RL @ PicsArt
Day 1, part 2

Value-based methods
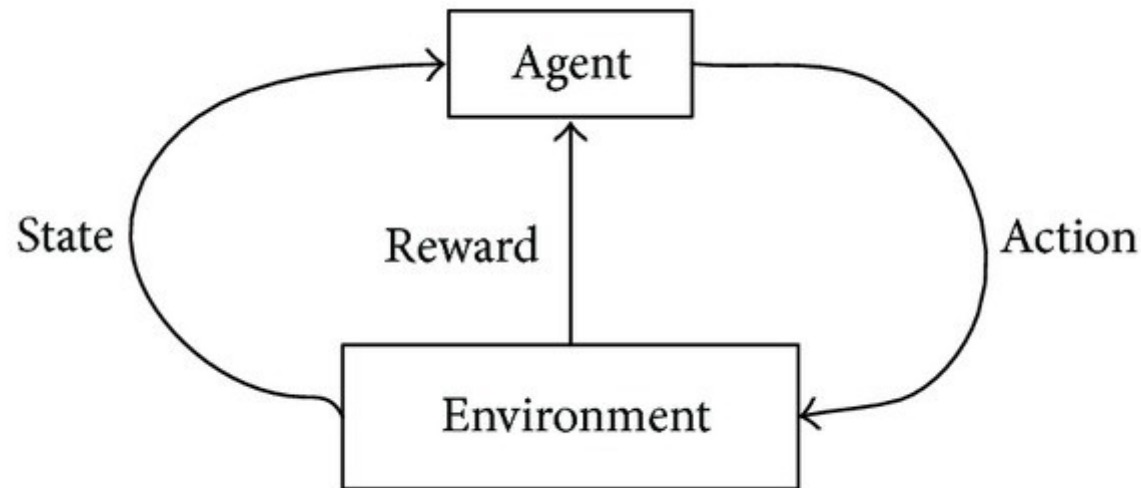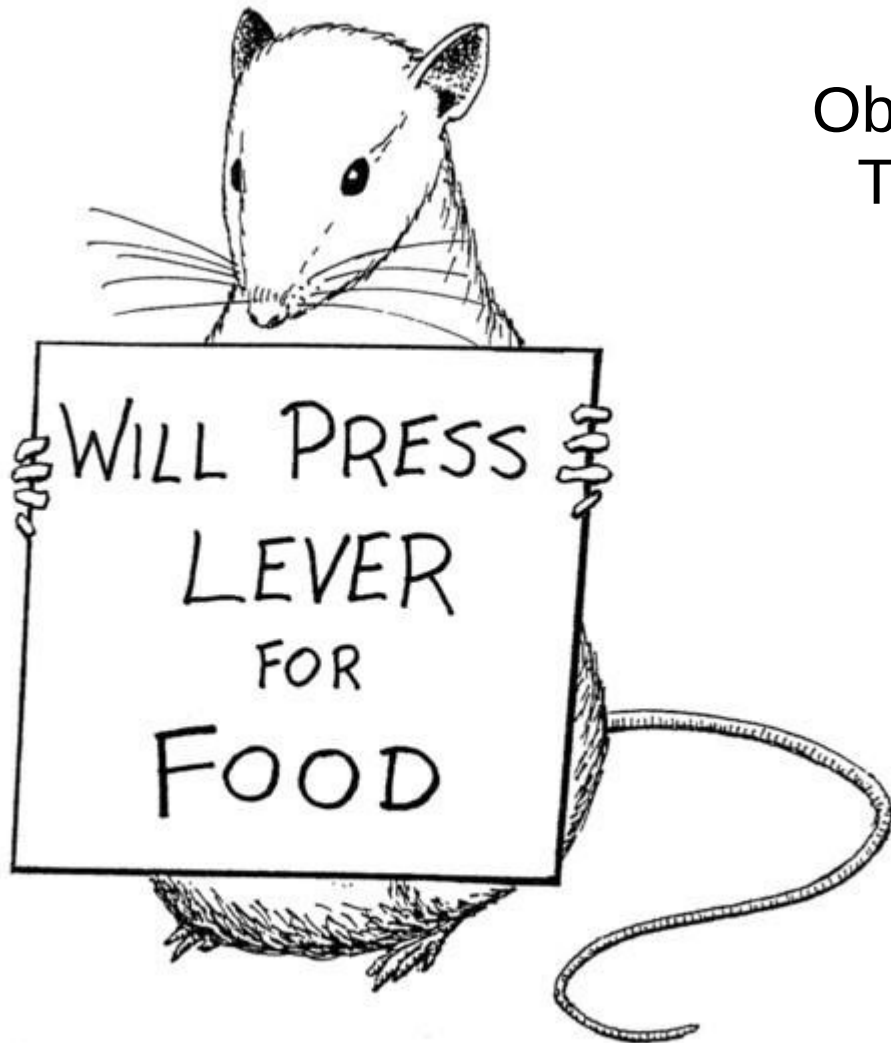
Recap: Discounted reward MDP

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

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

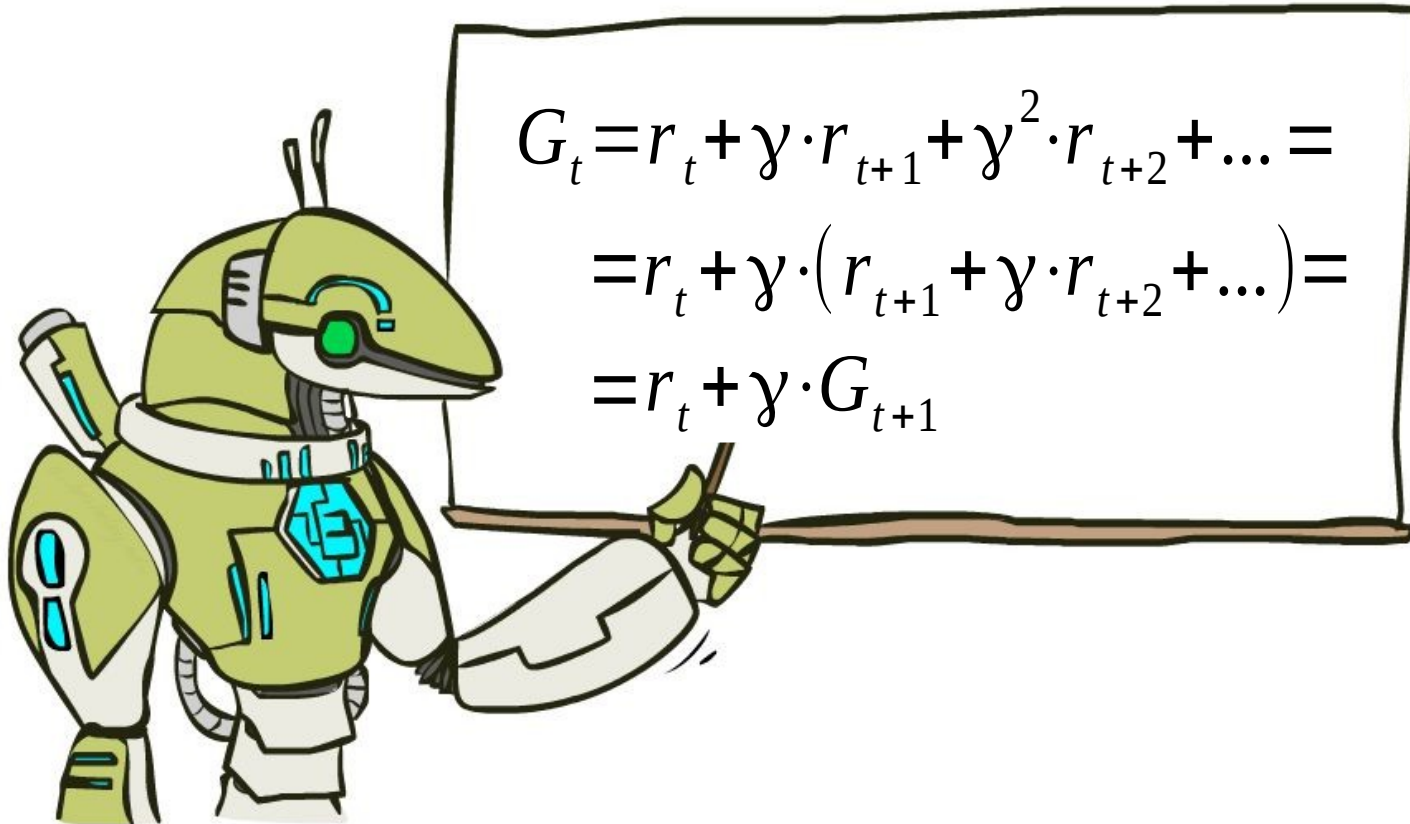$$G_t = \sum_i \gamma^i \cdot r_{t+i} \quad \gamma \in (0,1) \, const$$

