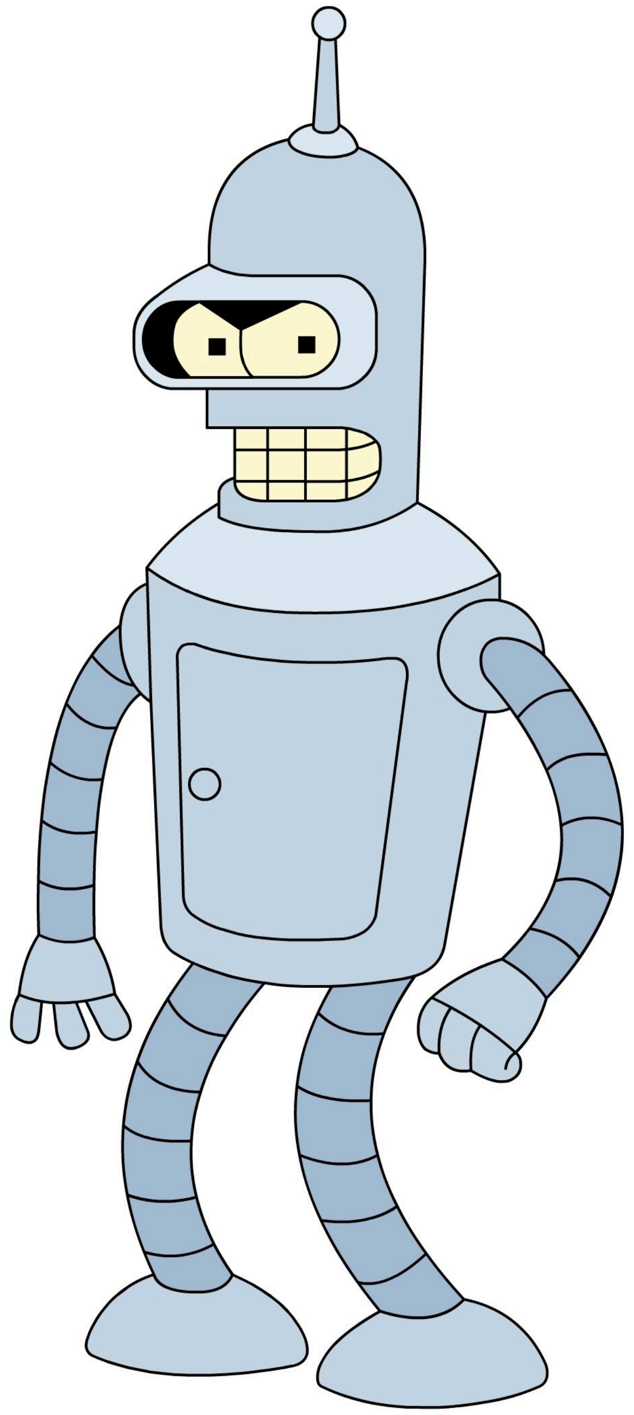
Reinforcement Learning

HSE, winter - spring 2025

Lecture 2: Model-free RL



Sergei Laktionov
slaktionov@hse.ru
LinkedIn

Recap: MDP

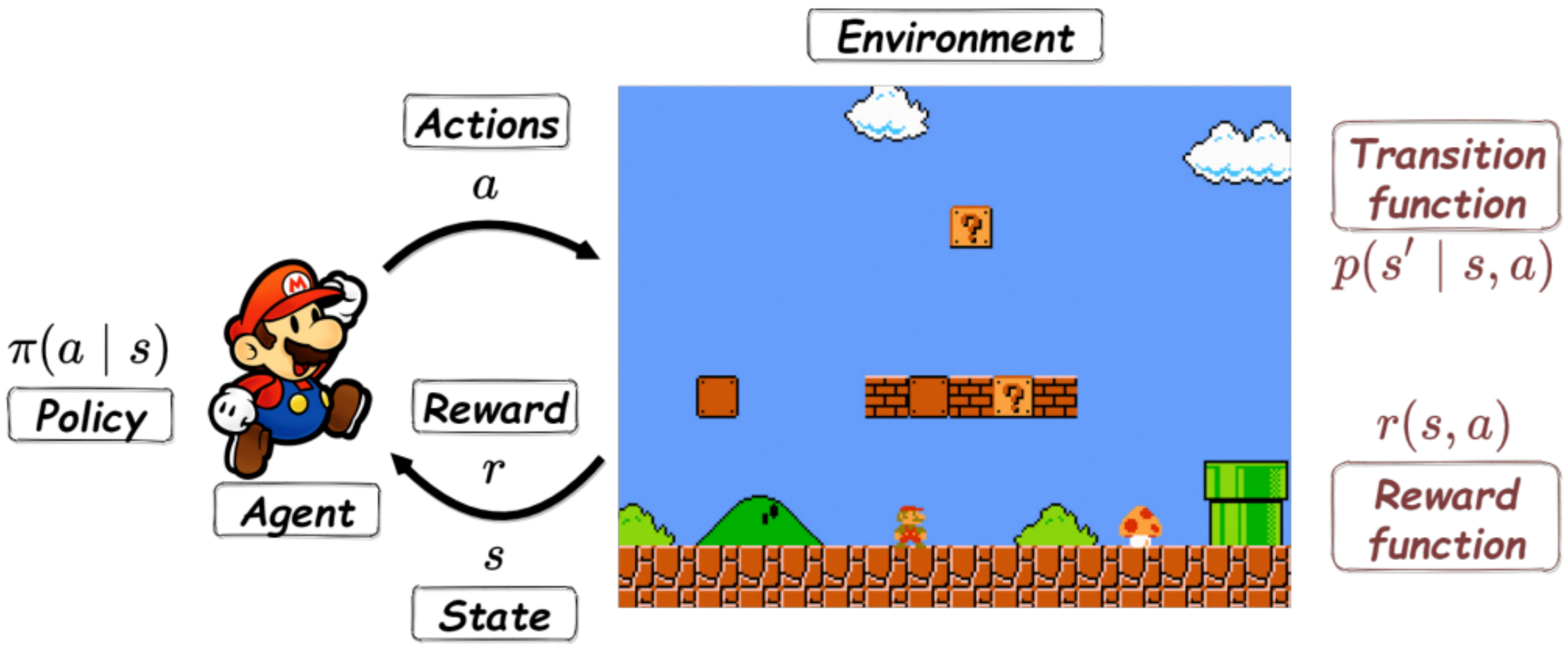
MDP is a 4-tuple $(\mathcal{S}, \mathcal{A}, p, r)$:

1. $\mathcal{A}$ is an action space

2. $\mathcal{S}$ is a state space

3. $p(s' \mid s, a) = \mathbb{P}(S_{t+1} = s' \mid S_t = s, A_t = a)$
   is a state-transition function

4. $r(s, a) \in \mathbb{R}$ is a reward function



Source

$$J(\pi) = \mathbb{E}_\pi[\sum_{t \geq 0} \gamma^t R_t] \rightarrow \max_\pi$$

Recap: Value Functions

$$G_t = \sum_{k \geq 0} \gamma^k R_{t+k}$$

$$V^\pi(s) = \mathbb{E}_\pi[G_t \,|\, S_t = s]$$

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a]$$

Recap: Objective

Let T is a final time step. If $T < \infty$ then environment is called *episodic*.

Cumulative reward is called a return or reward-to-go. Note that in general it is a random variable.

Episode length

$$\boxed{G_t} = \boxed{R_t} + \boxed{\gamma} R_{t+1} + \gamma^2 R_{t+2} + \ldots = \sum_{k=t}^{\boxed{T}} \gamma^{k-t} R_s$$

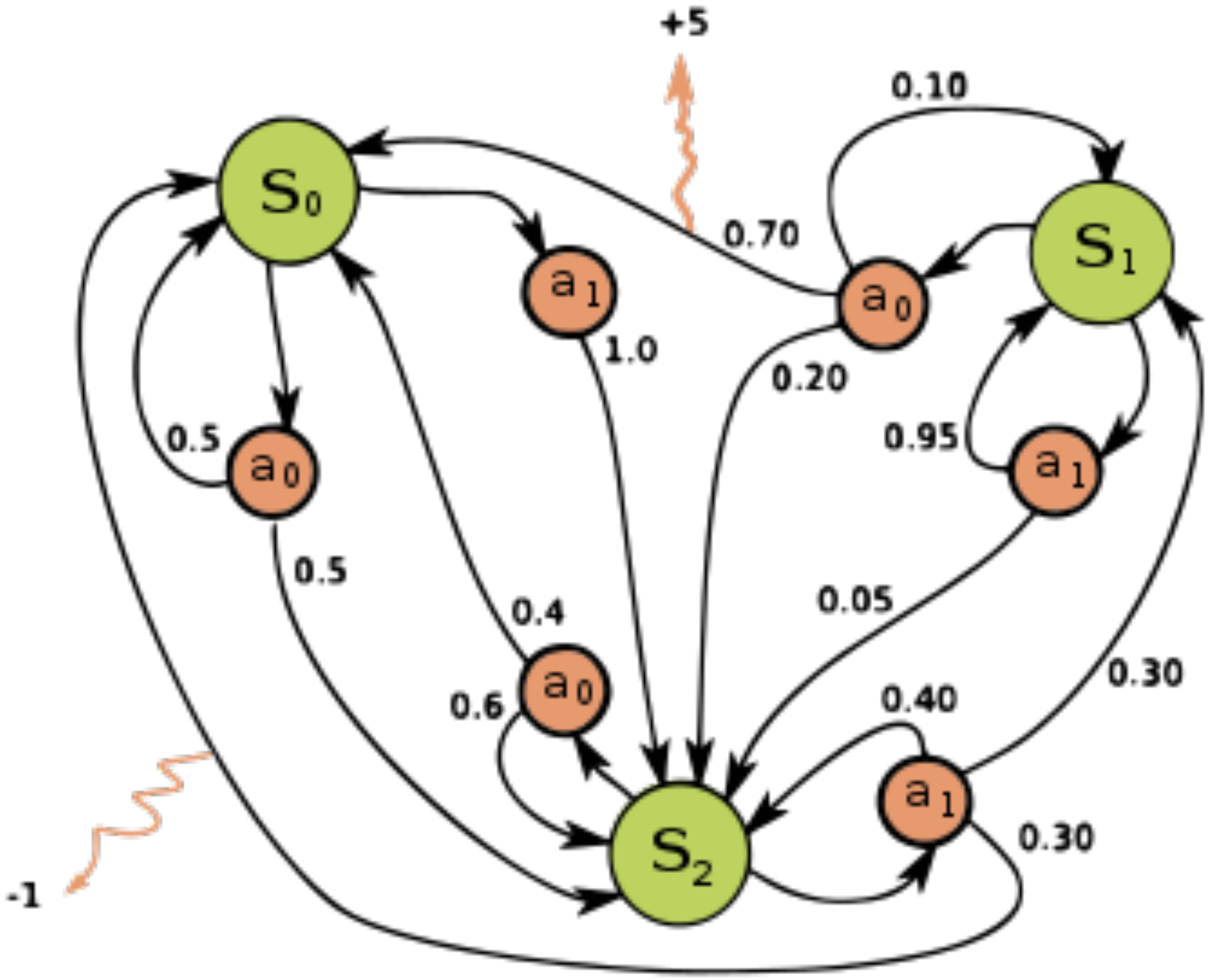
Immediate reward

Discount factor

$$J(\pi) = \mathbb{E}_\pi[G_0] \to \max_\pi$$

Recap: Assumptions

1. $p(s' \mid s, a)$ is known
2. State space is finite
3. Action space is finite



Recap: Bellman Equations

Bellman expectation equations:

$$V^{\pi}(s) = \sum_{a} \pi(a\,|\,s) \sum_{s'} p(s'\,|\,s,a)\big[r + \gamma V_{\pi}(s')\big]$$

$$Q^{\pi}(s,a) = \sum_{s'} p(s'\,|\,s,a)\Big[r + \gamma \sum_{a'} \pi(a'\,|\,s') Q_{\pi}(s',a')\Big]$$

