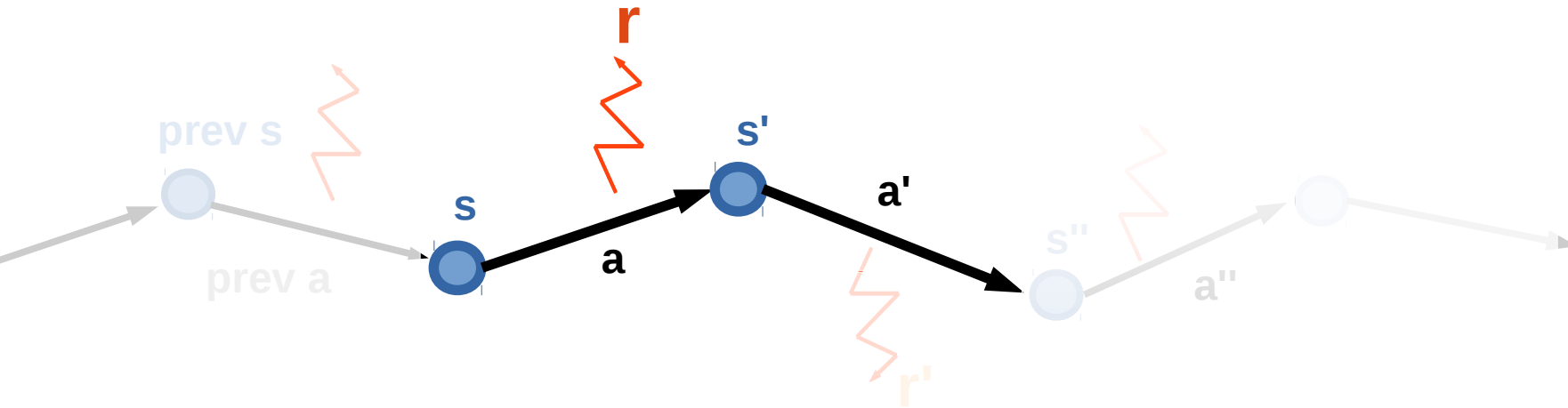
Loop:

- Sample $\langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}' \rangle$ from env

- Compute $\hat{Q}(s, a) = r(s, a) + \gamma \max_{a_i} Q(s', a_i)$

- Update $Q(s, a) \leftarrow \alpha \cdot \hat{Q}(s, a) + (1 - \alpha) Q(s, a)$

SARSA



$$\forall s \in S, \forall a \in A, Q(s, a) \leftarrow 0$$

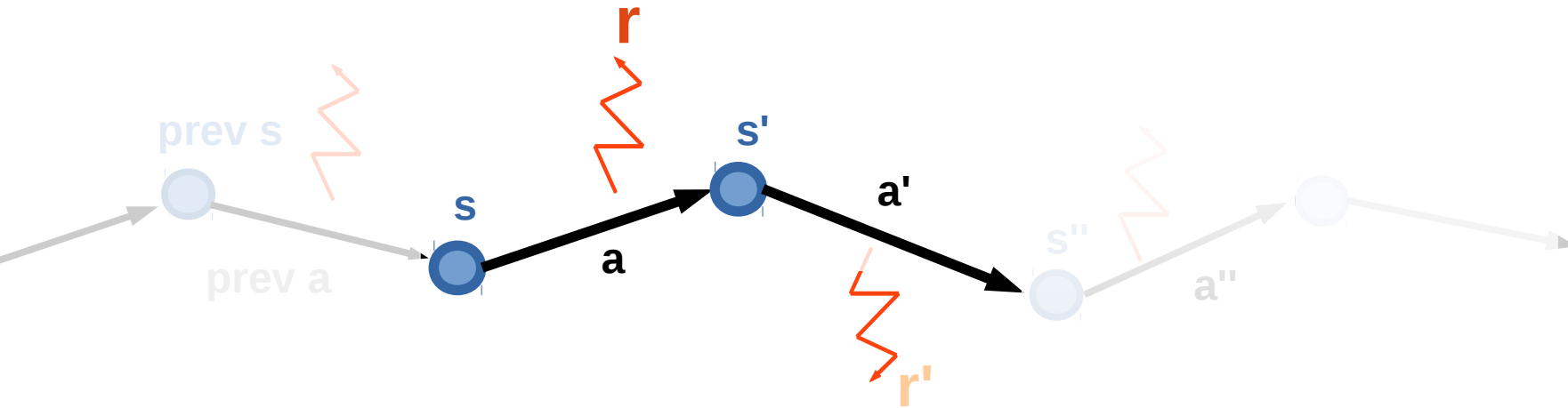
Loop:

- Sample $\langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}', \mathbf{a}' \rangle$ from env

- Compute $\hat{Q}(s, a) = r(s, a) + \gamma Q(s', a')$

- Update $Q(s, a) \leftarrow \alpha \cdot \hat{Q}(s, a) + (1 - \alpha) Q(s, a)$

SARSA



$$\forall s \in S, \forall a \in A, Q(s, a) \leftarrow 0$$

Loop:

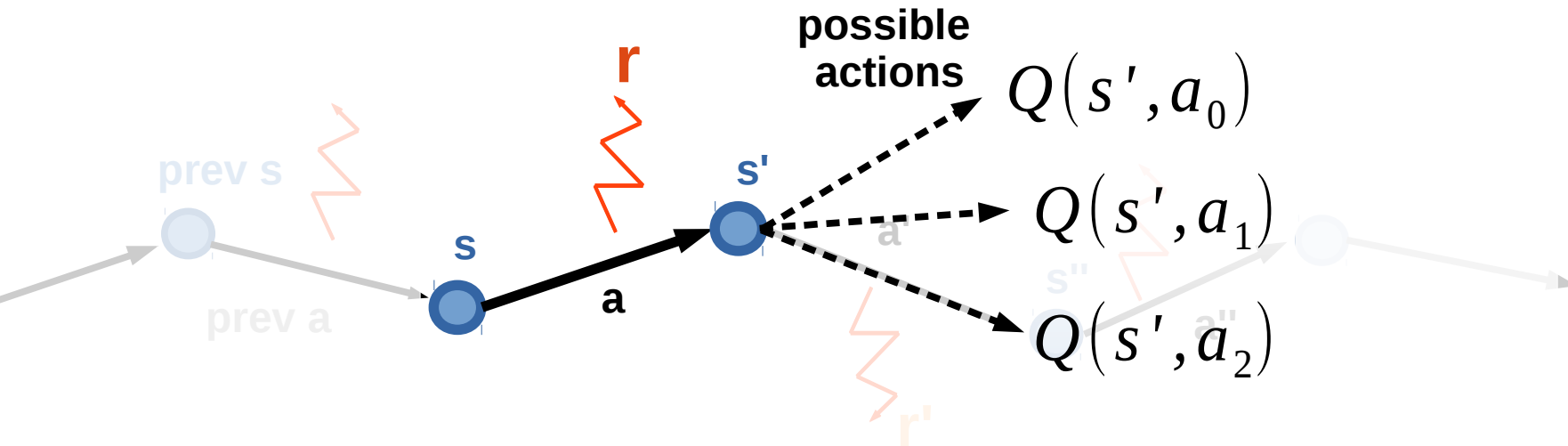
– Sample $\langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}', \mathbf{a}' \rangle$ from env

hence “SARSA”

– Compute $\hat{Q}(s, a) = r(s, a) + \gamma Q(\underline{s'}, a')$ **next action (not max)**

– Update $Q(s, a) \leftarrow \alpha \cdot \hat{Q}(s, a) + (1 - \alpha) Q(s, a)$

Expected value SARSA



$$\forall s \in S, \forall a \in A, Q(s, a) \leftarrow 0$$

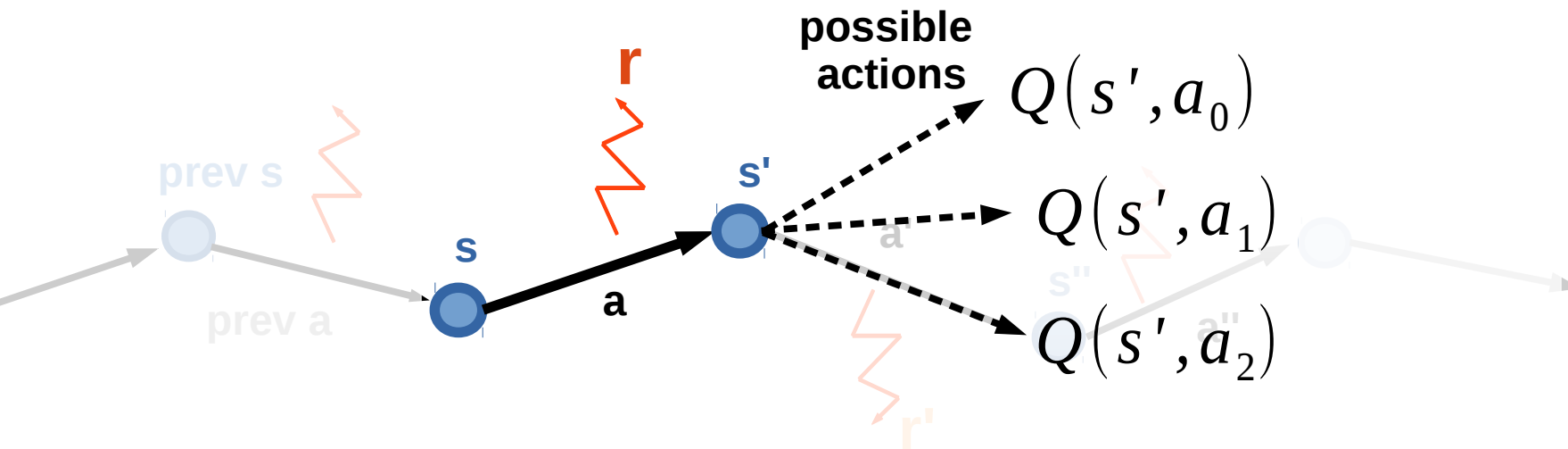
Loop:

- Sample $\langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}' \rangle$ from env

- Compute $\hat{Q}(s, a) = r(s, a) + \gamma \mathop{E}_{a_i \sim \pi(a|s')} Q(s', a_i)$

- Update $Q(s, a) \leftarrow \alpha \cdot \hat{Q}(s, a) + (1 - \alpha) Q(s, a)$

Expected value SARSA



$$\forall s \in S, \forall a \in A, Q(s, a) \leftarrow 0$$

Loop:

- Sample $\langle \mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}' \rangle$ from env

- Compute $\hat{Q}(s, a) = r(s, a) + \gamma \underset{a_i \sim \pi(a|s')}{E} Q(s', a_i)$

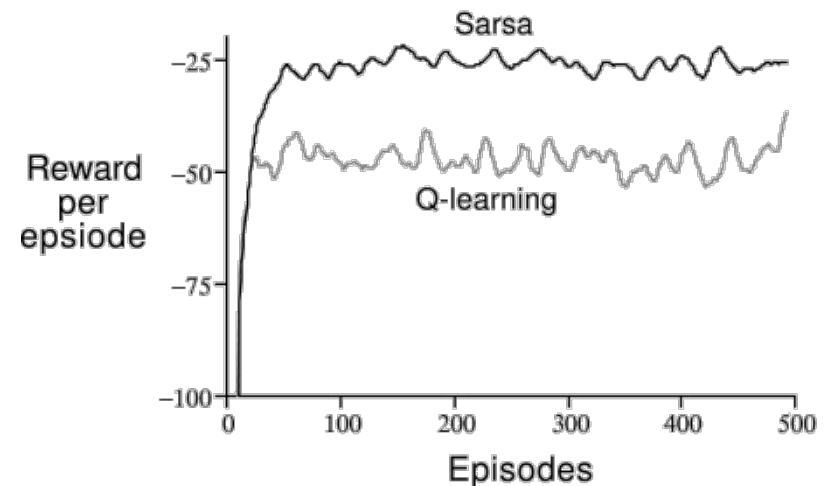
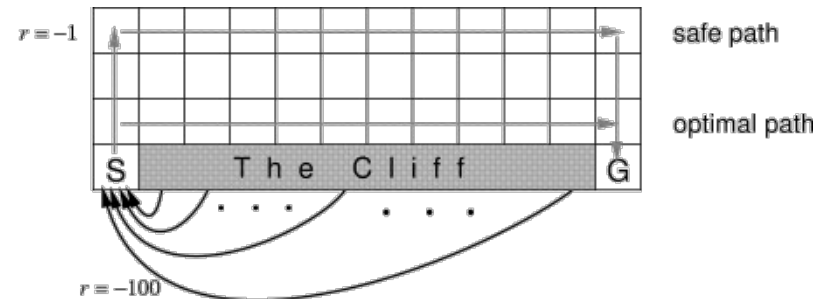
Expected value



- Update $Q(s, a) \leftarrow \alpha \cdot \hat{Q}(s, a) + (1 - \alpha) Q(s, a)$

Difference

- SARSA converges to optimal policy
- Q-learning policy **would be** optimal without exploration



On-policy vs Off-policy

Two problem setups

on-policy

Agent **can** pick actions

- Most obvious setup :)
- Agent always follows his **own** policy

off-policy

Agent **can't** pick actions

- Learning with exploration,
playing without exploration
- Learning from expert
(expert is imperfect)
- Learning from sessions
(recorded data)

On-policy vs Off-policy

Two problem setups

on-policy

Agent **can** pick actions

- On-policy algorithms **can't** learn off-policy

(but they be faster/better)

off-policy

Agent **can't** pick actions

- Off-policy algorithms **can** learn on-policy

learn optimal policy even if agent takes random actions

Trivia: which of Q-learning, SARSA and exp. val. SARSA will **only** work on-policy?

On-policy vs Off-policy

Two problem setups

on-policy

Agent **can** pick actions

- On-policy algorithms **can't** learn off-policy
- SARSA
- more coming soon

off-policy

Agent **can't** pick actions

- Off-policy algorithms **can** learn on-policy
- Q-learning
- Expected Value SARSA

Trivia: will Crossentropy Method converge if it learns off-policy from agent that takes random actions?