γ ~ patience
Cake tomorrow is γ as good as now

Reinforcement learning:
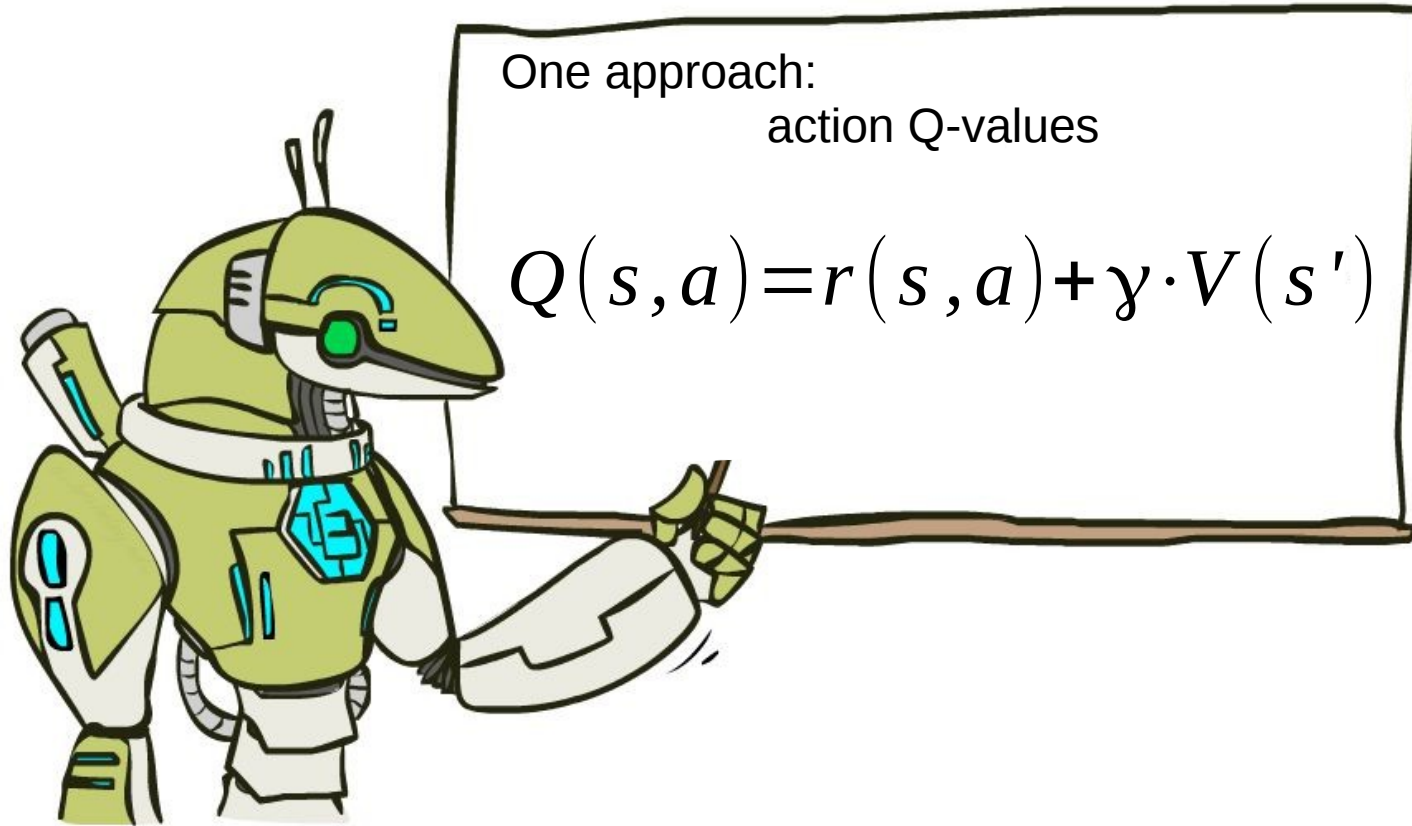* Find policy that maximizes
  the expected reward

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

3

Recap: Optimal policy

$$G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots =$$
$$= r_t + \gamma \cdot \left(r_{t+1} + \gamma \cdot r_{t+2} + \ldots \right) =$$
$$= r_t + \gamma \cdot G_{t+1}$$

We rewrite G with sheer power of math!

Recap: Q-learning

One approach:
action Q-values

$$Q(s,a) = r(s,a) + \gamma \cdot V(s')$$
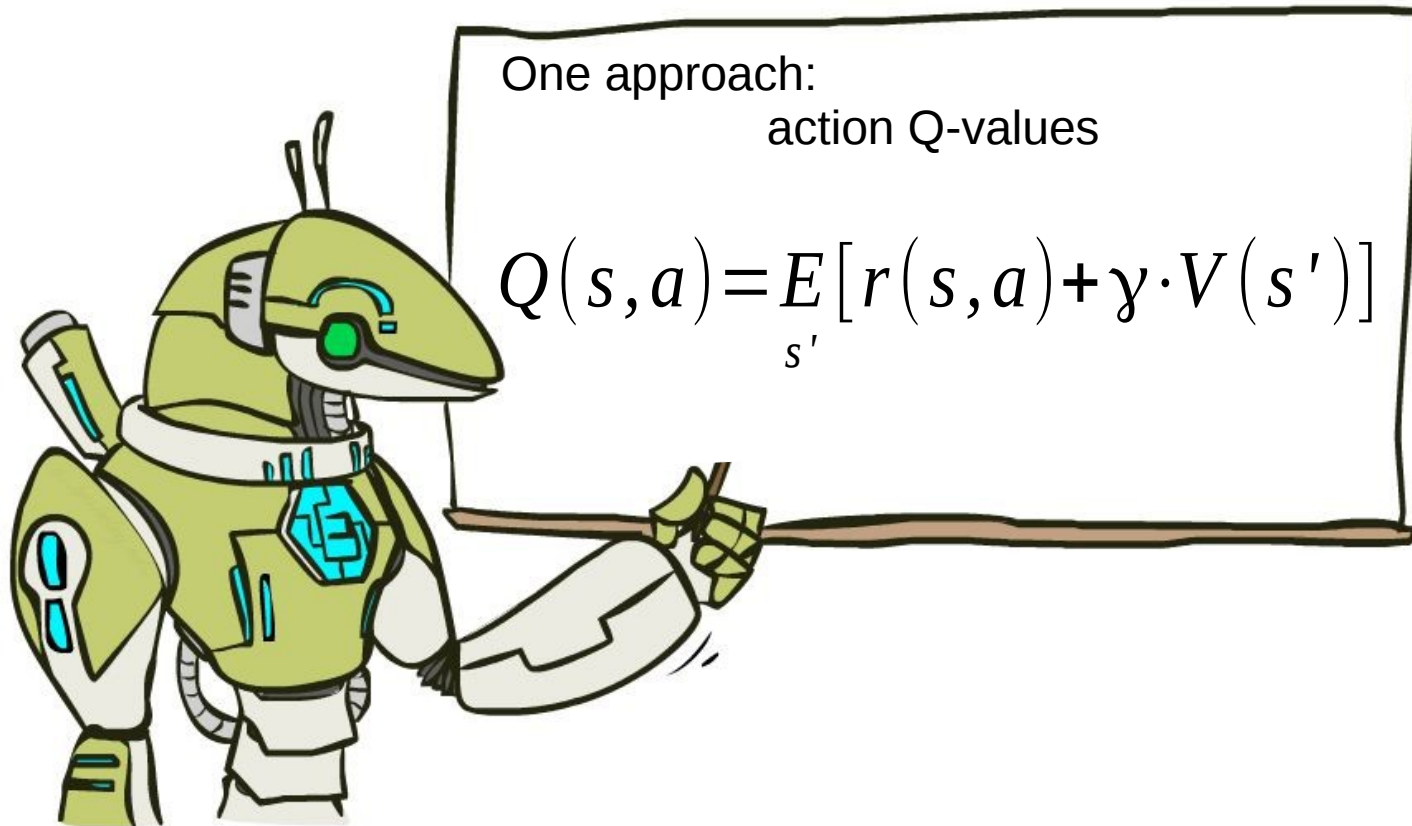
Action value Q(s,a) is the expected total reward **G** agent gets from state **s** by taking action **a** and following policy **π** from next state.

Trivia: how do we get policy **π**(a|s) given Q(s,a)?