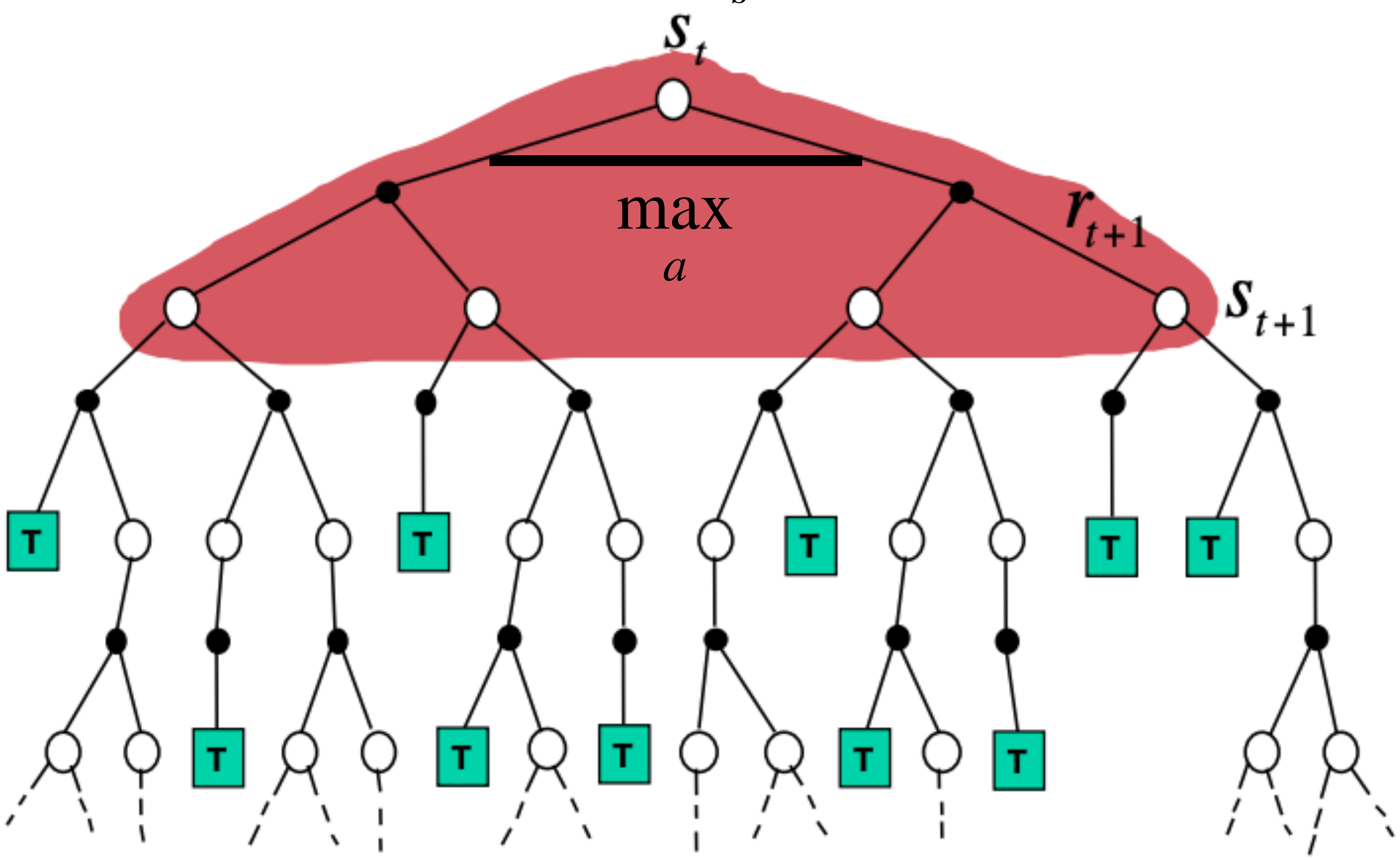
Bellman optimality equations:

$$V^{*}(s) = V^{\pi^{*}}(s) = \max_{a}[r + \gamma \sum_{s'} p(s'\,|\,s,a) V^{*}(s')]$$

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

Recap: Value Iteration

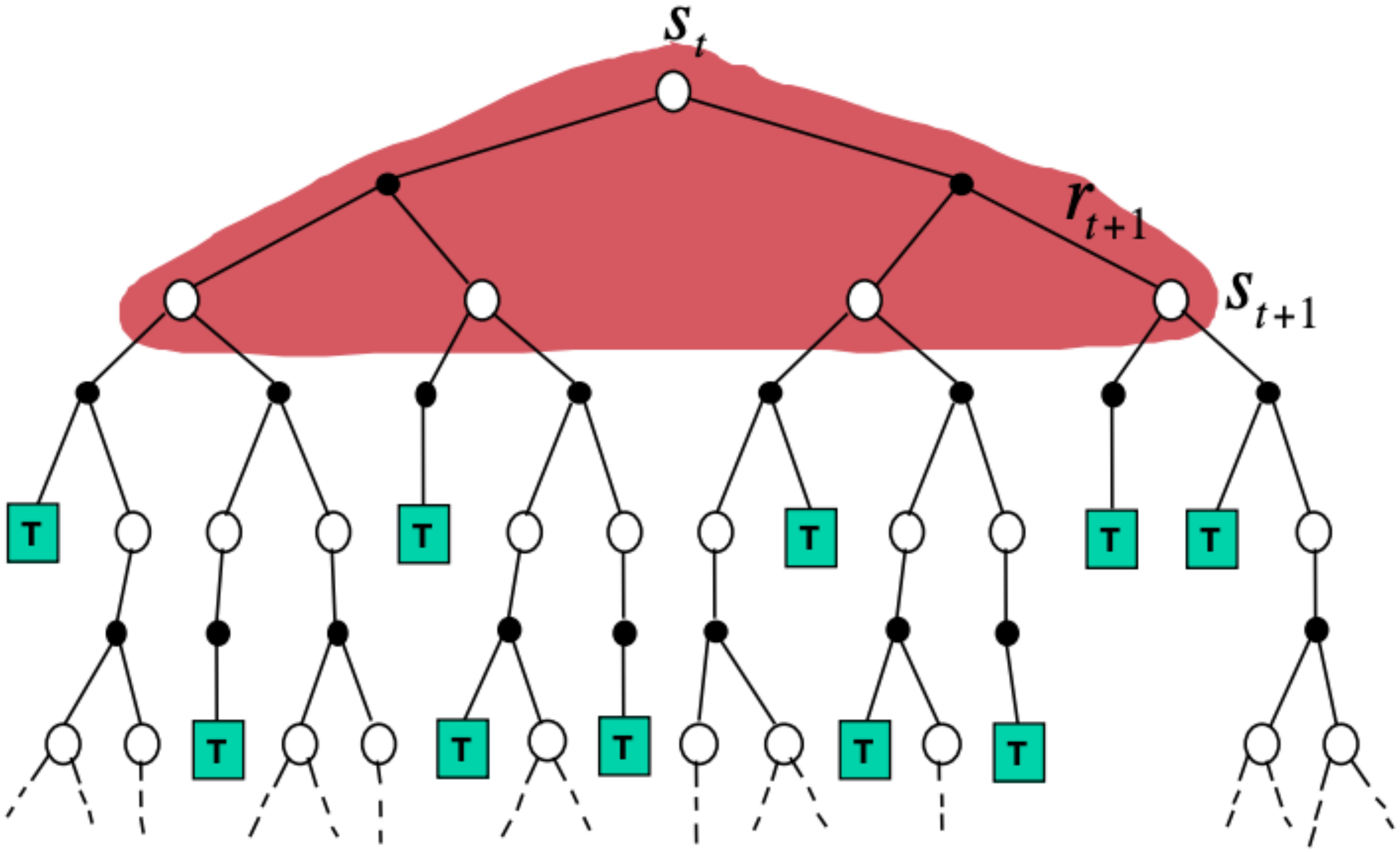
$$V_{k+1}(s) = \max_a \left[r + \gamma \sum_{s'} p(s' \mid s, a) V_k(s') \right]$$



Source

Recap: Policy Evaluation

$$V_{k+1}(s) = \sum_{a} \pi(a \mid s) \sum_{s'} p(s' \mid s, a)\left[r + \gamma V_k(s')\right]$$