On-policy vs Off-policy

Two problem setups

on-policy

Agent **can** pick actions

- On-policy algorithms **can't** learn off-policy
- SARSA
- more coming soon

off-policy

Agent **can't** pick actions

- Off-policy algorithms **can** learn on-policy
- Q-learning
- Expected Value SARSA

Trivia: will Crossentropy Method converge if it learns off-policy from agent that takes random actions? **Well, no :)**

On-policy vs Off-policy

Two problem setups

on-policy

Agent **can** pick actions

- On-policy algorithms **can't** learn off-policy
- SARSA
- more coming soon

off-policy

Agent **can't** pick actions

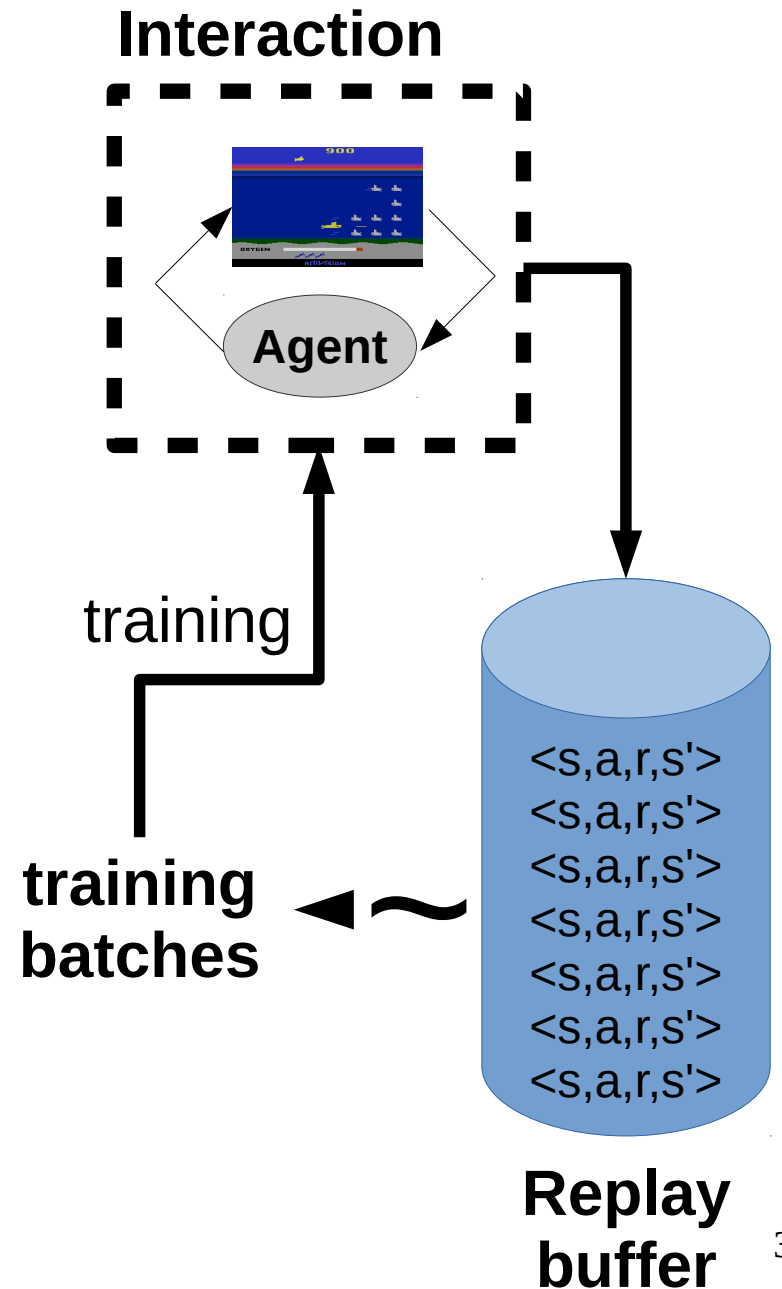
- Off-policy algorithms **can** learn on-policy
- Q-learning
- Expected Value SARSA

Experience replay

Idea: store several past interactions

$\langle s, a, r, s' \rangle$

Train on random subsamples



Experience replay

Idea: store several past interactions
 $\langle s, a, r, s' \rangle$
Train on random subsamples