Recap: Q-learning

One approach:
action Q-values

$$Q(s,a) = E_{s'}[r(s,a) + \gamma \cdot V(s')]$$

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.
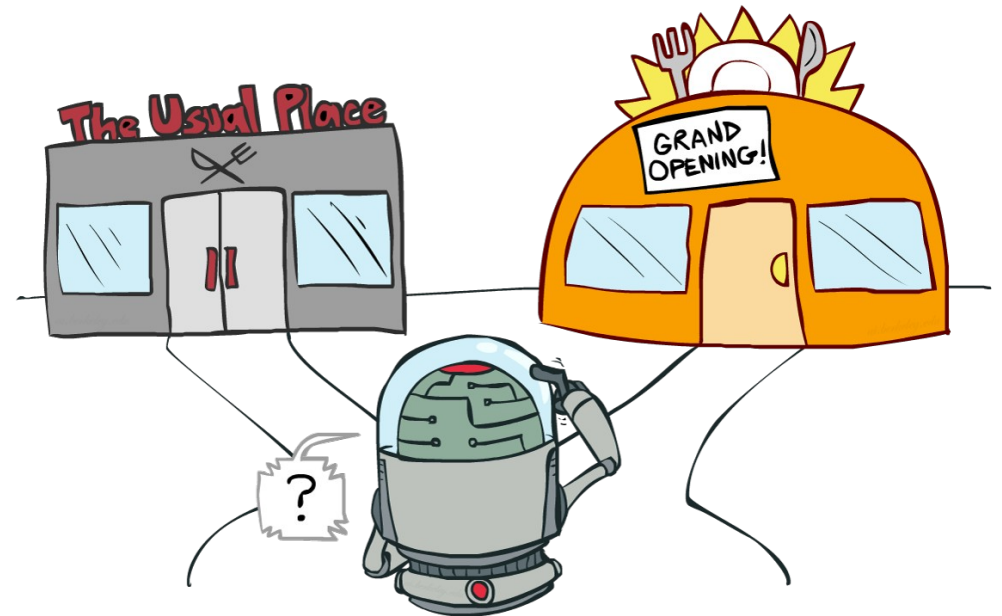
$$\pi(s): argmax_a Q(s,a)$$

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 $\pi\left(a\middle|s\right)$ now?

Exploration Vs Exploitation

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

ε-greedy

· With probability ε
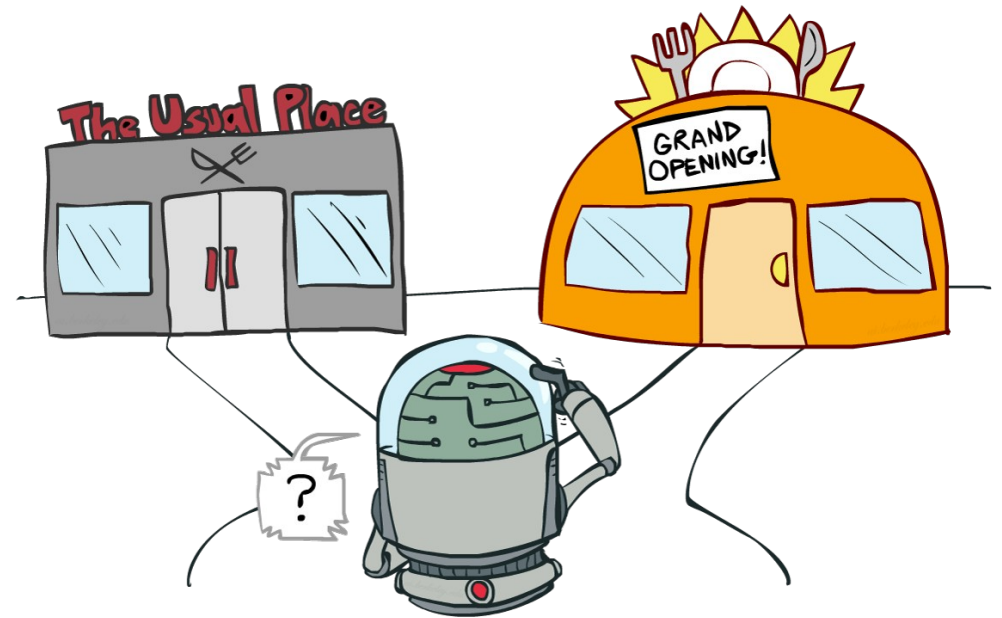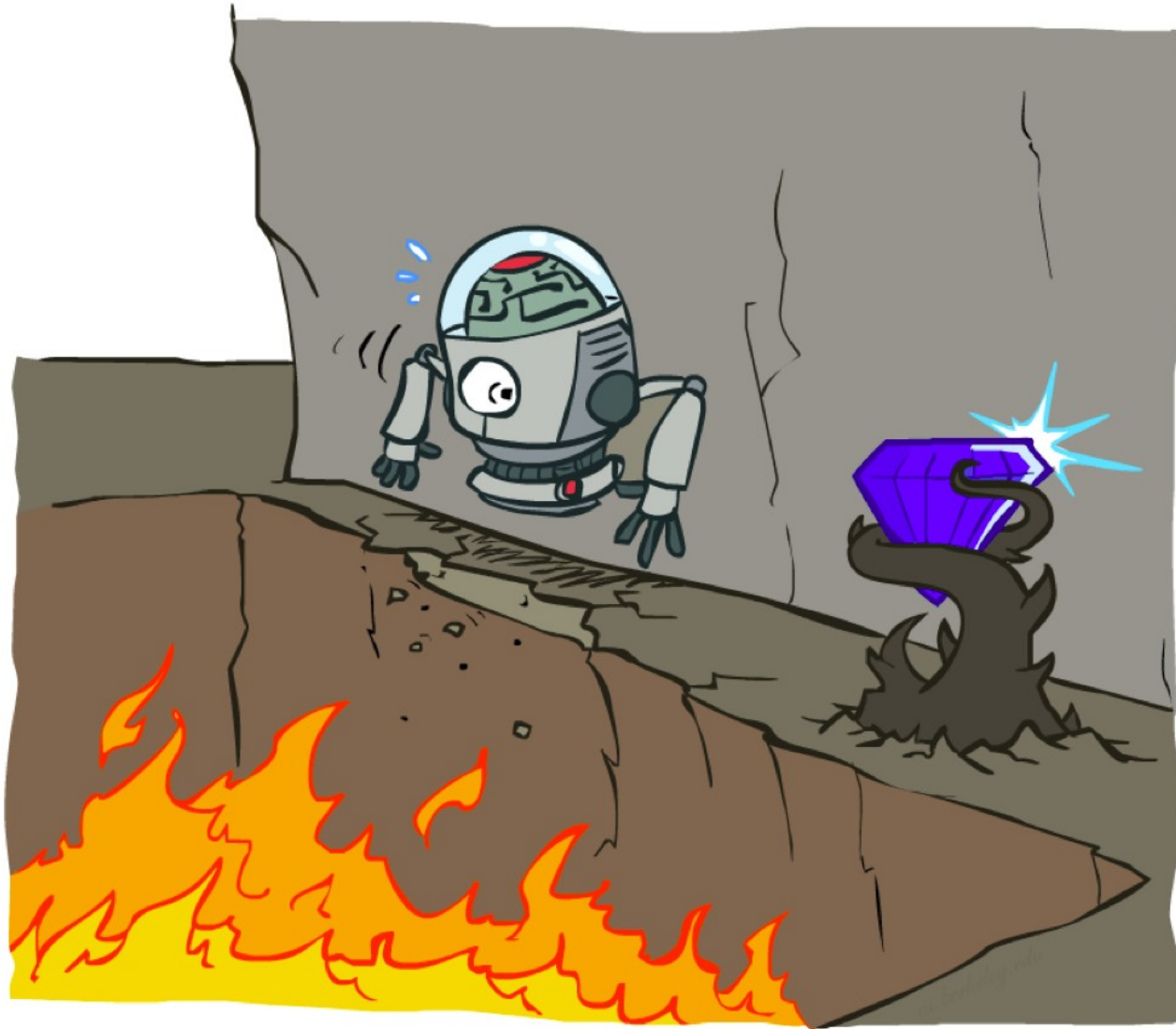take random action;

· Otherwise take
optimal action.