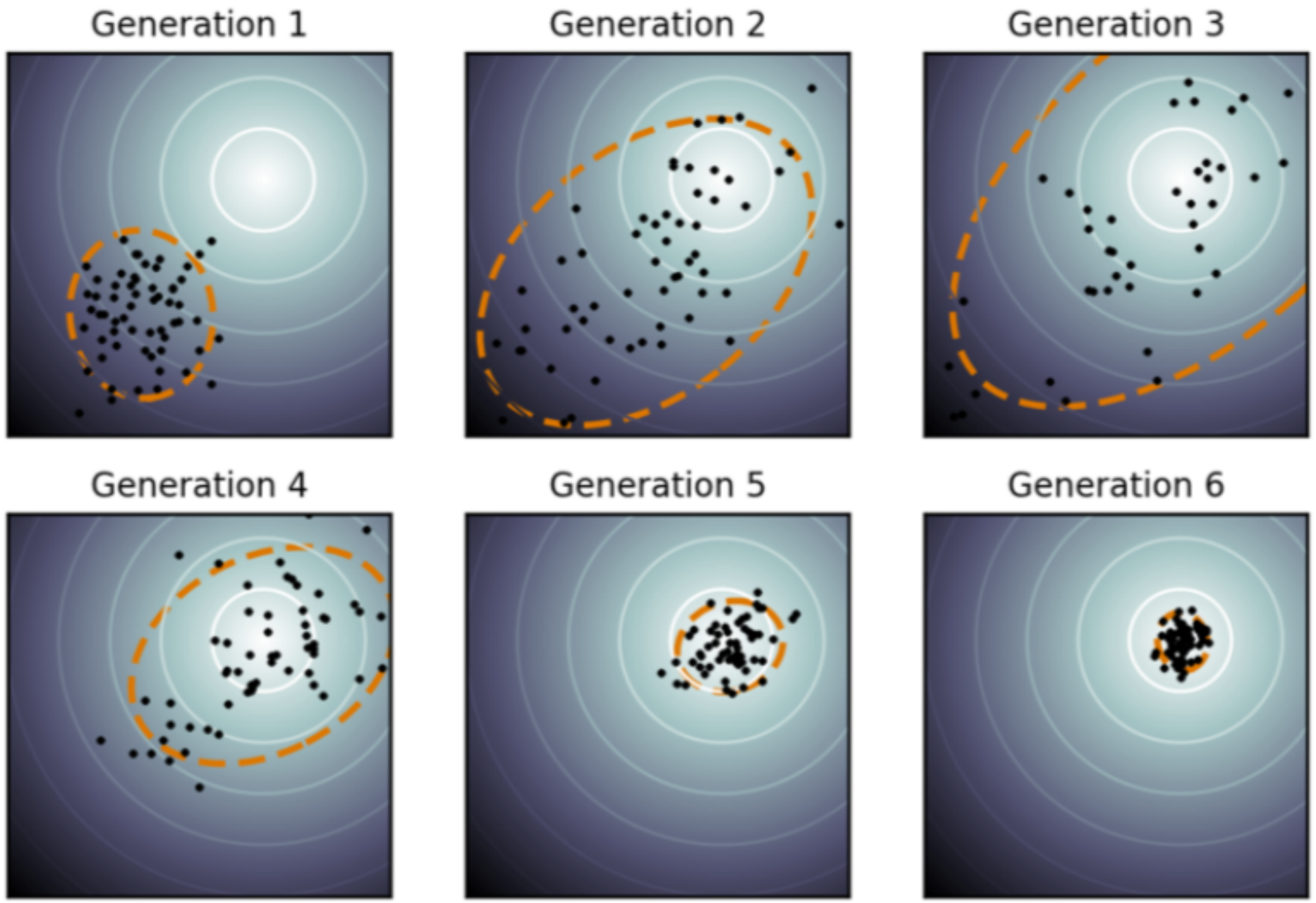


Source

Recap: Policy Improvement

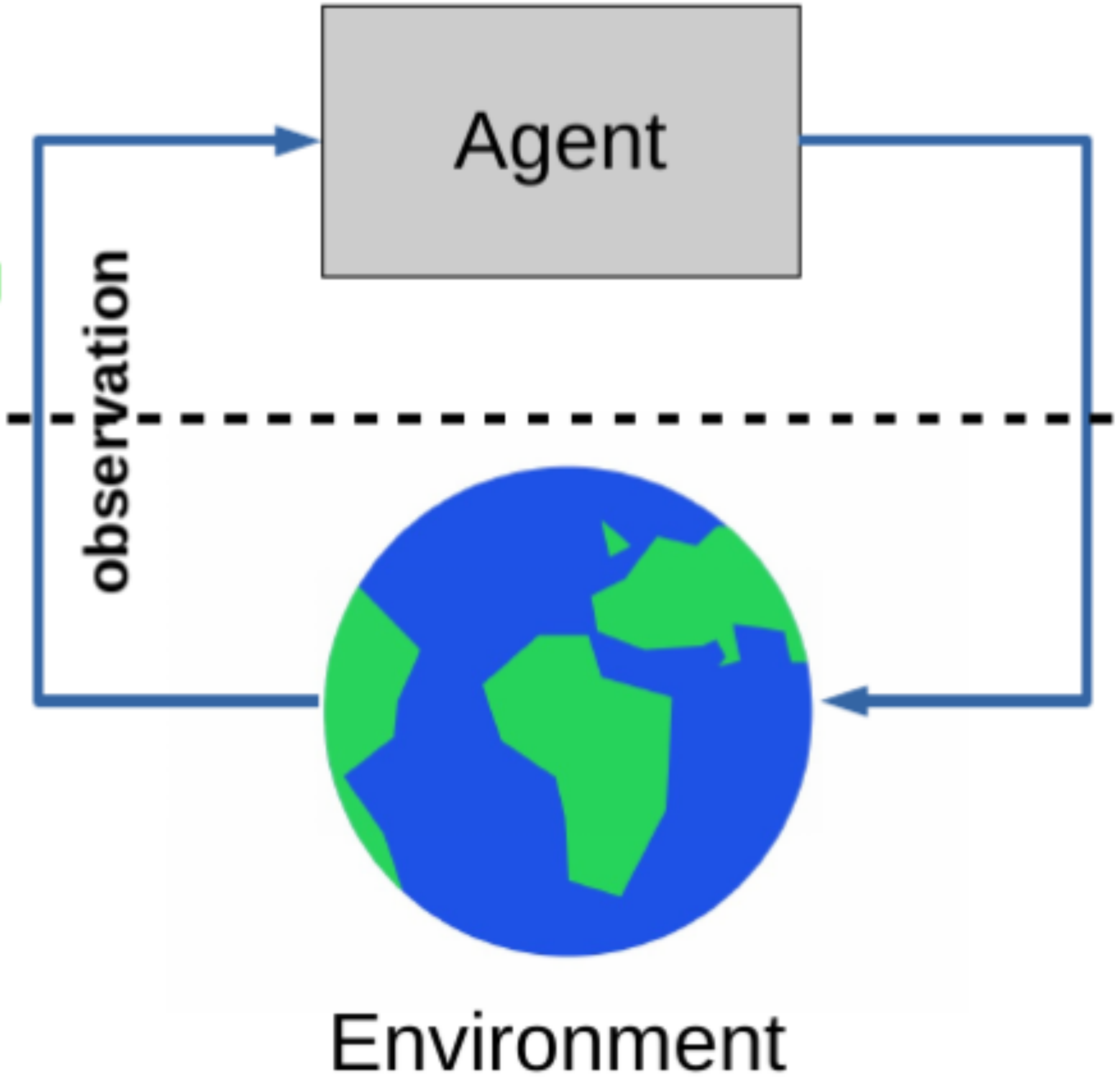
1. If Q_k is known: $\pi(s) = argmax_a Q_k(s, a)$
2. If V_k is known: $\pi(s) = argmax_a \sum_{s'} p(s' \mid s, a)[r + \gamma V_k(s')]$

Recap: Evolution Strategies



https://lilianweng.github.io/posts/2019-09-05-evolution-strategies/

Decision Processes



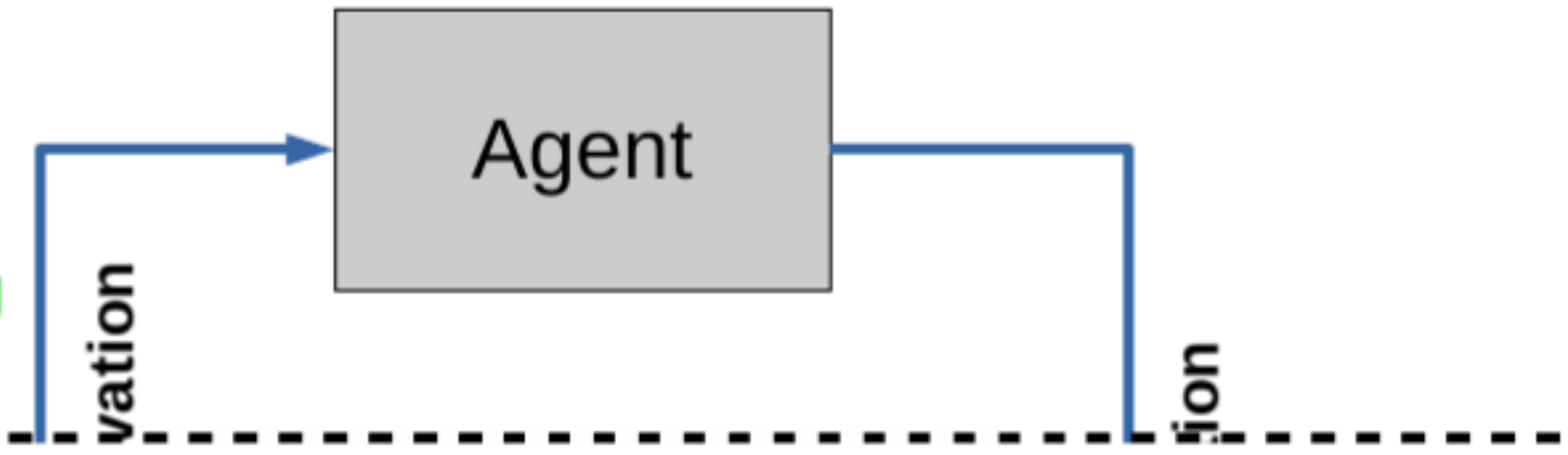
Can do anything

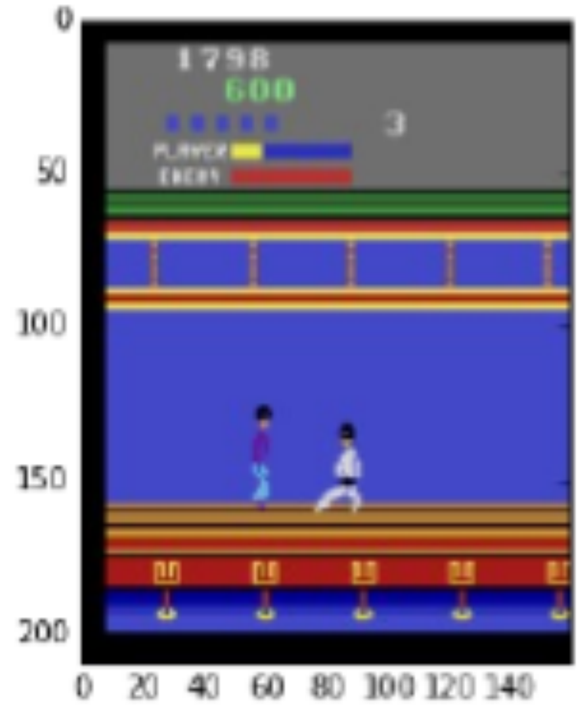
Can't even see
(worst case)

Source

Decision Processes

Can do anything





Source

Policy Improvement

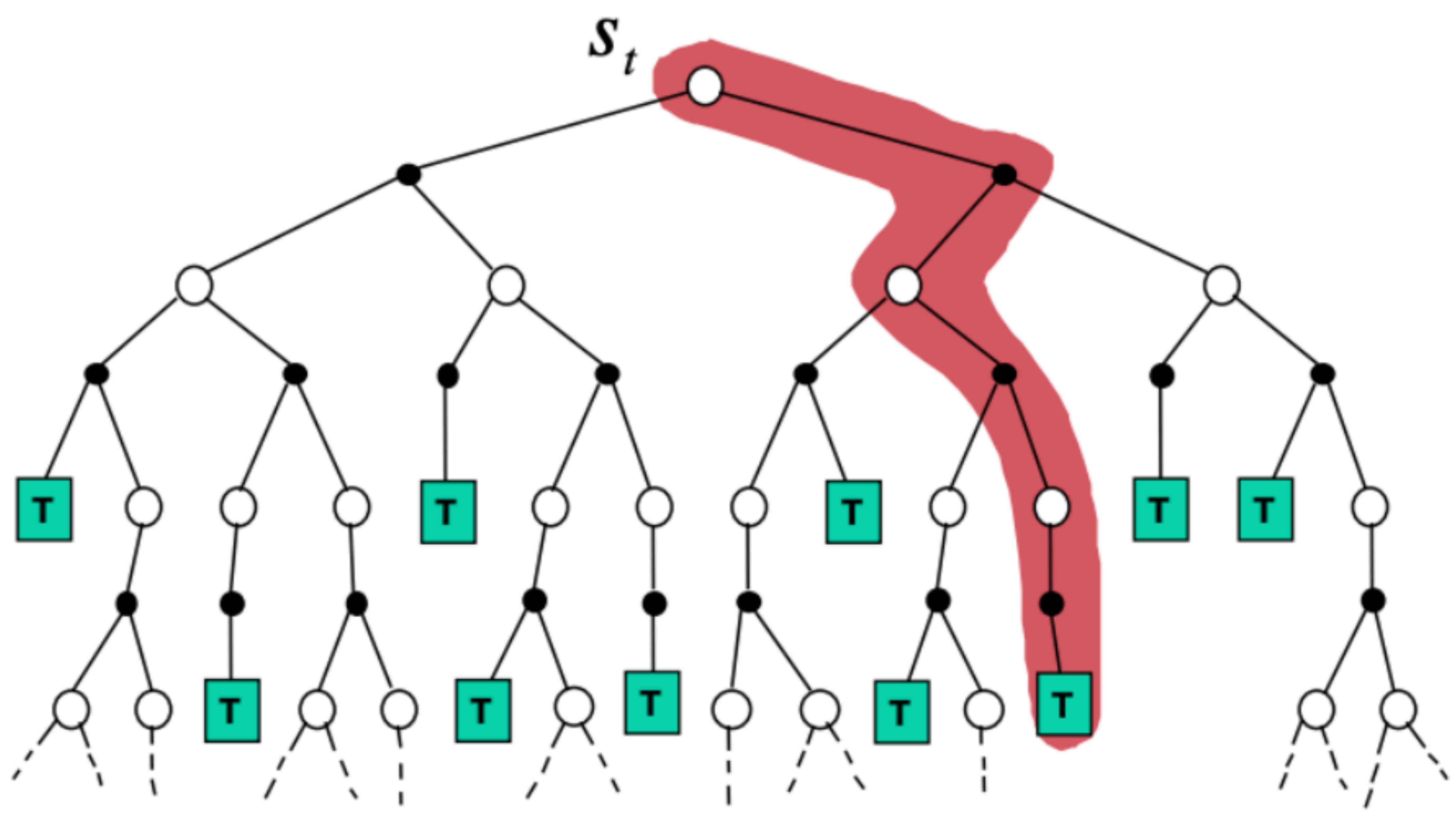
1. If Q_k is known: $\pi(s) = argmax_a Q_k(s, a)$
2. If V_k is known: $\pi(s) = argmax_a \sum_{s'} \cancel{p(s'|s, a)}[r + \gamma V_k(s')]$

No more available

Monte-Carlo Policy Evaluation

$$\tau_k = \{s_{k0} = s, a_{k1} = a, s_{k1}, a_{k1}, \ldots\}, G(\tau_k) = \sum_{t=0}^{T} \gamma^t r_{ko}$$

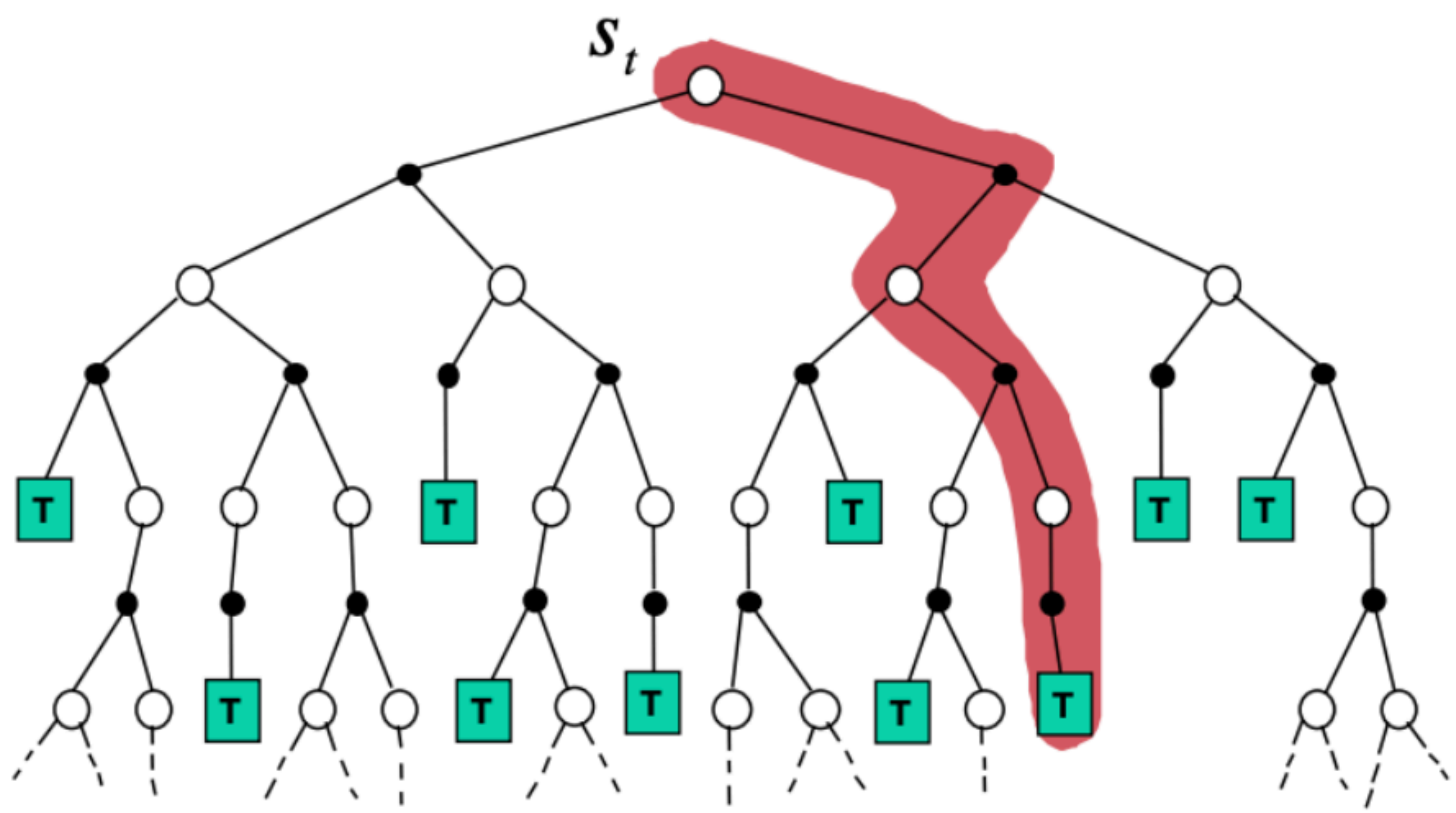
$$Q(s,a) \approx \frac{1}{N}\sum_{k=1}^{N} G(\tau_k) = \frac{N-1}{N}\sum_{k=1}^{N-1} G(\tau_k) + \frac{1}{N}G(\tau_N)$$