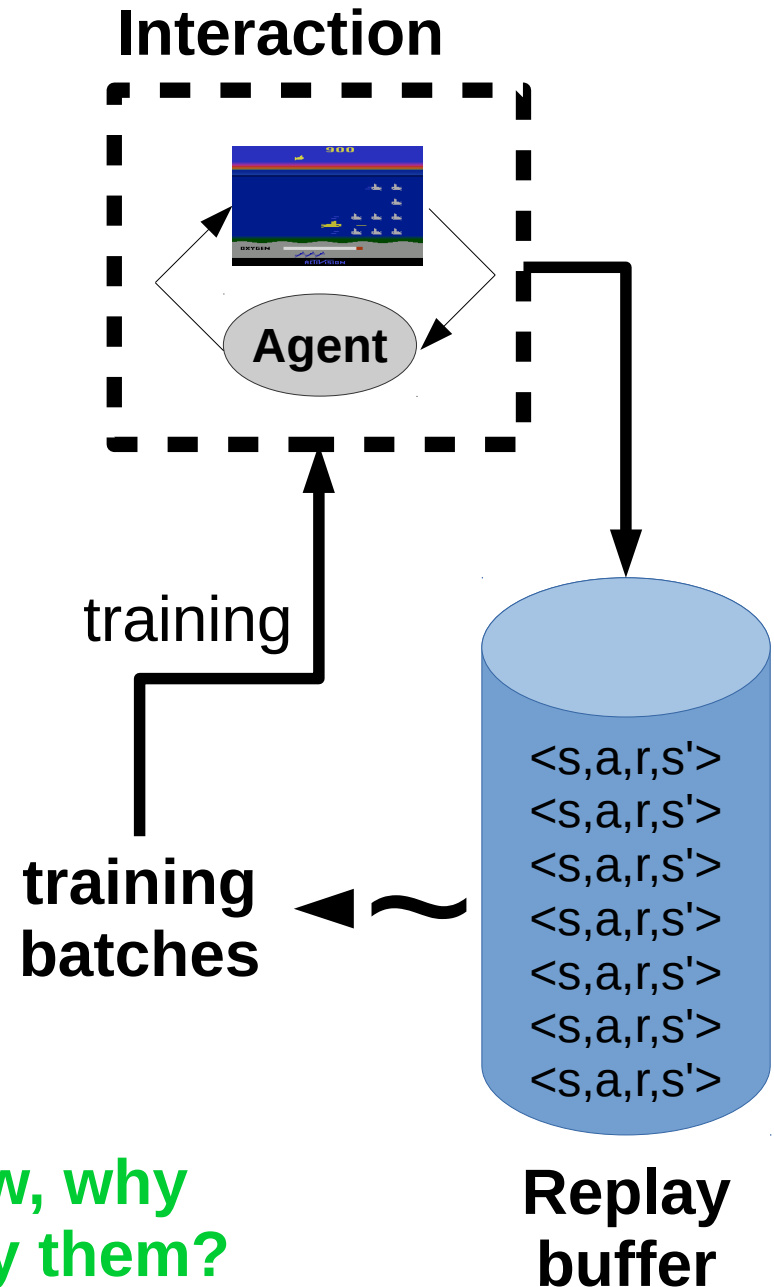
Training curriculum:

- play 1 step and record it
- pick N random transitions to train

Profit: you don't need to re-visit same
(s,a) many times to learn it.

**Only works with
off-policy algorithms!**

**Btw, why
only them?**



Experience replay

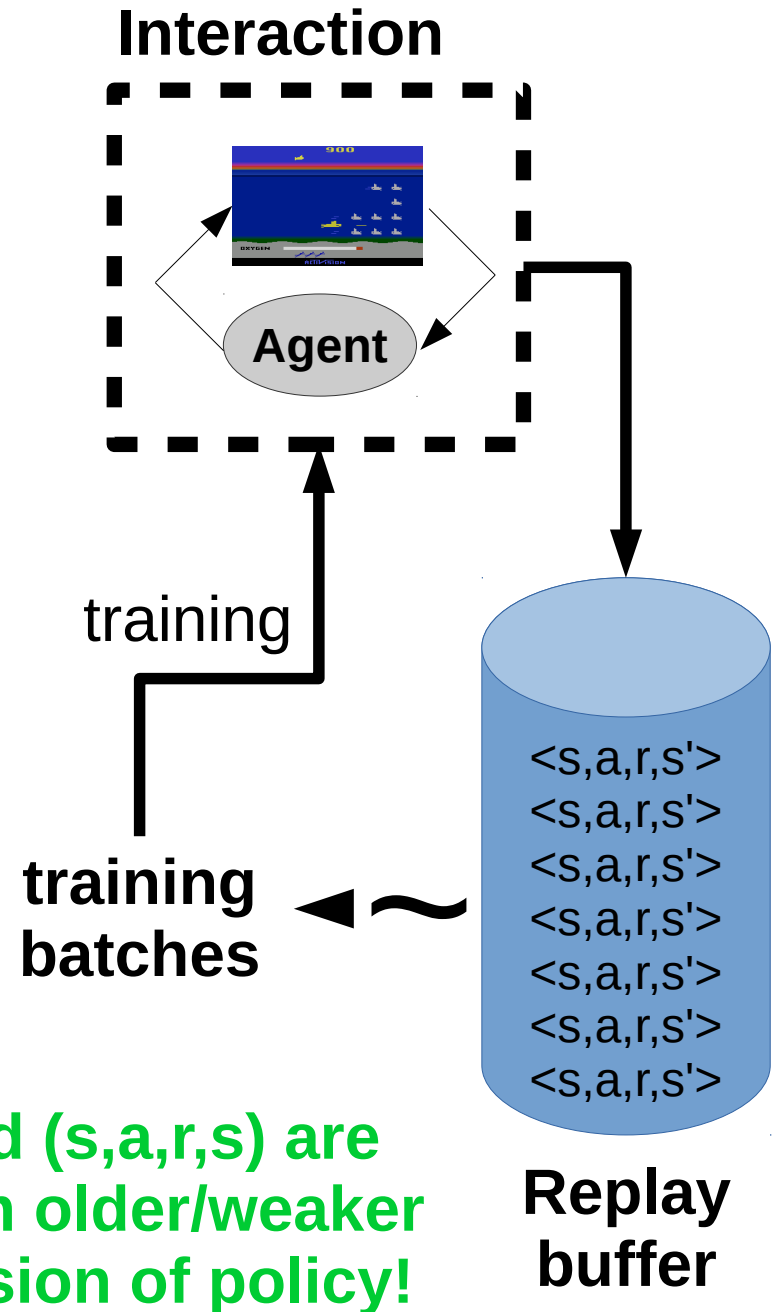
Idea: store several past interactions
 $\langle s, a, r, s' \rangle$
Train on random subsamples

Training curriculum:

- play 1 step and record it
- pick N random transitions to train

Profit: you don't need to re-visit same
(s,a) many times to learn it.