$$\pi(a|s):(1-\epsilon)[a=argmax_a Q(s,a)]+\frac{\epsilon}{|A|}$$
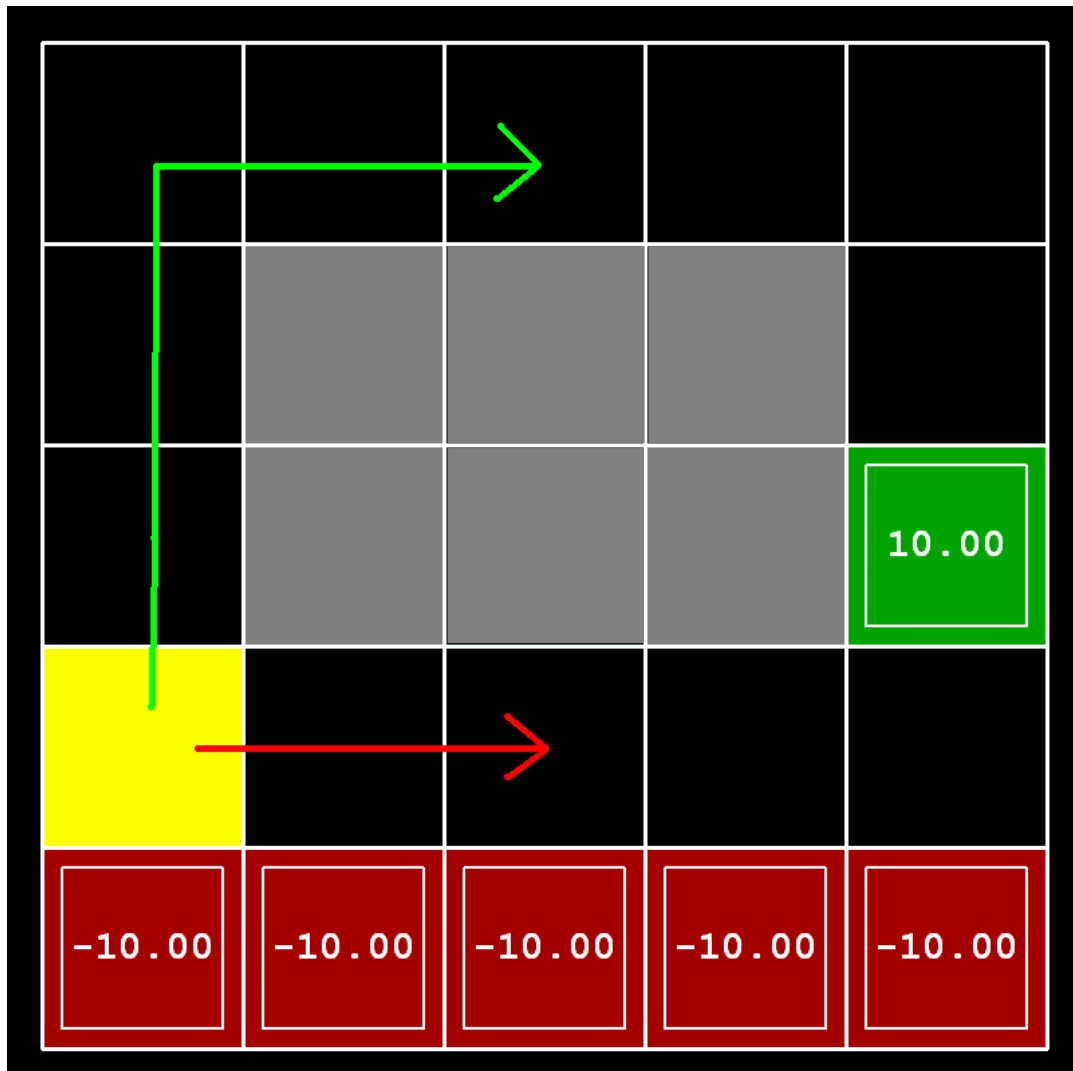
Cliff world

Picture from Berkeley CS188x
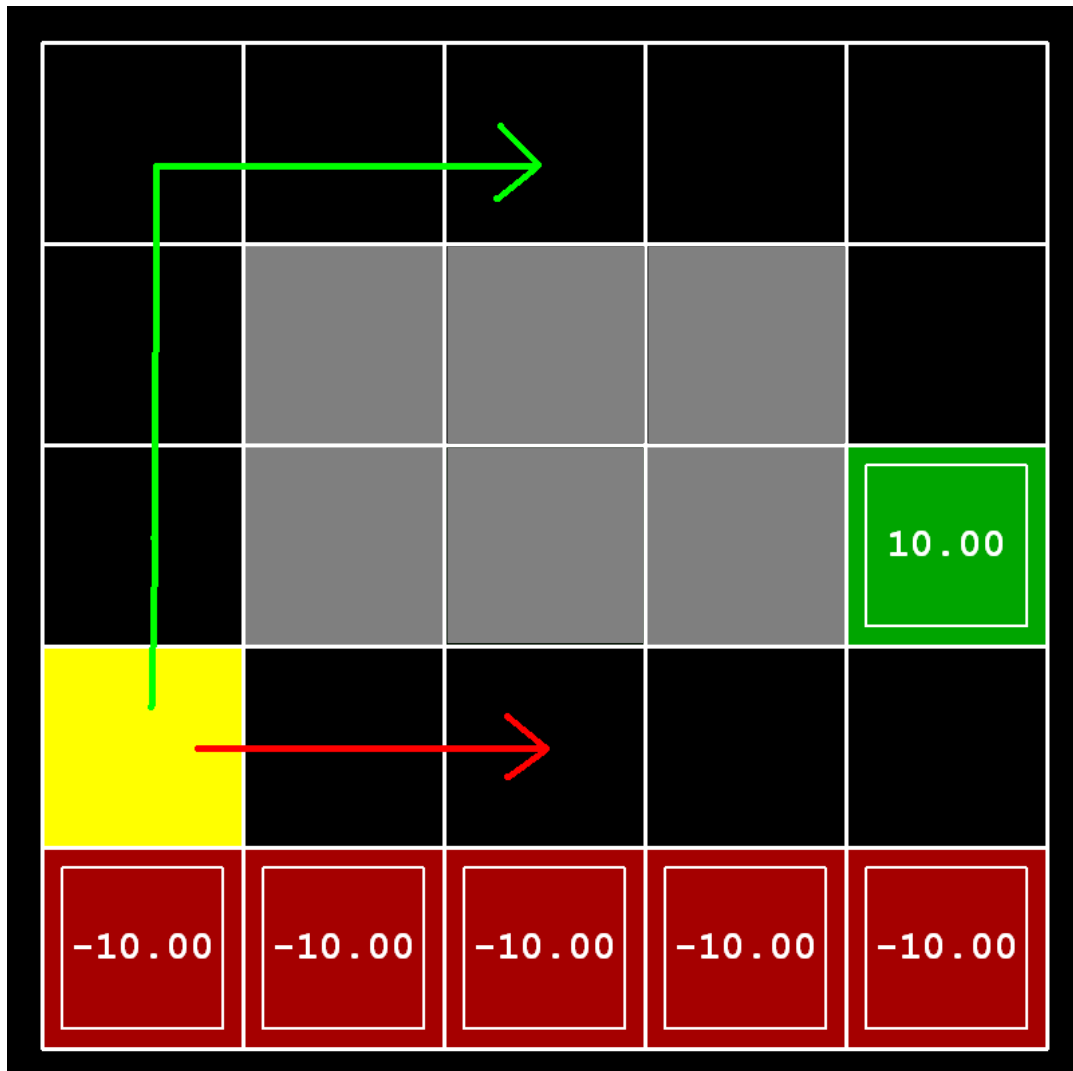
Cliff world

Conditions
  · Q-learning

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

  · 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?