Monte-Carlo Policy Evaluation

$$\tau_k = \{s_{k0} = s, a_{k1} = a, s_{k1}, a_{k1}, \dots\}, G(\tau_k) = \sum_{t=0}^{T} \gamma^t r_{ko}$$

$$Q(s,a) \approx \frac{1}{N}\sum_{k=1}^{N} G(\tau_k) = \frac{N-1}{N}\sum_{k=1}^{N-1} G(\tau_k) + \frac{1}{N} G(\tau_N)$$



Pros:

- Unbiased
- Convergence's guarantees

Cons:

- High variance
- Must complete the episodes

Bellman Equations

Bellman expectation equations:

$$V^{\pi}(s) = \mathbb{E}_{a,s'}\left[r(s,a) + \gamma V^{\pi}(s')\right]$$

$$Q^{\pi}(s,a) = \mathbb{E}_{s',a'}\left[r + \gamma Q^{\pi}(s',a')\right]$$

Bellman optimality equations:

$$V^{*}(s) = \max_{a} \mathbb{E}_{s'}[r + \gamma V^{*}(s')]$$

$$Q^{*}(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^{*}(s',a')\right]$$

Stochastic Approximation

https://en.wikipedia.org/wiki/Stochastic_approximation

- We would like to estimate $\theta^* = \mathbb{E}[X]$
- Replace it with the following iterative procedure:

$$\theta_{k+1} = \theta_k - \alpha_k[\theta_k - X_k]$$

Stochastic Approximation

https://en.wikipedia.org/wiki/Stochastic_approximation

- We would like to estimate $\theta^* = \mathbb{E}[X]$
- Replace it with the following iterative procedure:

$$\theta_{k+1} = \theta_k - \alpha_k[\theta_k - X_k] = (1 - \alpha_k)\theta_k + \alpha_k X_k$$

Stochastic Approximation

https://en.wikipedia.org/wiki/Stochastic_approximation

- We would like to estimate $\theta^* = \mathbb{E}[X]$
- Replace it with the following iterative procedure:

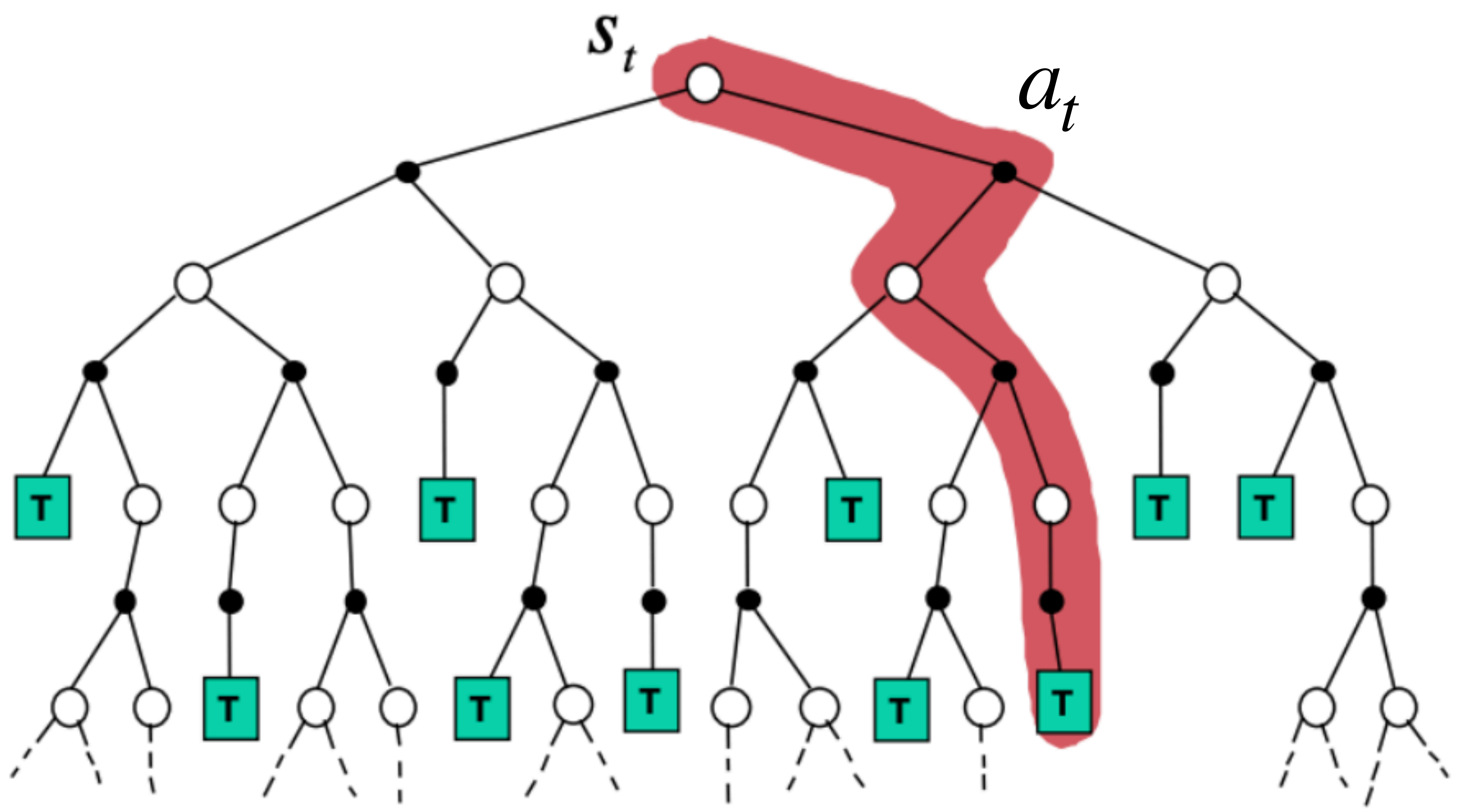
$$\theta_{k+1} = \theta_k - \alpha_k[\theta_k - X_k] = (1 - \alpha_k)\theta_k + \alpha_k X_k$$

Robbins–Monro theorem:

- $\sum_{k=0}^{+\infty} \alpha_k = +\infty, \sum_{k=0}^{+\infty} \alpha_k^2 < +\infty$ $\Longrightarrow$ $\theta_k \to \theta^*$ in squared mean
- Some technical conditions

Monte-Carlo Policy Evaluation

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(G_k - Q_k(s,a))$$



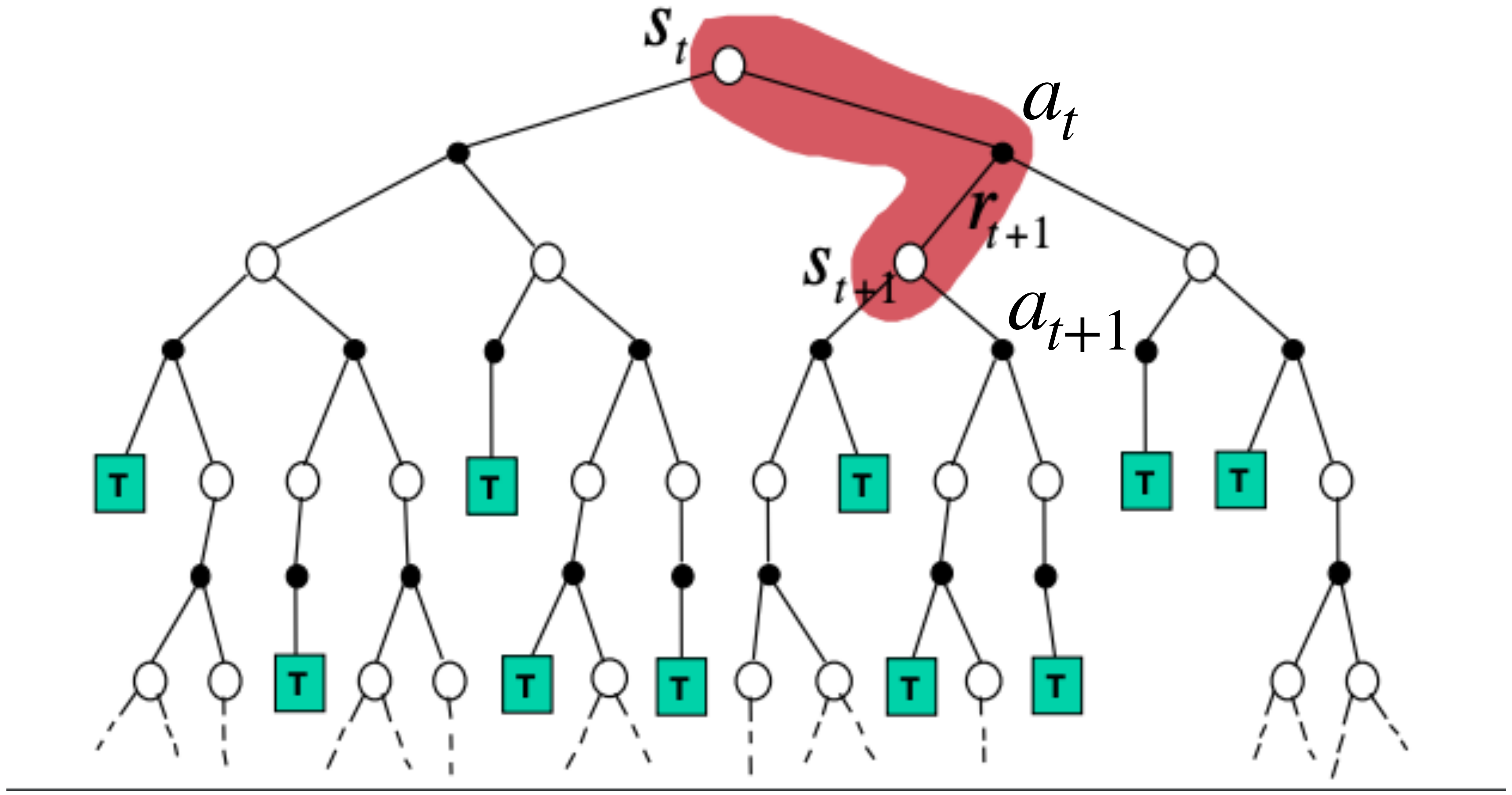
Temporal Difference Learning

Target

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)\left(\boxed{r + \gamma Q_k(s',a')} - Q_k(s,a)\right)$$