**Only works with
off-policy algorithms!**



N-step algorithms

Recall R?

$$\begin{aligned} G_t &= r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \dots = \\ &= r_t + \gamma \cdot (r_{t+1} + \gamma \cdot r_{t+2} + \dots) = \\ &= r_t + \gamma \cdot G_{t+1} = \\ &= r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot G_{t+2} \end{aligned}$$

N-step SARSA

- General formula

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})$$

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 Q(s_{t+2}, a_{t+2})$$

N-step SARSA

- General formula

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

1-step SARSA

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})$$

2-step SARSA

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 Q(s_{t+2}, a_{t+2})$$

3-step SARSA

$$\hat{Q}(s_t, a_t) = ???$$

N-step SARSA

- General formula

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

1-step SARSA

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})$$

2-step SARSA

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 Q(s_{t+2}, a_{t+2})$$

3-step SARSA

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \gamma^3 Q(s_{t+3}, a_{t+3})$$

N-step SARSA

- General formula

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

1-step SARSA

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})$$

N-step SARSA

$$\hat{Q}(s_t, a_t) = \left[\sum_{\tau=t}^{\tau=t+n} \gamma^{\tau-t} r(s_{t+\tau}, a_{t+\tau}) \right] + \gamma^n Q(s_{t+n}, a_{t+n})$$

N-step algorithms

- General formula

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

N-step SARSA

$$\hat{Q}(s_t, a_t) = \left[\sum_{\tau=t}^{\tau < t+n} \gamma^\tau r(s_{t+\tau}, a_{t+\tau}) \right] + \gamma^n Q(s_{t+n}, a_{t+n})$$

N-step Q-learning

$$\hat{Q}(s_t, a_t) = \left[\sum_{\tau=t}^{\tau < t+n} \gamma^\tau r(s_{t+\tau}, a_{t+\tau}) \right] + \gamma^n \cdot \max_a Q(s_{t+n}, a)$$

Trivia: which of these methods works off-policy?

N-step algorithms

- General formula

$$Q(s_t, a_t) \leftarrow \alpha \cdot \hat{Q}(s_t, a_t) + (1 - \alpha) Q(s_t, a_t)$$

N-step SARSA

$$\hat{Q}(s_t, a_t) = \left[\sum_{\tau=t}^{\tau=t+n} \gamma^\tau r(s_{t+\tau}, a_{t+\tau}) \right] + \gamma^n Q(s_{t+n}, a_{t+n})$$

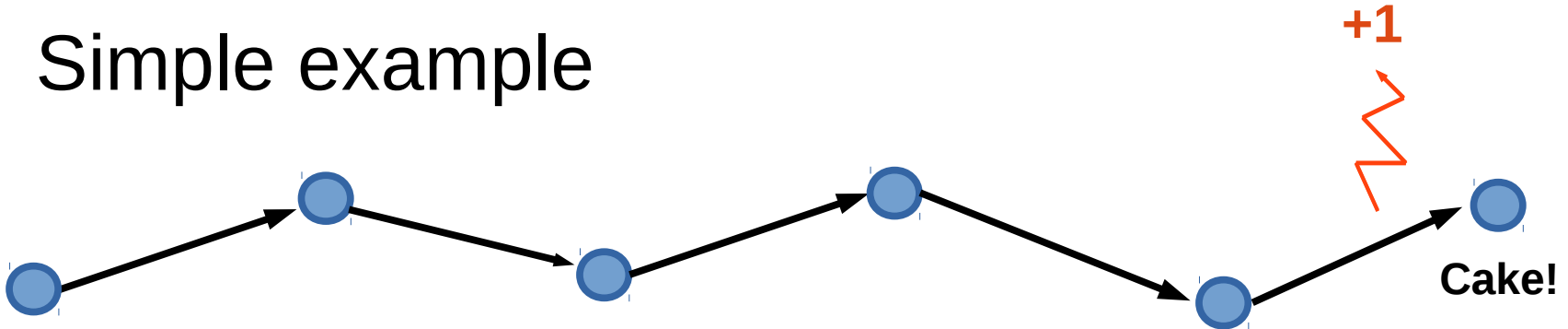
N-step Q-learning

$$\hat{Q}(s_t, a_t) = \left[\sum_{\tau=t}^{\tau=t+n} \gamma^\tau r(s_{t+\tau}, a_{t+\tau}) \right] + \gamma^n \cdot \max_a Q(s_{t+n}, a)$$