11
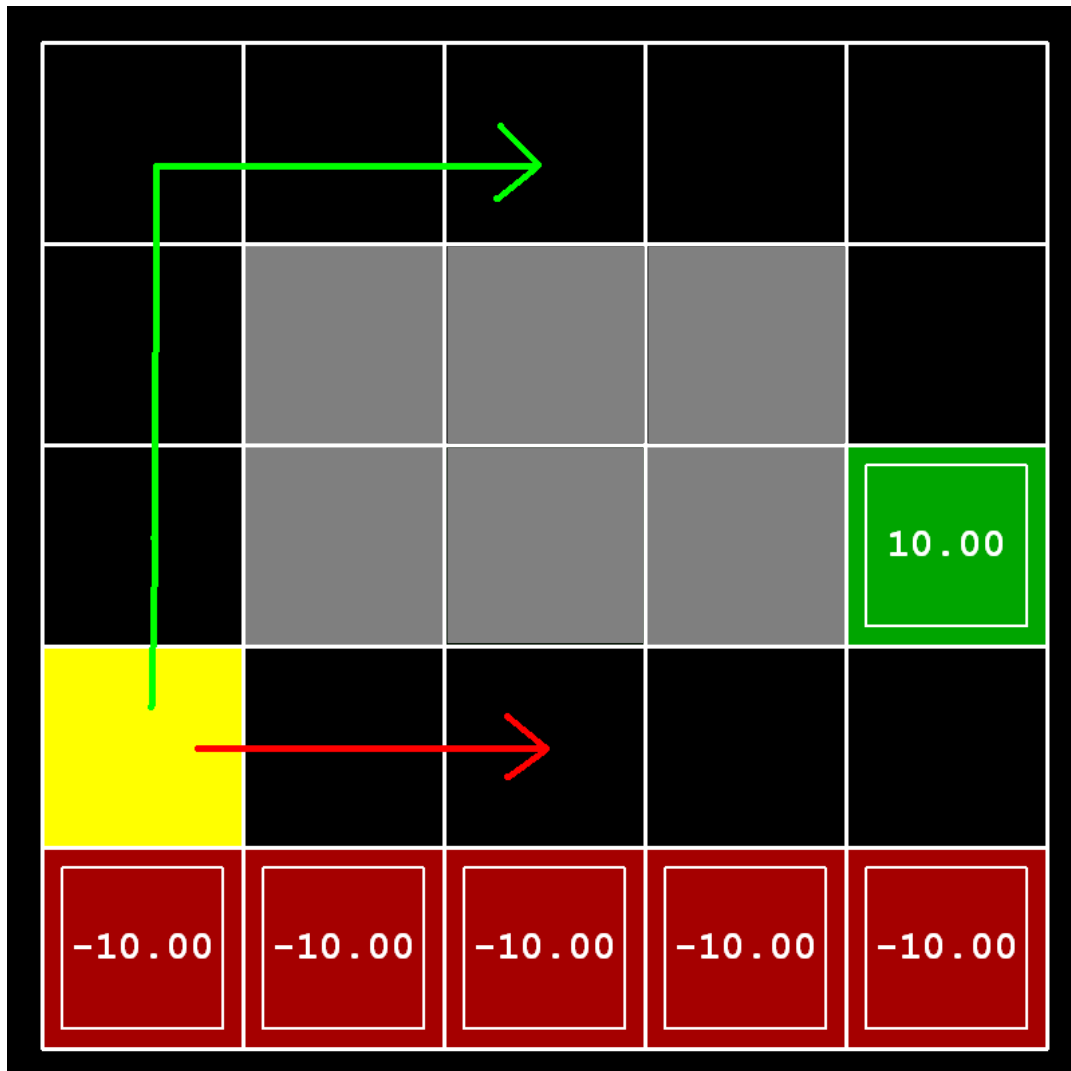
Cliff world

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
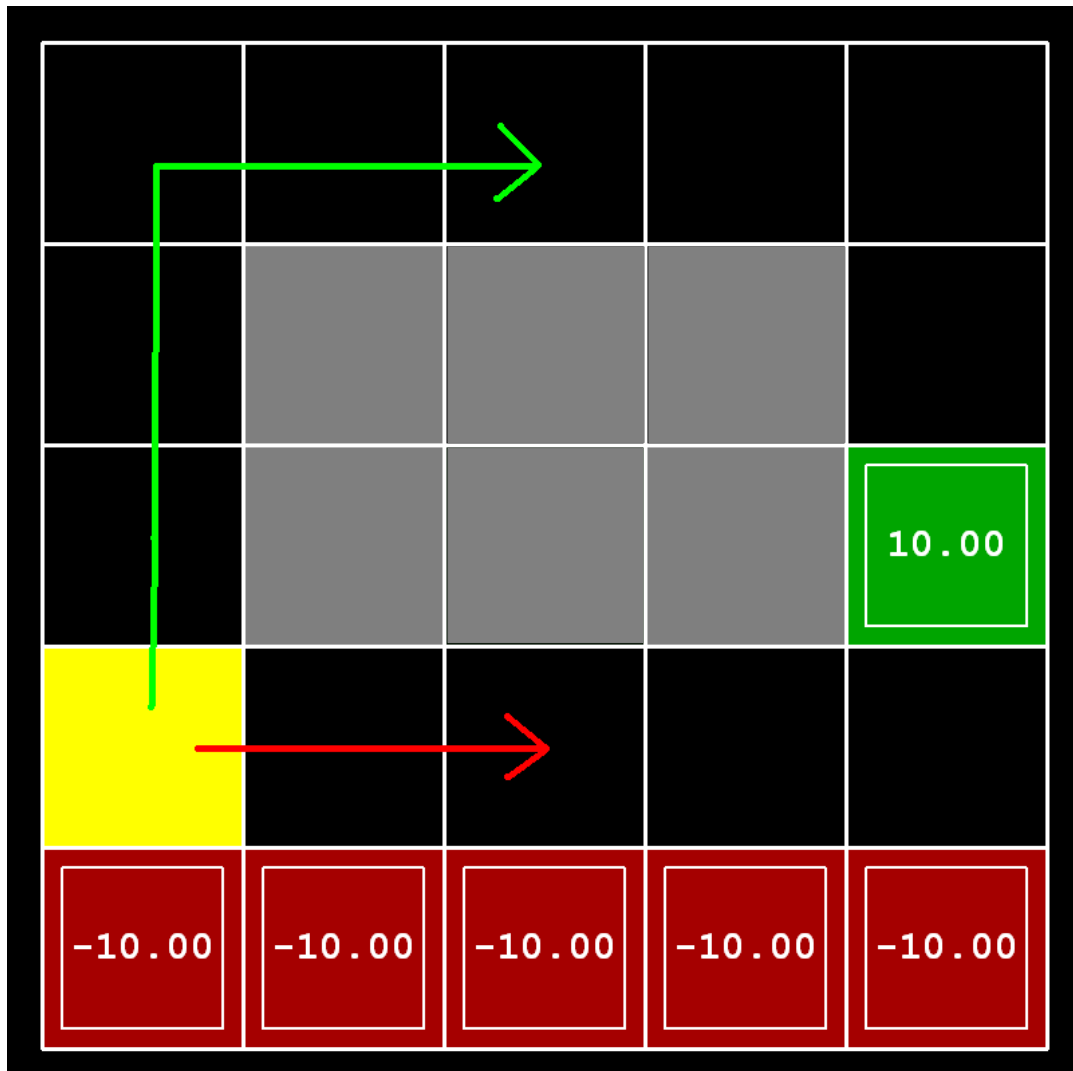
Conditions
- Q-learning

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

- 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.)

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

"better Q(s,a)"

Q-learning

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

Q-learning VS SARSA

Update rule (from Bellman eq.)

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

"better Q(s,a)"

Q-learning

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

SARSA

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

Q-learning VS SARSA

Update rule (from Bellman eq.)

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

"better Q(s,a)"

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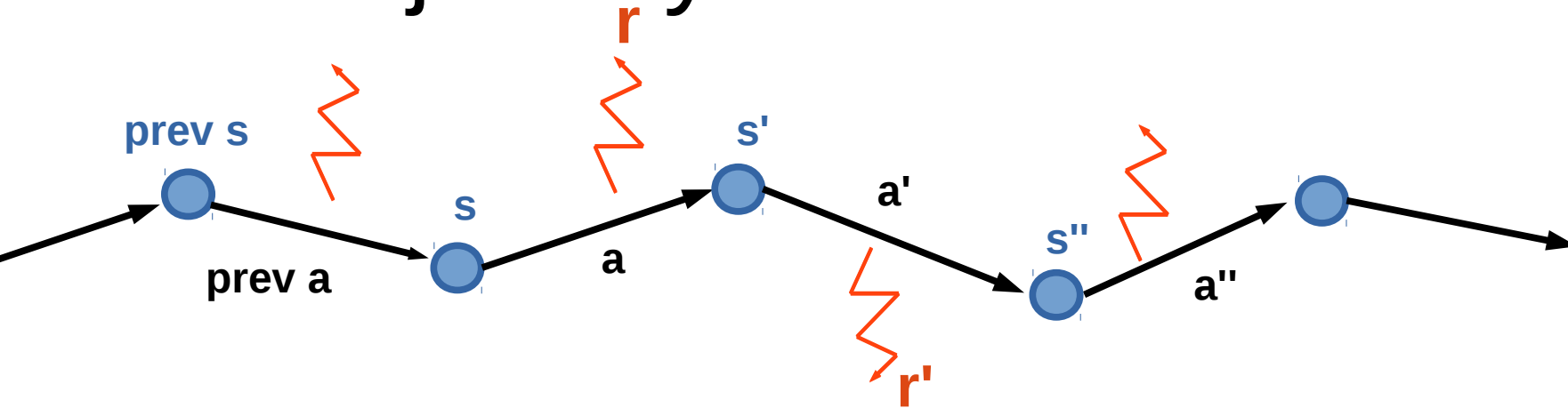
$$\hat{Q}(s, a) = r(s, a) + \gamma \cdot \max_{a'} Q(s', a')$$

**Expected from
s' ~ P(s'|s,a)**

SARSA

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

MDP trajectory

- sample sequence of
  - states (s)
  - actions (a)
  - rewards (r)