Temporal difference

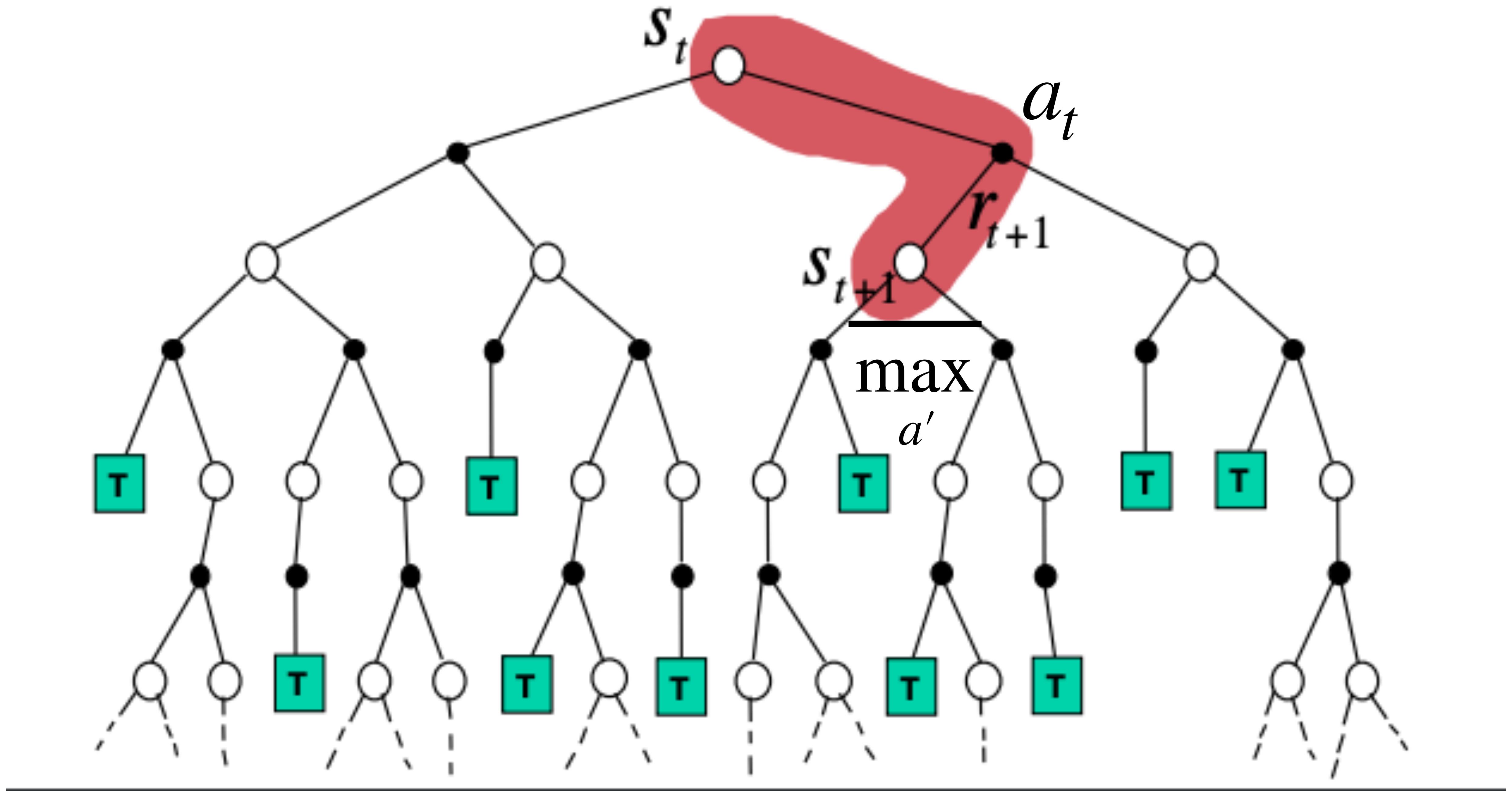
$$s' \sim p(\,.\,|\,s,a),\; a' \sim \pi(\,.\,|\,s)$$



Temporal Difference Learning

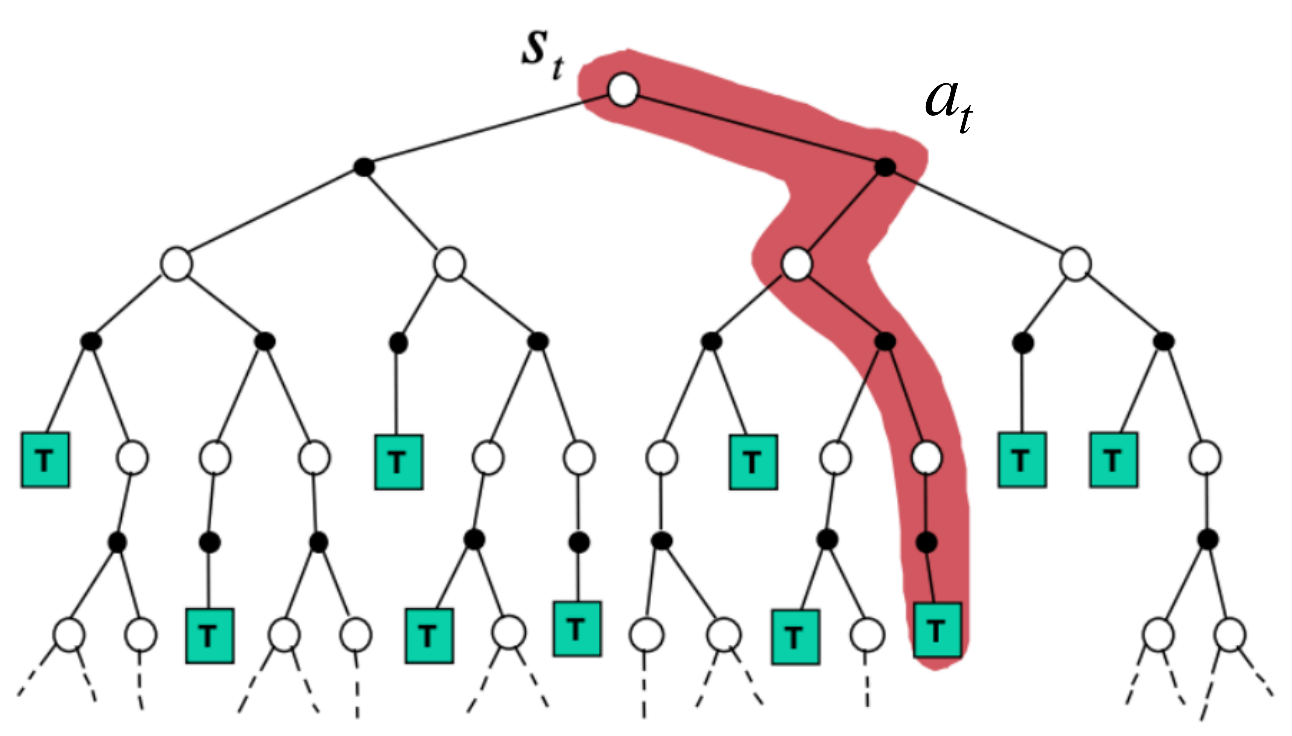
$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(r + \gamma \max_{a'} Q_k(s',a') - Q_k(s,a))$$

$$s' \sim p(\,.\,|\,s,a)$$

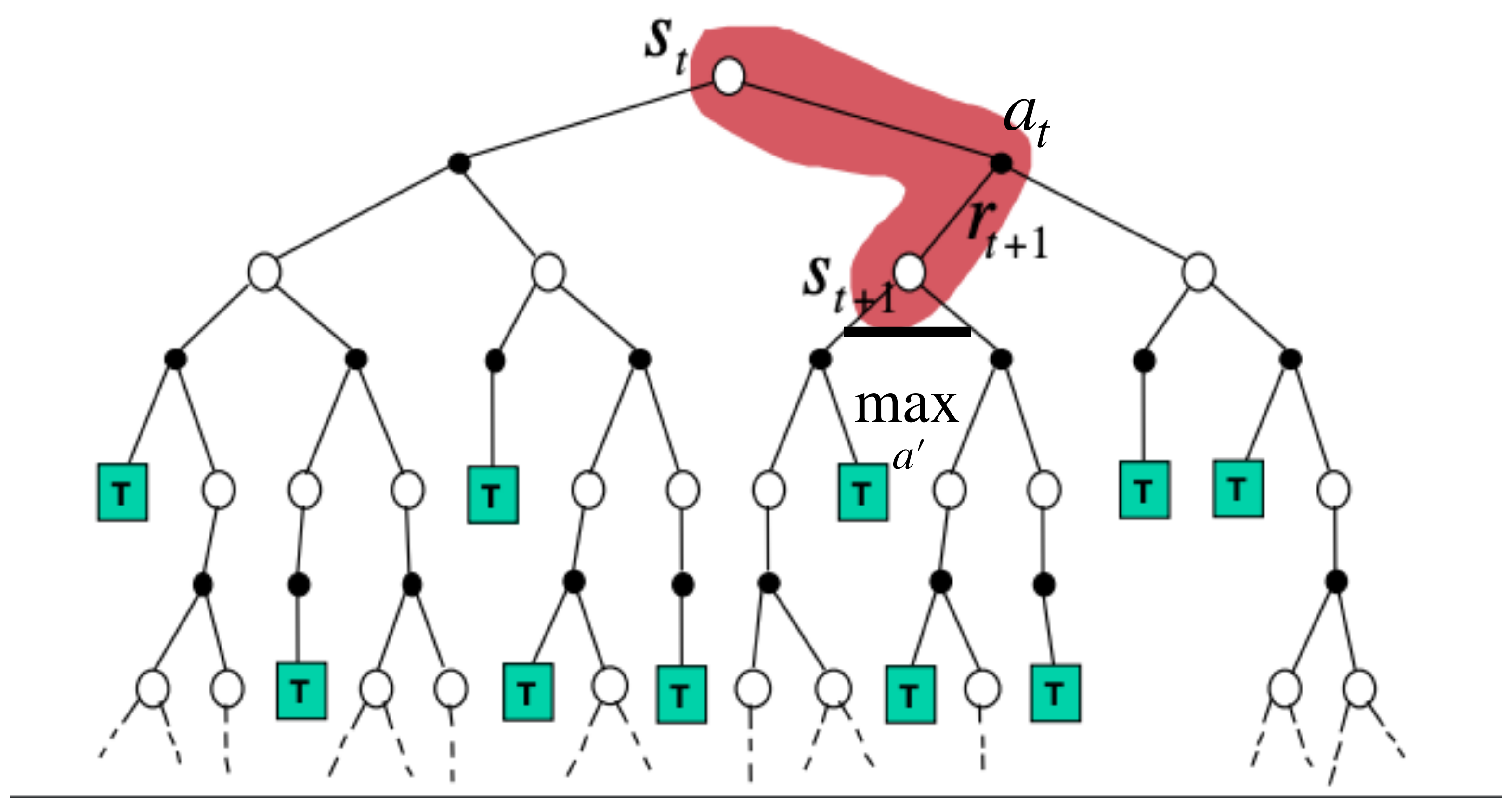


Policy Evaluation

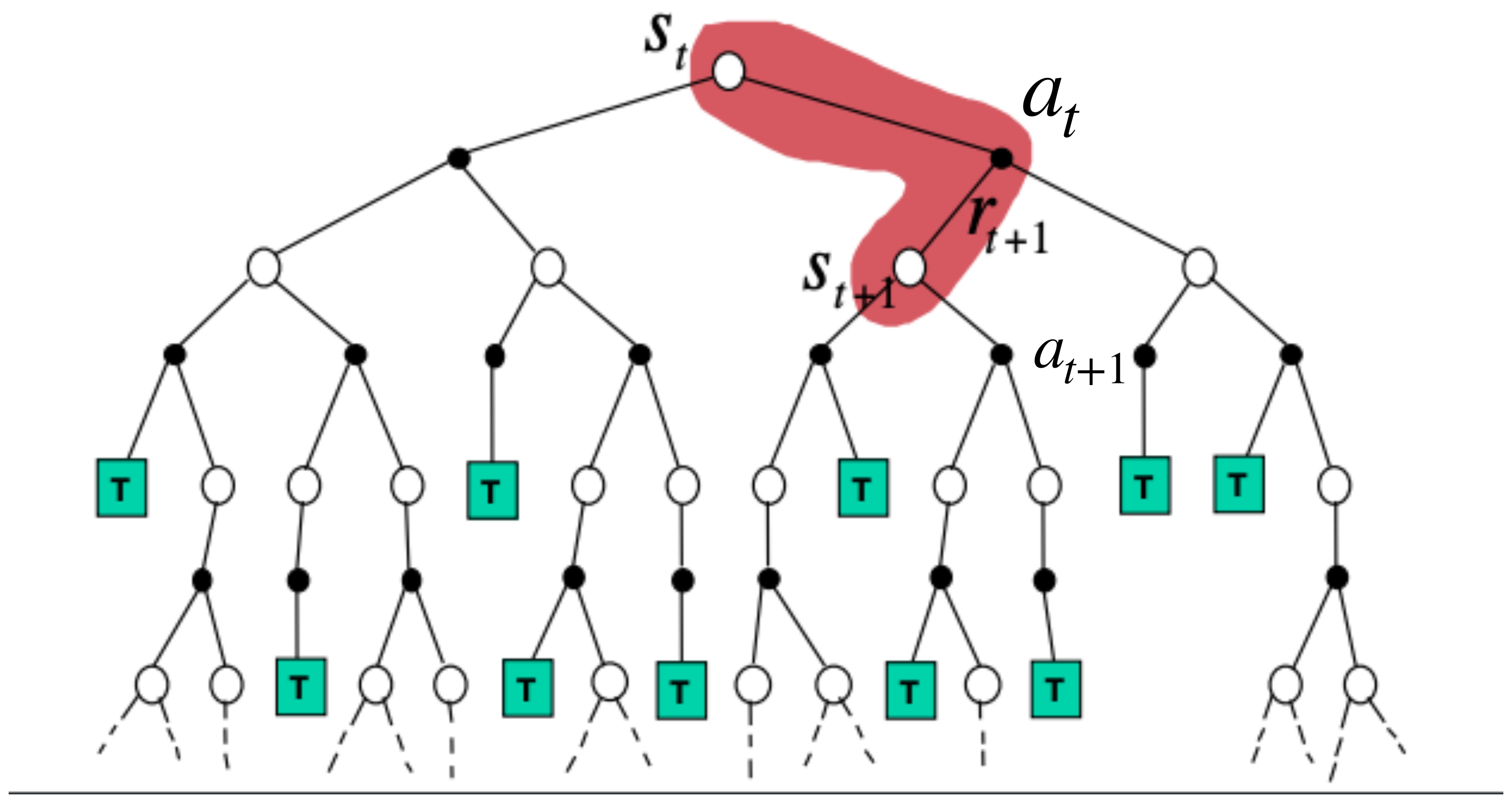
$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(G_k - Q_k(s,a))$$