Trivia: which of these methods works off-policy? **None of them!**

1-step Vs n-step

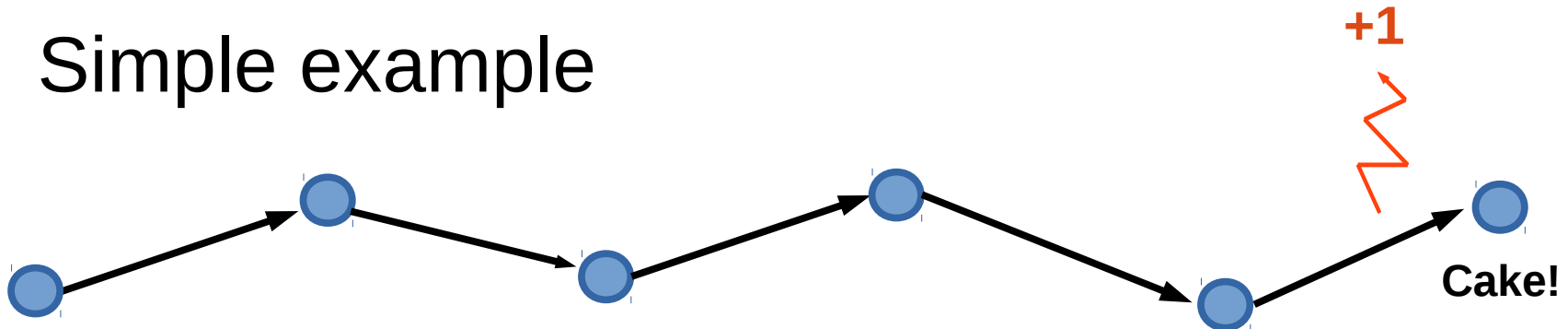
- Simple example



How many games does it take for **SARSA** to learn the optimal policy?

1-step Vs n-step

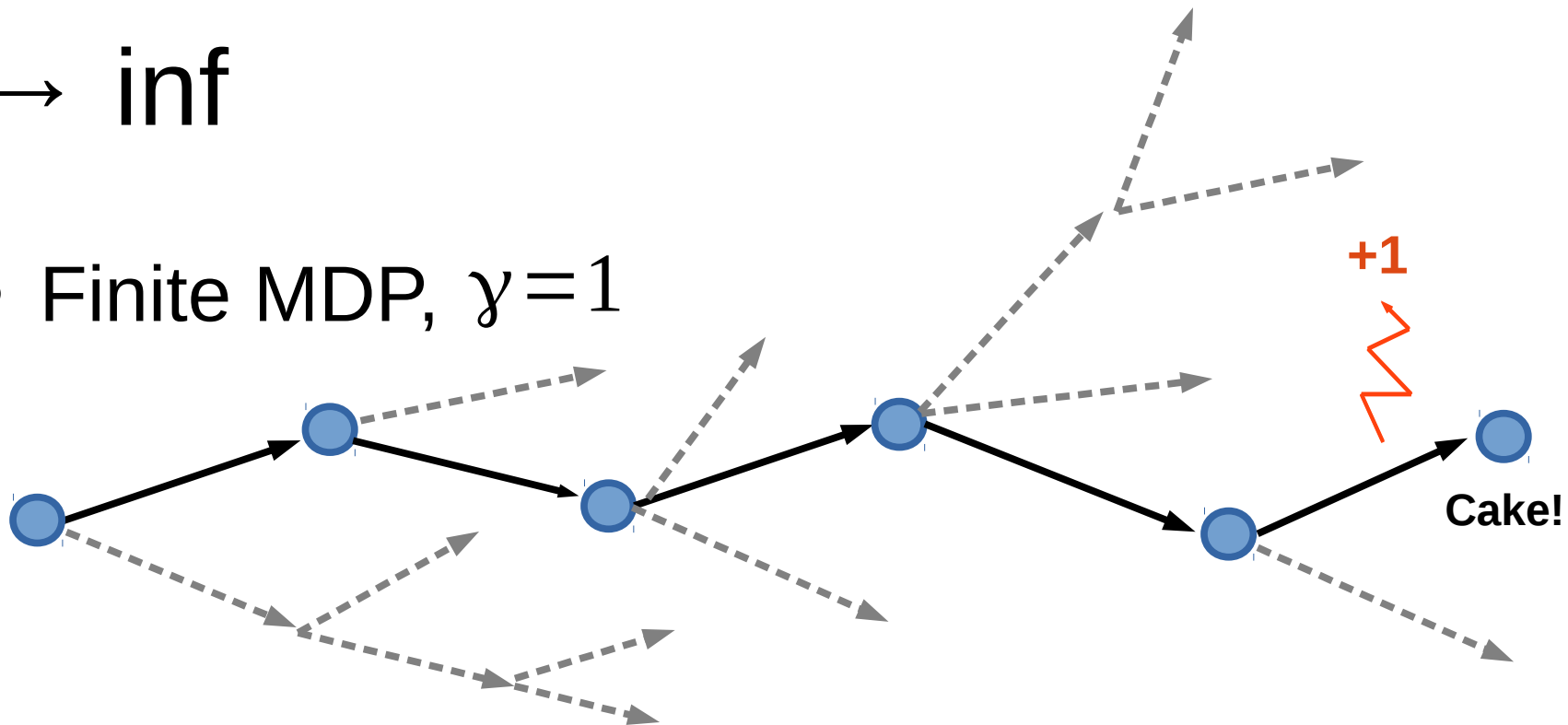
- Simple example



- SARSA needs 5 steps, n-step SARSA needs 1
- Nuts and bolts
 - Nonlinear approximations learn much faster!
 - Play for N steps, then learn (batched)