- Can be infinite, we can't wait that long

Recap: Q-learning
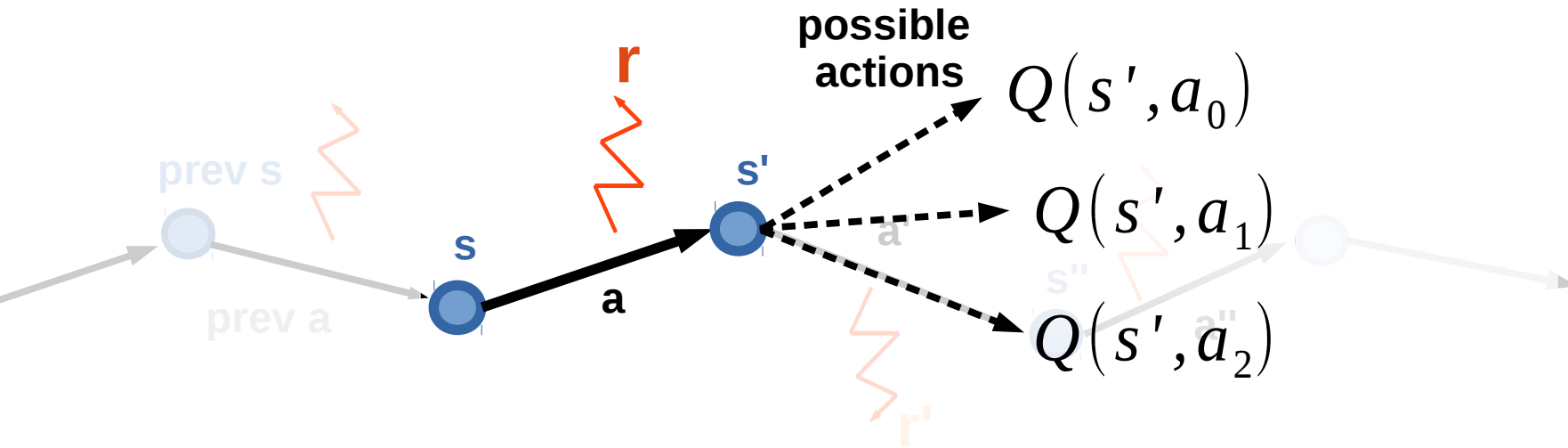
$$\forall s \in S, \forall a \in A, Q(s,a) \leftarrow 0$$

Loop:

- Sample <**s,a,r,s'**> from env

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

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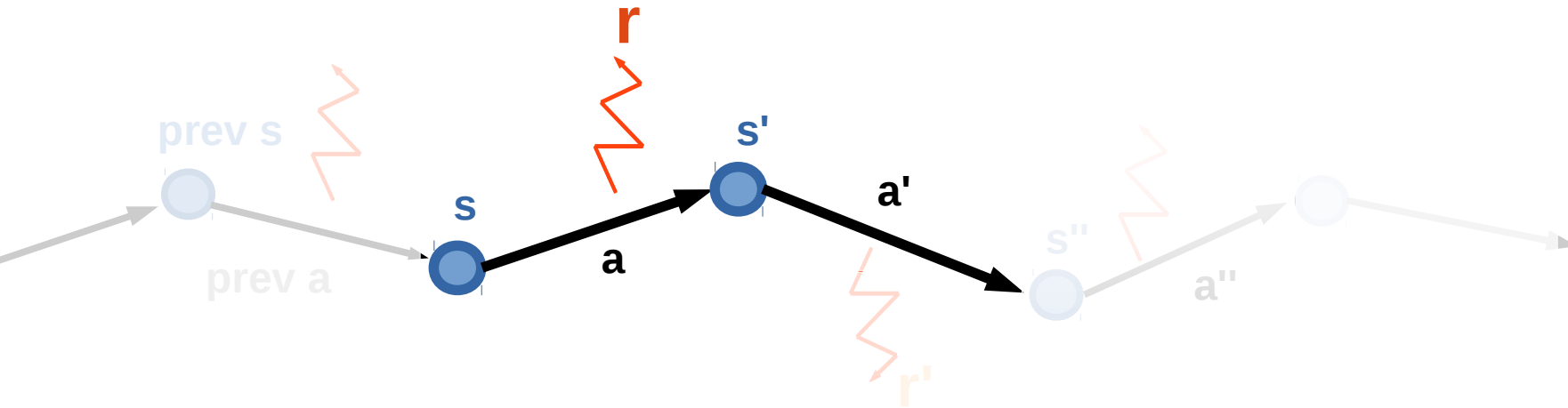
SARSA

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

Loop:

- Sample <**s,a,r,s',a'**> 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:

hence "SARSA"

– Sample <**s,a,r,s',a'**> from env

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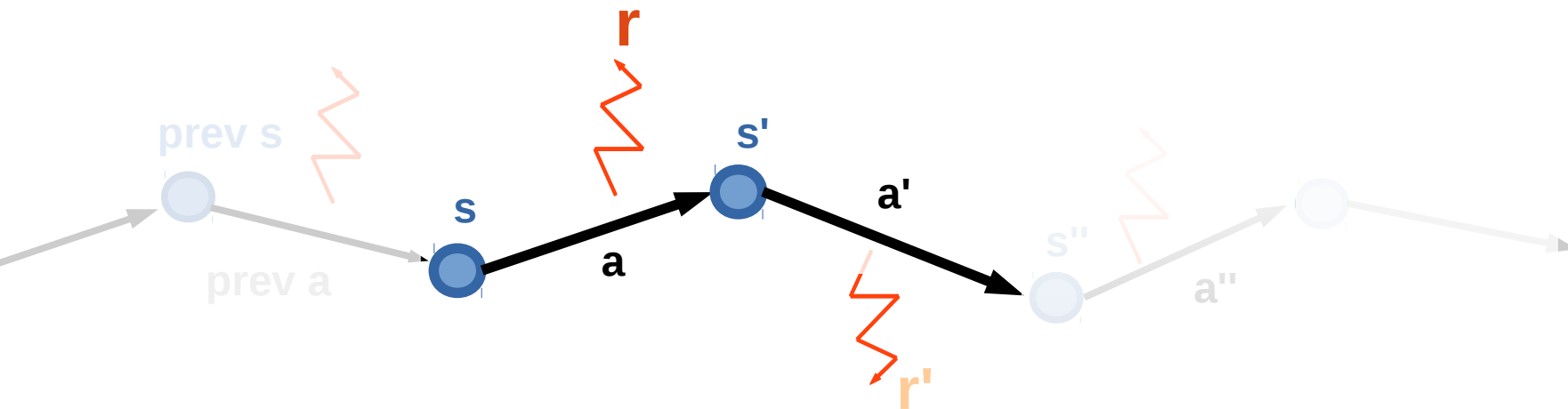
next action (not max)

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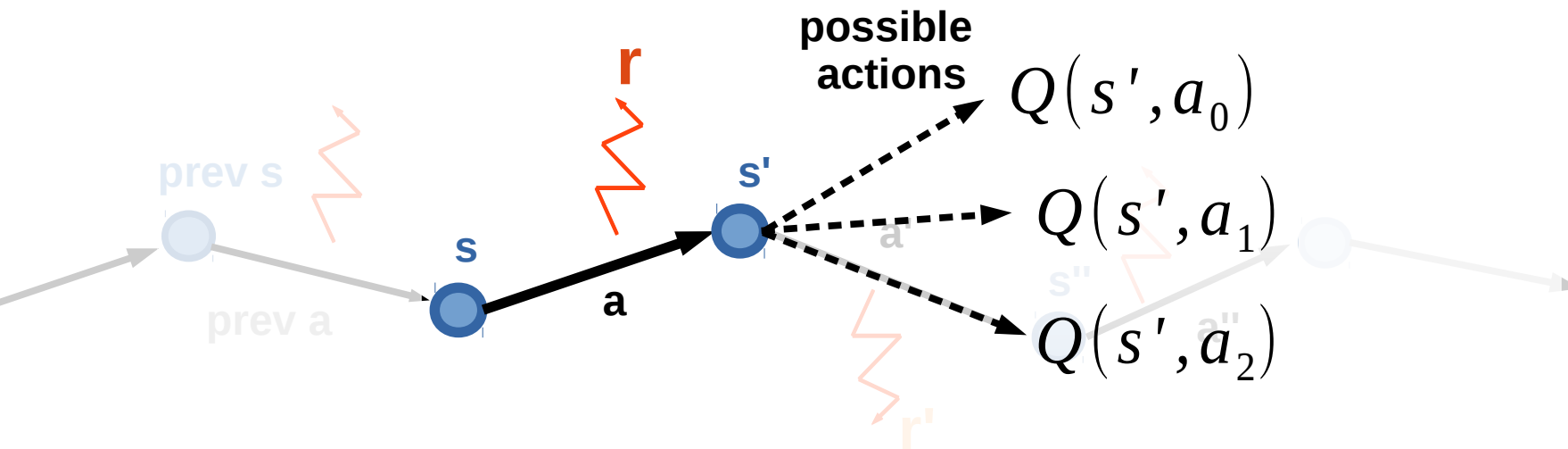
Expected value SARSA

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

Loop:

– Sample <**s,a,r,s'**> from env

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

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

Expected value SARSA

possible actions

$$Q(s',a_0)$$
$$Q(s',a_1)$$
$$Q(s',a_2)$$

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

Loop:

– Sample <**s,a,r,s'**> from env

Expected value

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

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

23
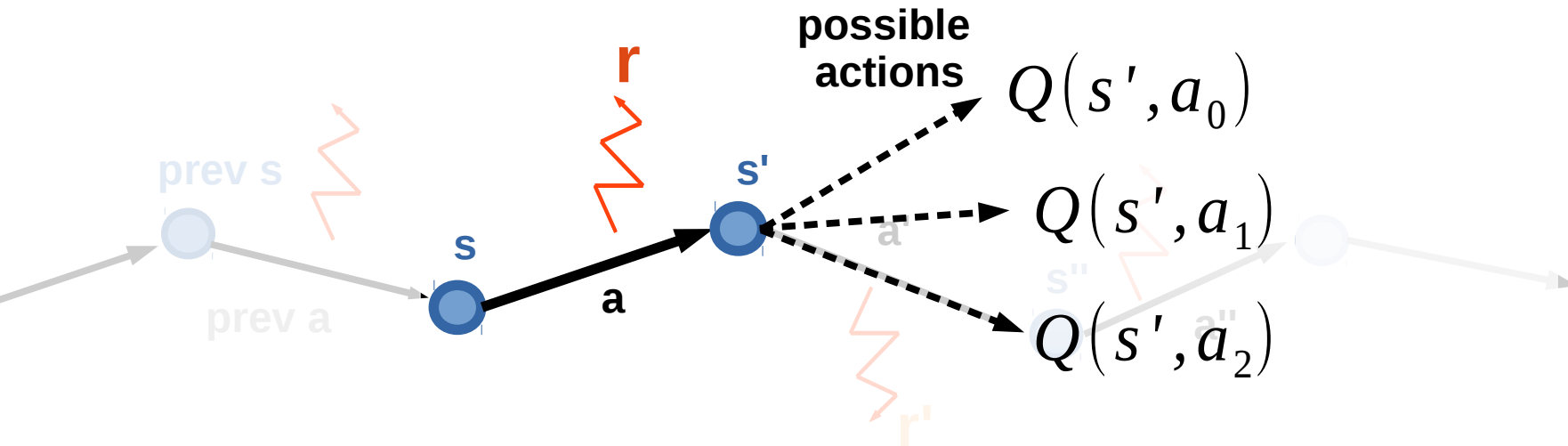
Difference

- SARSA converges to optimal policy

- Q-learning policy **would be** optimal without exploration

On-policy vs Off-policy
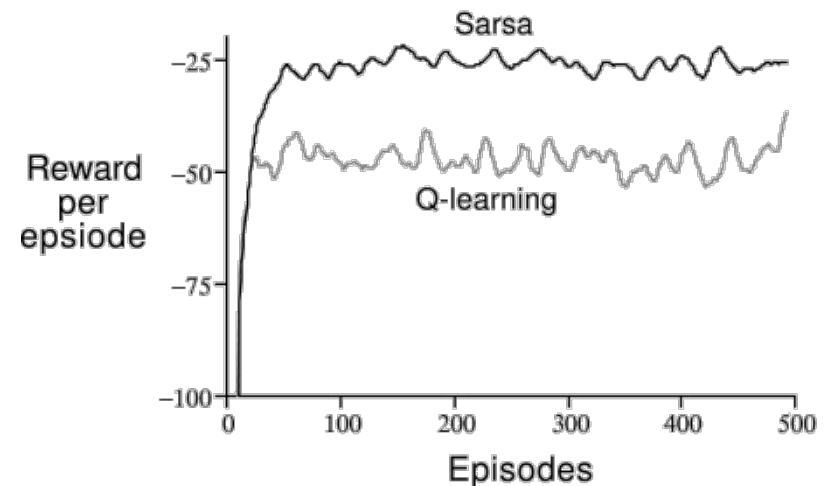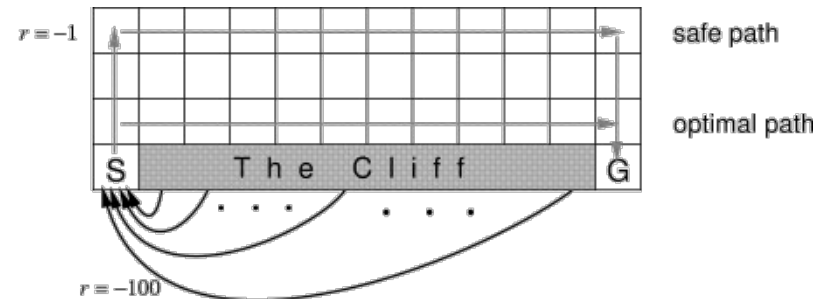
Two problem setups

on-policy	**off-policy**
Agent **can** pick actions	Agent **can't** pick actions
– Most obvious setup :)	– Learning with exploration, playing without exploration
– Agent always follows his **own** policy	– Learning from expert (expert is imperfect)
	– Learning from sessions (recorded data)

On-policy vs Off-policy

Two problem setups

on-policy	**off-policy**
Agent **can** pick actions	Agent **can't** pick actions
– On-policy algorithms **can't** learn off-policy	– Off-policy algorithms **can** learn on-policy
(but they be faster/better)	learn optimal policy even if agent takes random actions

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

26

On-policy vs Off-policy

Two problem setups

on-policy	**off-policy**
Agent **can** pick actions	Agent **can't** pick actions
– On-policy algorithms **can't** learn off-policy	– Off-policy algorithms **can** learn on-policy
– SARSA	– Q-learning
– more coming soon	– 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	**off-policy**
Agent **can** pick actions	Agent **can't** pick actions
– On-policy algorithms **can't** learn off-policy	– Off-policy algorithms **can** learn on-policy
– SARSA	– Q-learning
– more coming soon	– Expected Value SARSA

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

N-step algorithms

Recall R?

$$G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + ... =$$
$$= r_t + \gamma \cdot (r_{t+1} + \gamma \cdot r_{t+2} + ...) =$$
$$= r_t + \gamma \cdot G_{t+1} =$$
$$= r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot G_{t+2}$$

N-step SARSA

- General formula

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

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

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

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

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

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

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

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

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

N-step Q-learning

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

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

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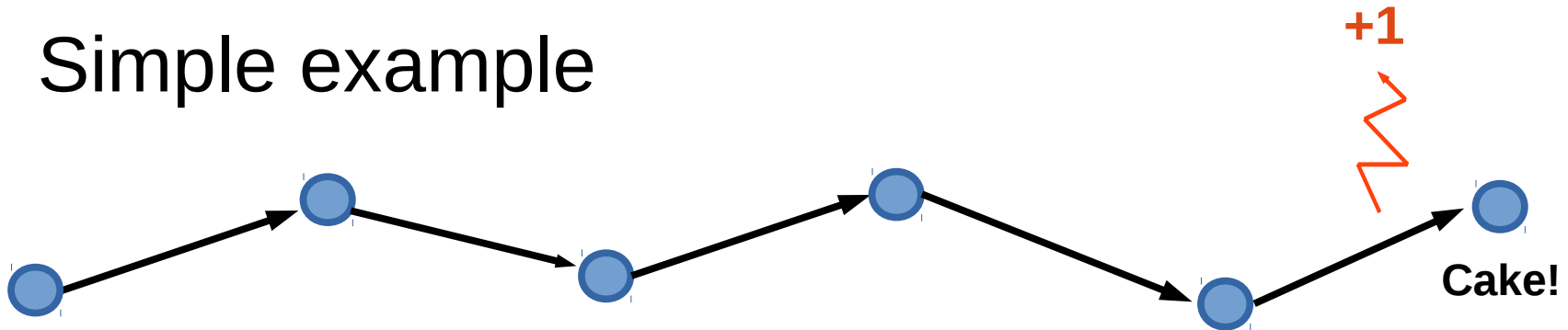
N-step Q-learning

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

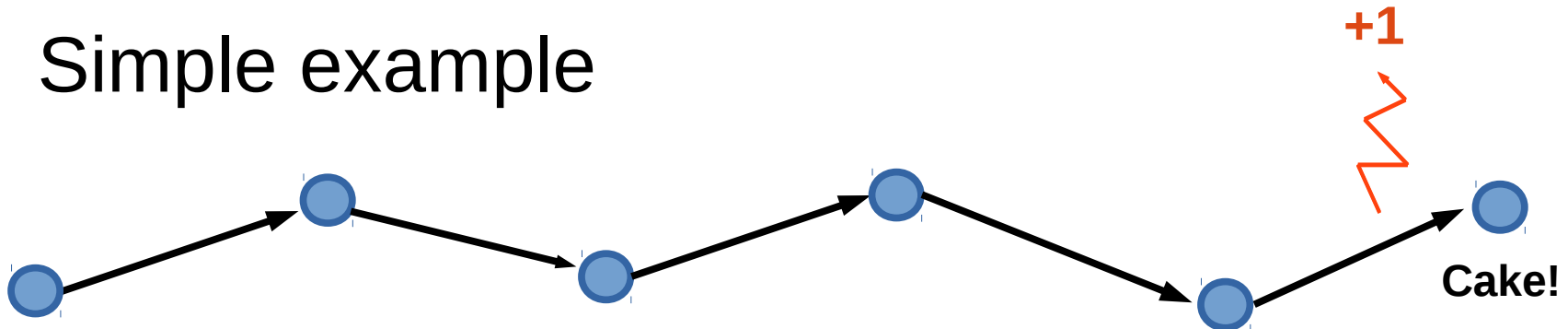**Trivia:** which of these methods works off-policy? **None of them!**

1-step Vs n-step

- Simple example

+1

Cake!