$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(r + \gamma \max_{a'} Q_k(s',a') - Q_k(s,a))$$



$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(r + \gamma Q_k(s',a') - Q_k(s,a))$$



Temporal Difference Learning

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(y_Q - Q_k(s,a))$$

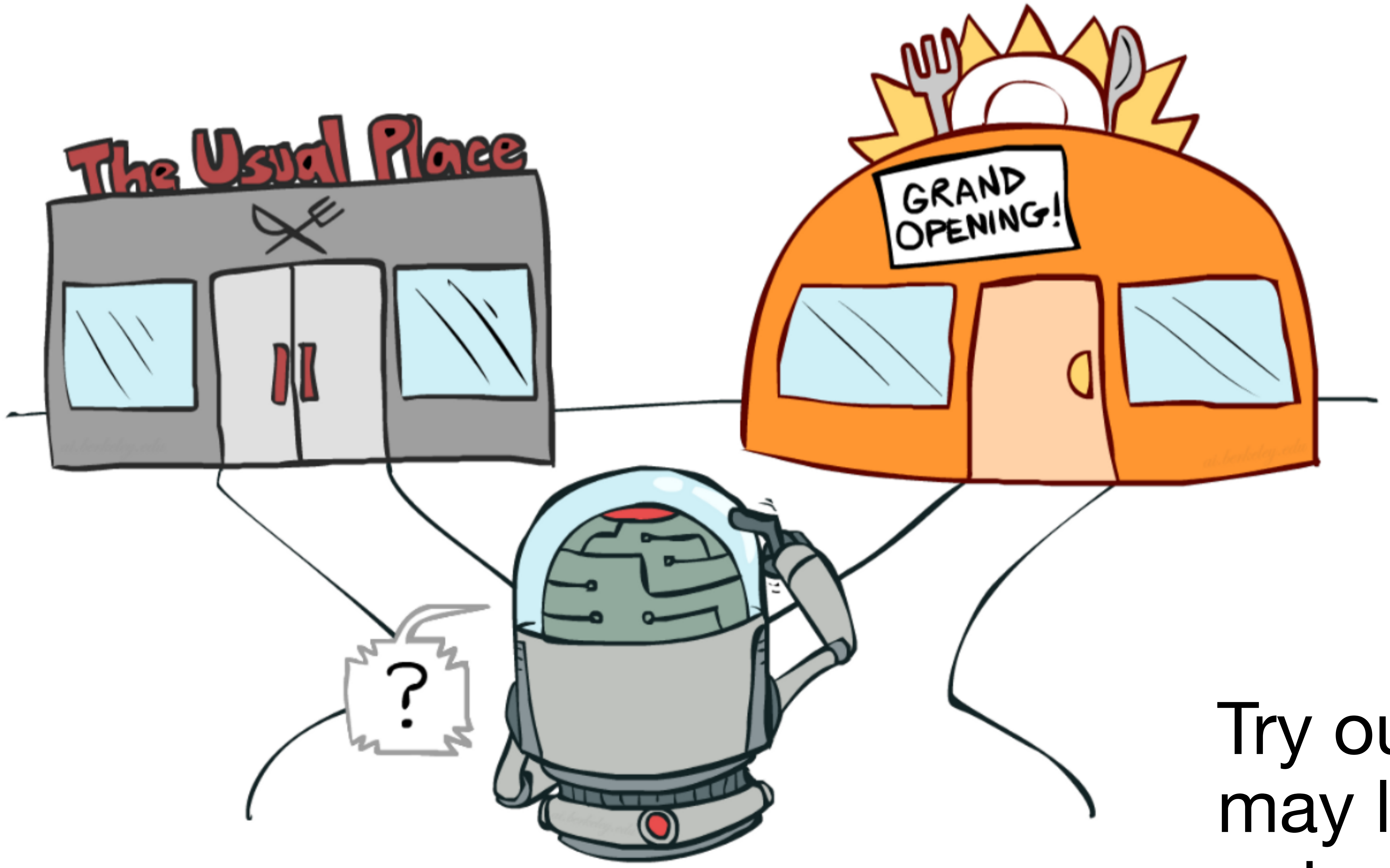
Infinite visitation:

$$\sum_{k \geq 0} \alpha_k(s,a) = +\infty$$

Stochastic policy $\mu(a \mid s)$ s.t.

$\forall s, a : \mu(a \mid s) > 0$:

Exploration-Exploitation Trade-off



Choose the best option based on current knowledge (which may be incomplete)

Try out new options that may lead to better outcomes in the future at the expense of an exploitation opportunity

Exploration vs Exploitation

- Uniform policy $\mu(a \mid s)$

- Greedy policy:
  $\pi(s) = argmax_a Q(s, a)$

Exploration vs Exploitation

- Uniform policy $\mu(a \mid s)$
- Greedy policy:
  $\pi(s) = argmax_a Q(s, a)$
- ε-greedy policy:

$$\mu(\,.\,|s) = \begin{cases} \text{select random action with probabily } \varepsilon \\ argmax_a Q(s, a) \text{ with probabily } 1 - \varepsilon \end{cases}$$

Exploration vs Exploitation

- Uniform policy $\mu(a \mid s)$

- Greedy policy:
  $\pi(s) = argmax_a Q(s, a)$

- ε-greedy policy:

$$\mu(\,.\mid s) = \begin{cases} \text{select random action with probabily } \varepsilon \\ argmax_a Q(s, a) \text{ with probabily } 1 - \varepsilon \end{cases}$$

- Boltzmann policy:

$$\mu(\,.\mid s) = \text{softmax}\left(\frac{Q(s,\,.\,)}{\alpha}\right)$$

Exploration vs Exploitation

- Uniform policy $\mu(a \mid s)$
- Greedy policy:
  $\pi(s) = argmax_a Q(s, a)$
- ε-greedy policy:

$$\mu(\,.\,|s) = \begin{cases} \text{select random action with probabily } \varepsilon \\ argmax_a Q(s,a) \text{ with probabily } 1 - \varepsilon \end{cases}$$

- Boltzmann policy:

$$\mu(\,.\,|s) = \text{softmax}\left(\frac{Q(s,\,.\,)}{\alpha}\right)$$

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(r + \gamma \max_{a'} Q_k(s',a') - Q_k(s,a))$$

$$s' \sim p(\,.\,|\,s,a)$$

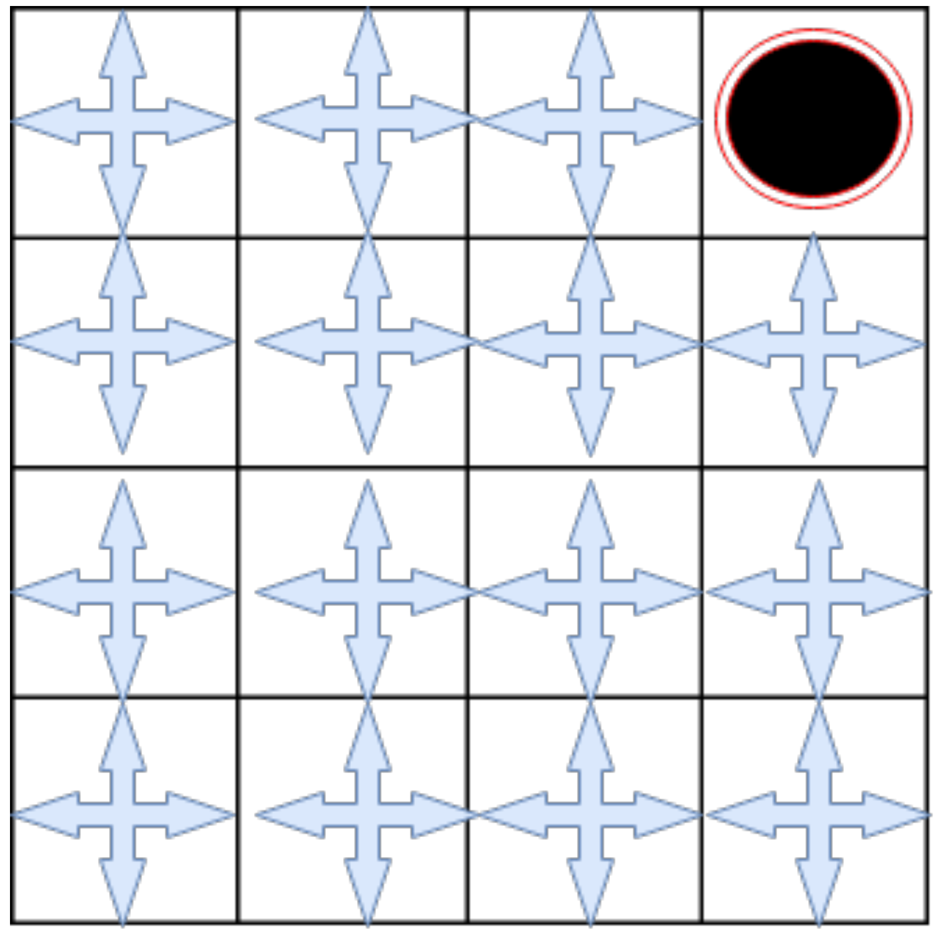
$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(r + \gamma Q_k(s',a') - Q_k(s,a))$$

$$s' \sim p(\,.\,|\,s,a), a' \sim \pi(\,.\,|\,s)$$

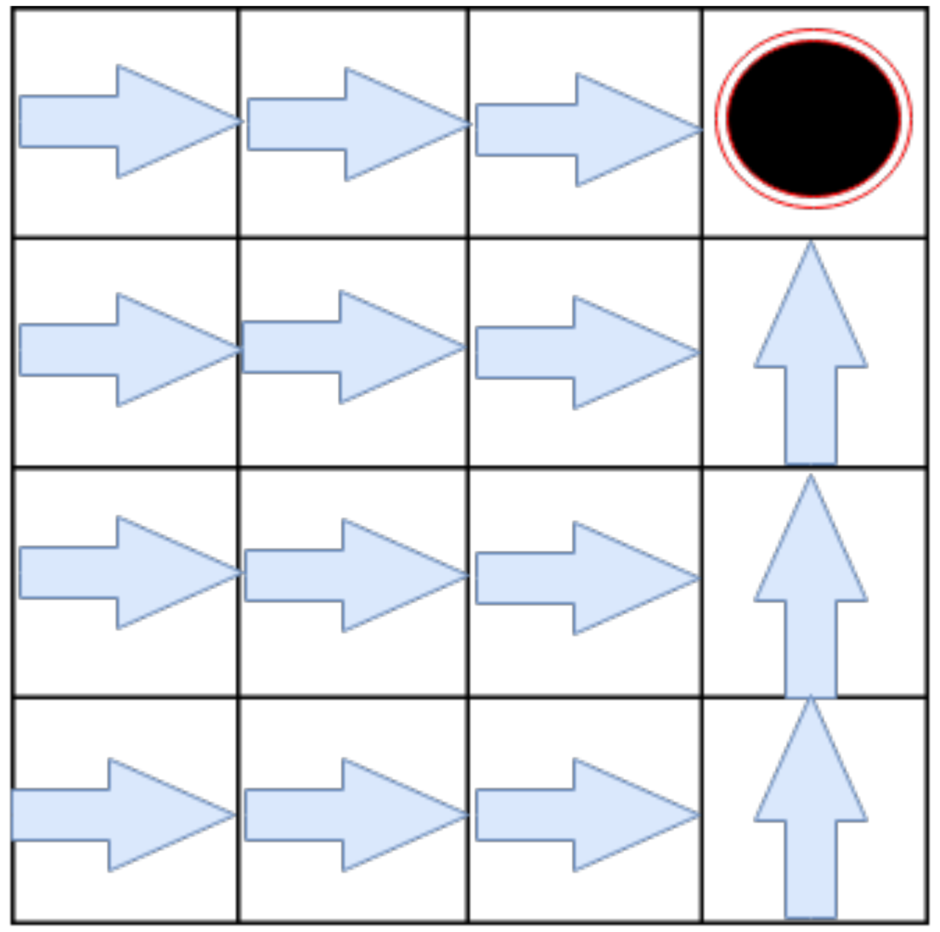
On-policy vs Off-Policy

On-policy learning

- Learn about behaviour policy μ from experience sampled from μ



Behavior Policy



Target Policy

Off-policy learning

- Learn about target policy π from experience sampled from μ
- Learn from observing humans or other agents (e.g., from logged data)
- Learn about multiple policies while following one policy
- Learn about greedy policy while following exploratory policy
- Reuse experience from old policies (e.g., from your own past experience)

Q-Learning

- Parameters: ε, α
- Initialise $Q_0(s, a) \; \forall s, a$
- For $k = 0, 1, \ldots$

1. $a \sim \mu(\,.\,|\,s) = \begin{cases} \text{select random action with probabily } \varepsilon \\ argmax_a Q_k(s,a) \text{ with probabily } 1 - \varepsilon \end{cases}$
2. Observe r and $s' \sim p(\,.\,|\,s, a)$
3. $Q_{k+1}(s,a) = Q_k(s,a) + \alpha(r + \gamma \max_{a'} Q_k(s', a') - Q_k(s,a))$

Q-Learning

- Parameters: ε, α
- Initialise $Q_0(s, a)\ \forall s, a$
- For $k = 0,1,\ldots$

1. $a \sim \mu(\,.\,|\,s) = \begin{cases} \text{select random action with probabily } \varepsilon \\ argmax_a Q_k(s, a) \text{ with probabily } 1 - \varepsilon \end{cases}$
2. Observe r and $s' \sim p(\,.\,|\,s, a)$
3. $Q_{k+1}(s, a) = Q_k(s, a) + \alpha(r + \gamma \max_{a'} Q_k(s', a') - Q_k(s, a))$

Q-Learning learns Q^* using samples from another policy!