$n \rightarrow \text{inf}$

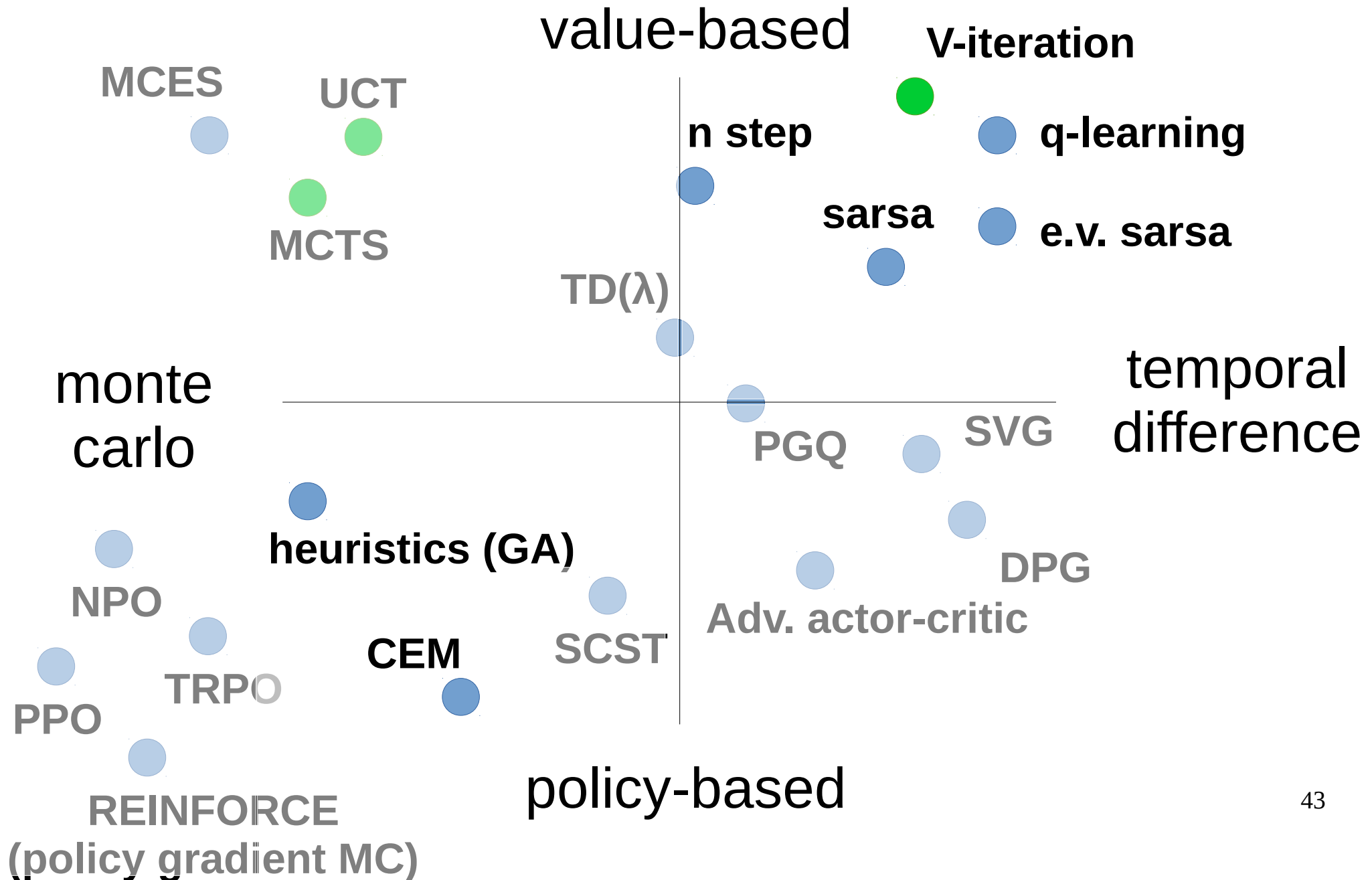
- Finite MDP, $\gamma=1$



- Sample many trajectories (or tree search)
- Compute expected $Q(s, a) = E_{\substack{s' \sim p(s'|s, a), \\ a' \sim \pi(a'|s'), \\ s'' \sim p(s''|s', a') \\ \dots}} G(s, a)$

minimal assumptions, unbiased, large variance⁴²

Methods, methods everywhere



Let's write some code!