How many games does it take for **SARSA** to learn the optimal policy?
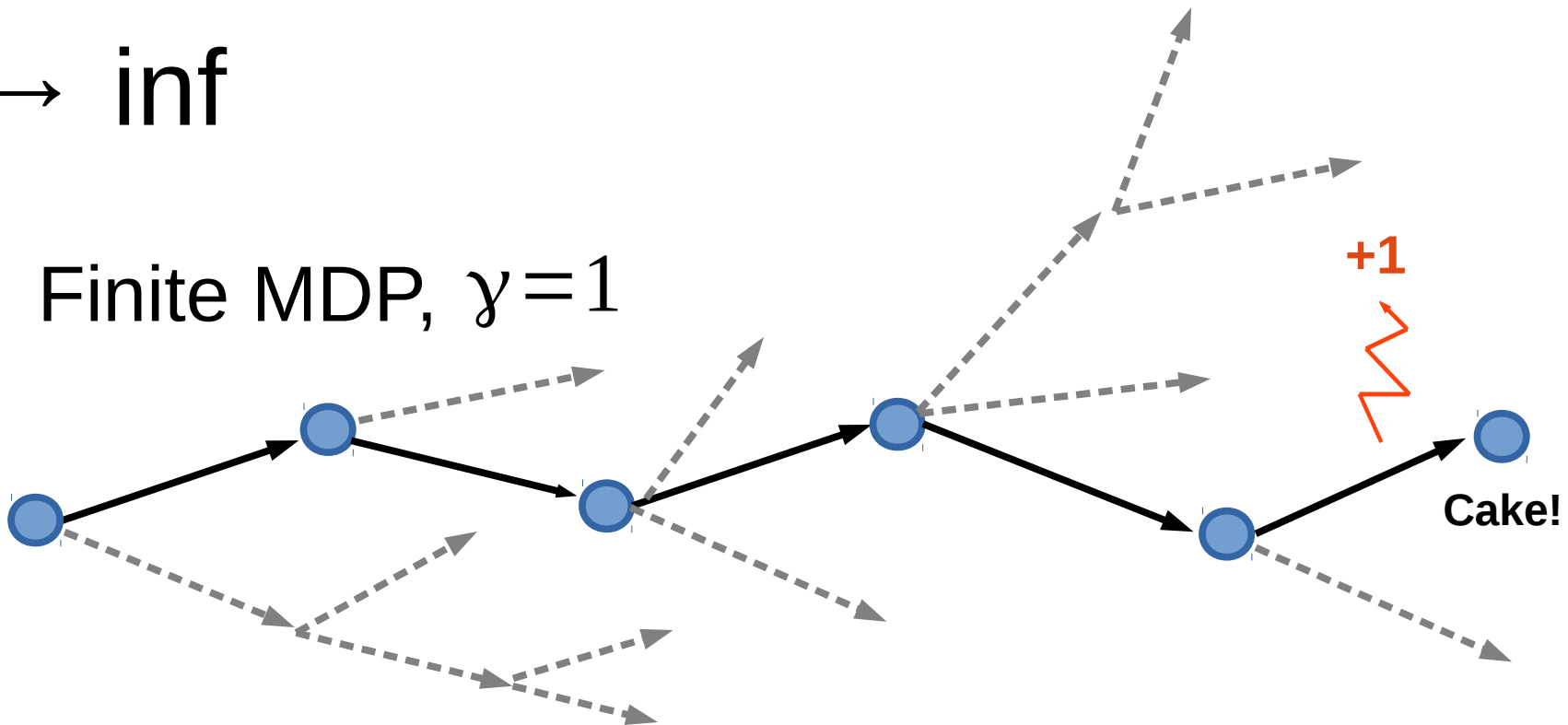
1-step Vs n-step

- Simple example

+1

Cake!

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

n → inf

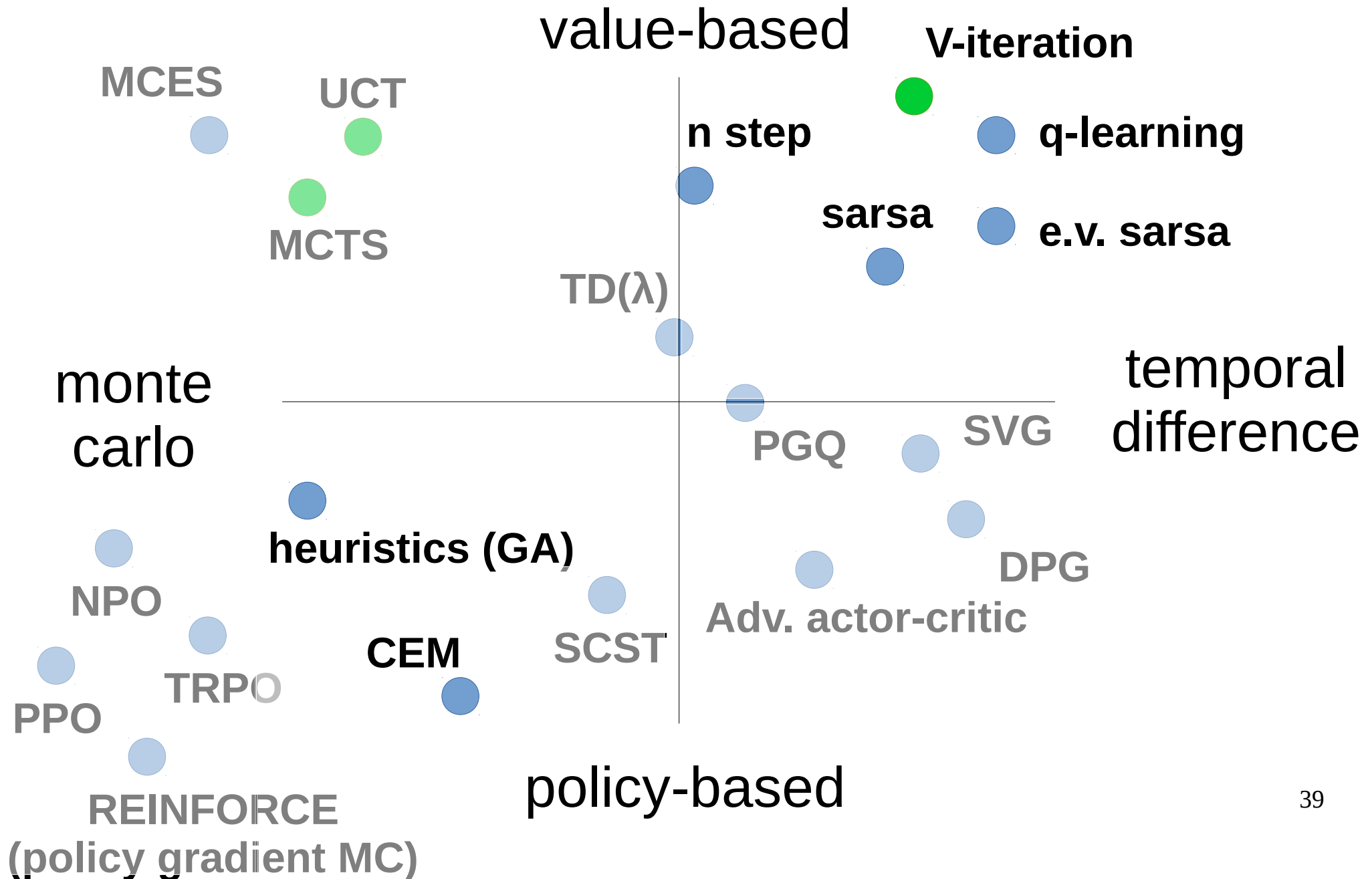- Finite MDP, $\gamma=1$

+1

Cake!

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

minimal assumptions, unbiased, large variance[38]

Methods, methods everywhere

value-based

V-iteration

MCES

UCT

n step

q-learning

MCTS

sarsa

e.v. sarsa

TD(λ)

monte
carlo

temporal
difference

PGQ

SVG

heuristics (GA)

DPG

NPO

Adv. actor-critic

SCST

CEM

TRPO

PPO

policy-based

REINFORCE
(policy gradient MC)

39

Let's write some code!