SARSA

- Parameters: ε, α
- Initialise $Q_0(s, a)\ \forall s, a$
- For $k = 0,1,\ldots$

1. $a \sim \mu(\,.\,|\,s) = \begin{cases} \text{select random action with probabily } \varepsilon \\ argmax_a Q_k(s, a) \text{ with probabily } 1 - \varepsilon \end{cases}$

2. Observe r and $s' \sim p(\,.\,|\,s, a)$

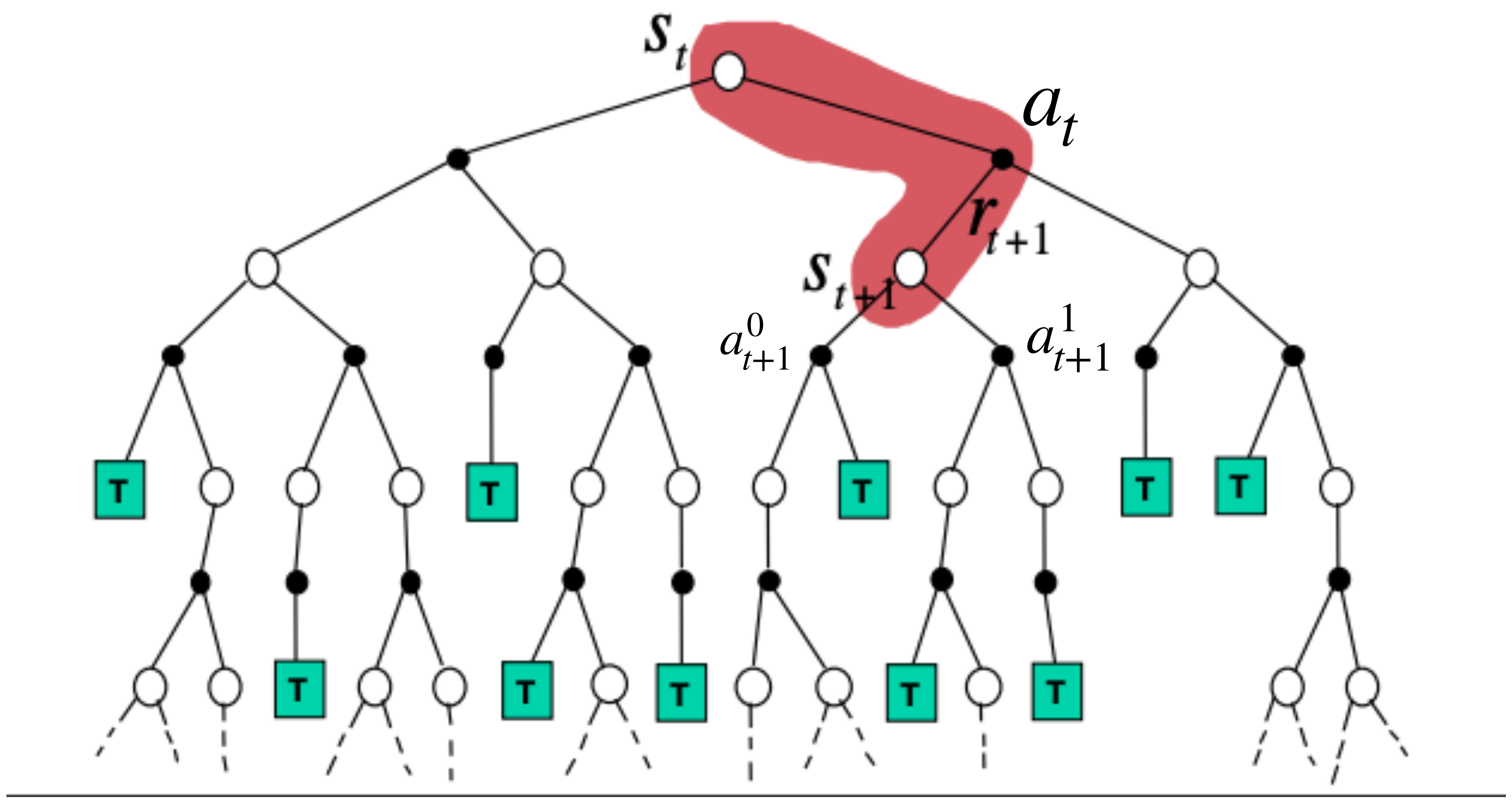
3. $a' \sim \mu(\,.\,|\,s') = \begin{cases} \text{select random action with probabily } \varepsilon \\ argmax_a Q_k(s', a) \text{ with probabily } 1 - \varepsilon \end{cases}$

4. $Q_{k+1}(s, a) = Q_k(s, a) + \alpha(r + \gamma Q_k(s', a') - Q_k(s, a))$

Expected SARSA

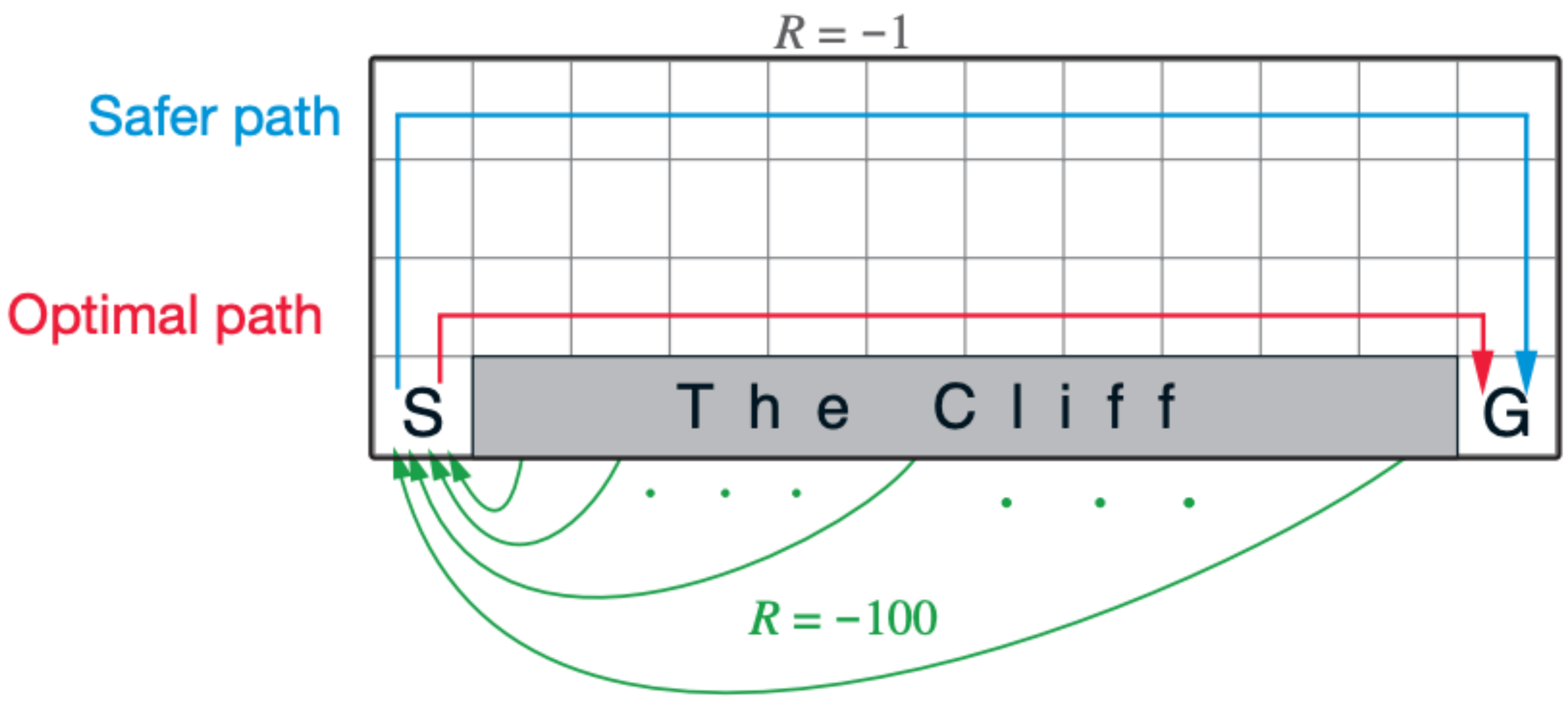
$$Q_{k+1}(s,a) \leftarrow Q_k(s,a) + \alpha(r + \gamma \mathbb{E}_{a' \sim \mu_k(.|s')} Q_k(s',a') - Q_k(s,a))$$

$$\mu_k(\,.\,|\,s') = \begin{cases} \text{select random action with probabily } \varepsilon \\ argmax_a Q_k(s',a) \text{ with probabily } 1-\varepsilon \end{cases}$$



Example: Cliff Walking

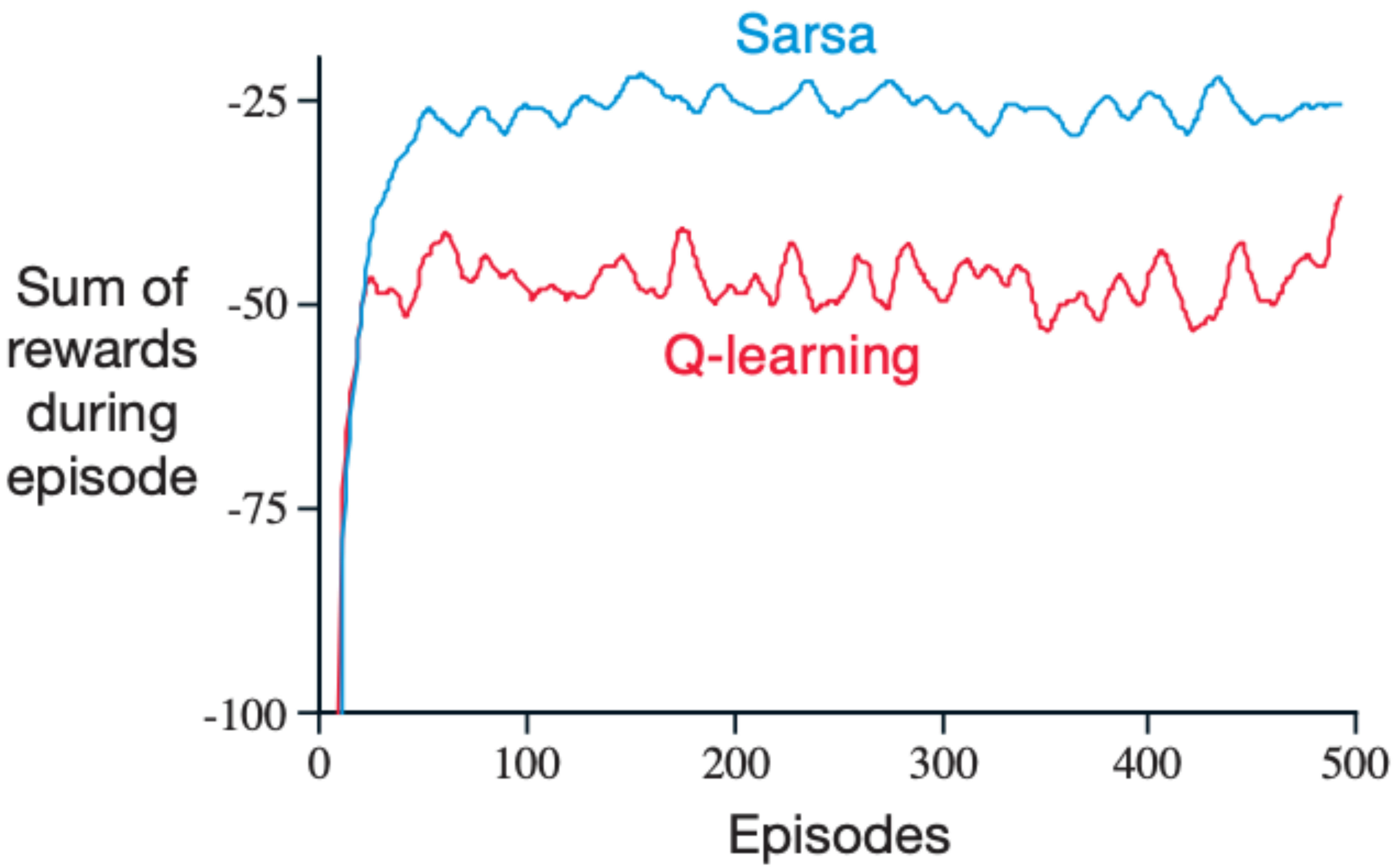
$\gamma = 1, \varepsilon = 0.1$. Agent gets -1 for each step.



Source

Which policy is learned by Q-learning and SARSA?

Example: Cliff Walking



Source

Of course, if ε were gradually reduced, both methods would asymptotically converge to the optimal policy.

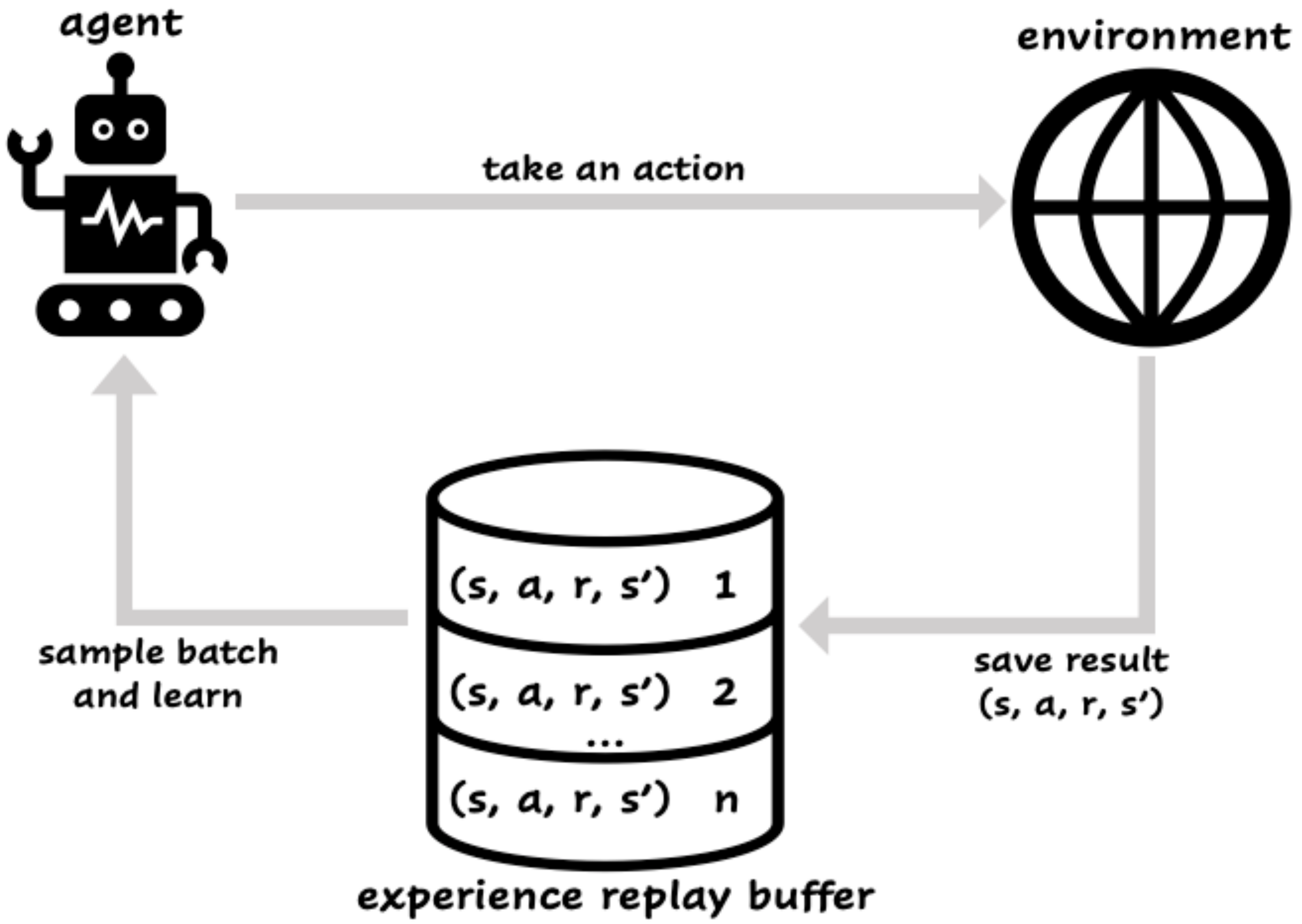
Experience Replay Buffer

Advantages:

- No need to revisit same states many times
- Make the estimators consistent with the current policy, update estimators
- Decorrelate update samples to maintain i.i.d. assumption

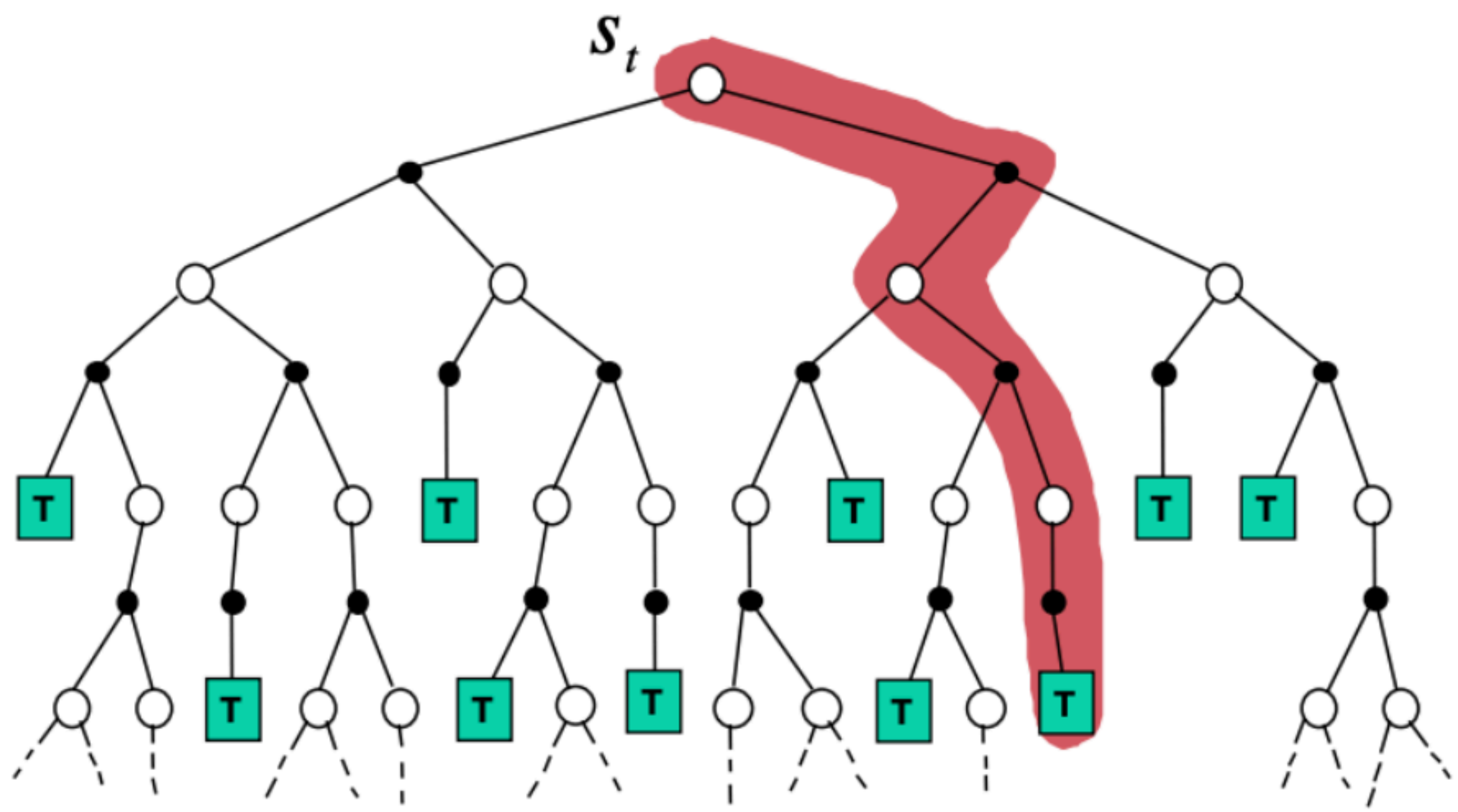
Disadvantages:

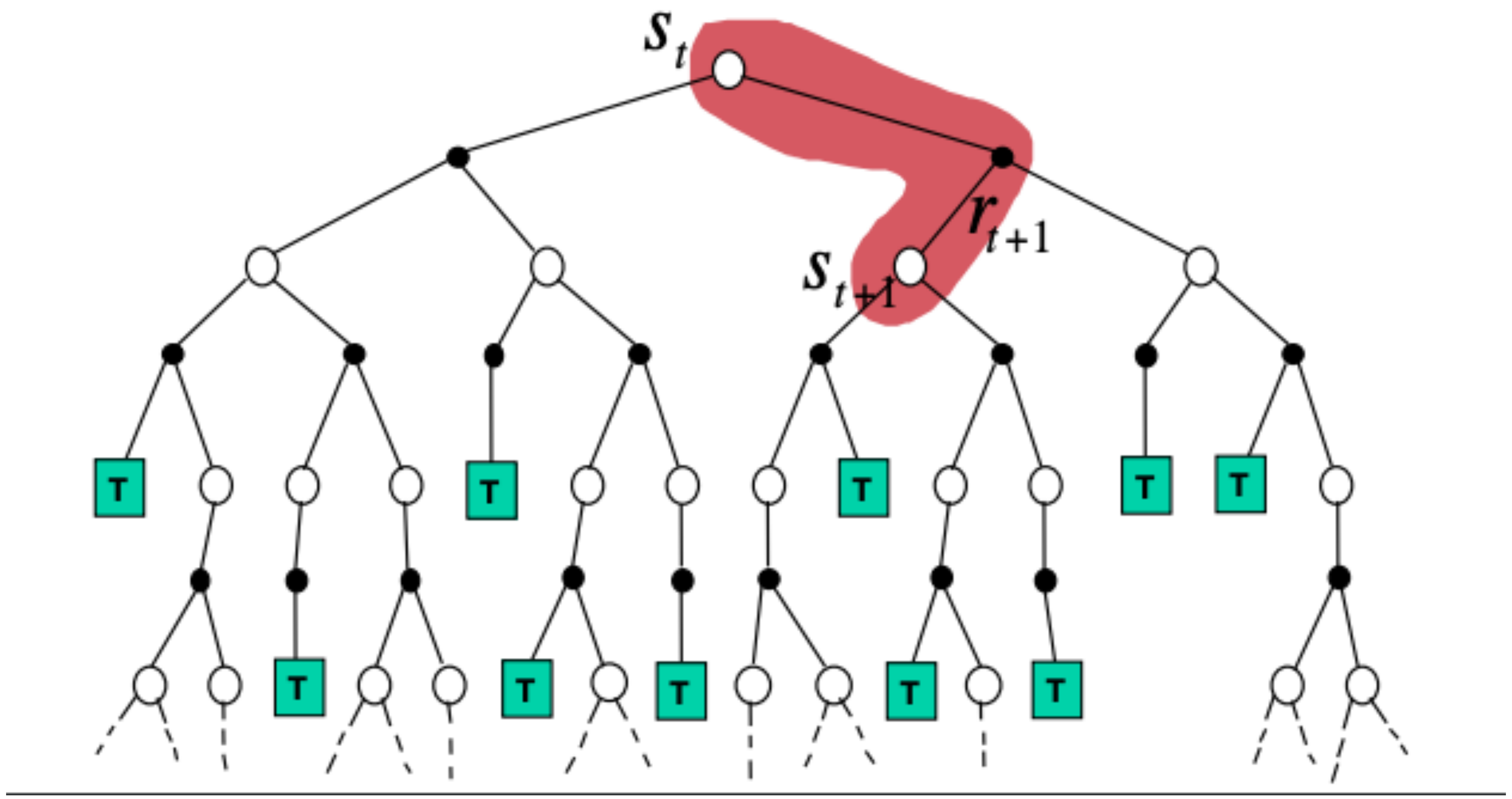
- Not applicable for the on-policy learning



Bias-Variance Trade-off

$$Q(s,a) \leftarrow Q(s,a) + \alpha(y_Q - Q(s,a))$$





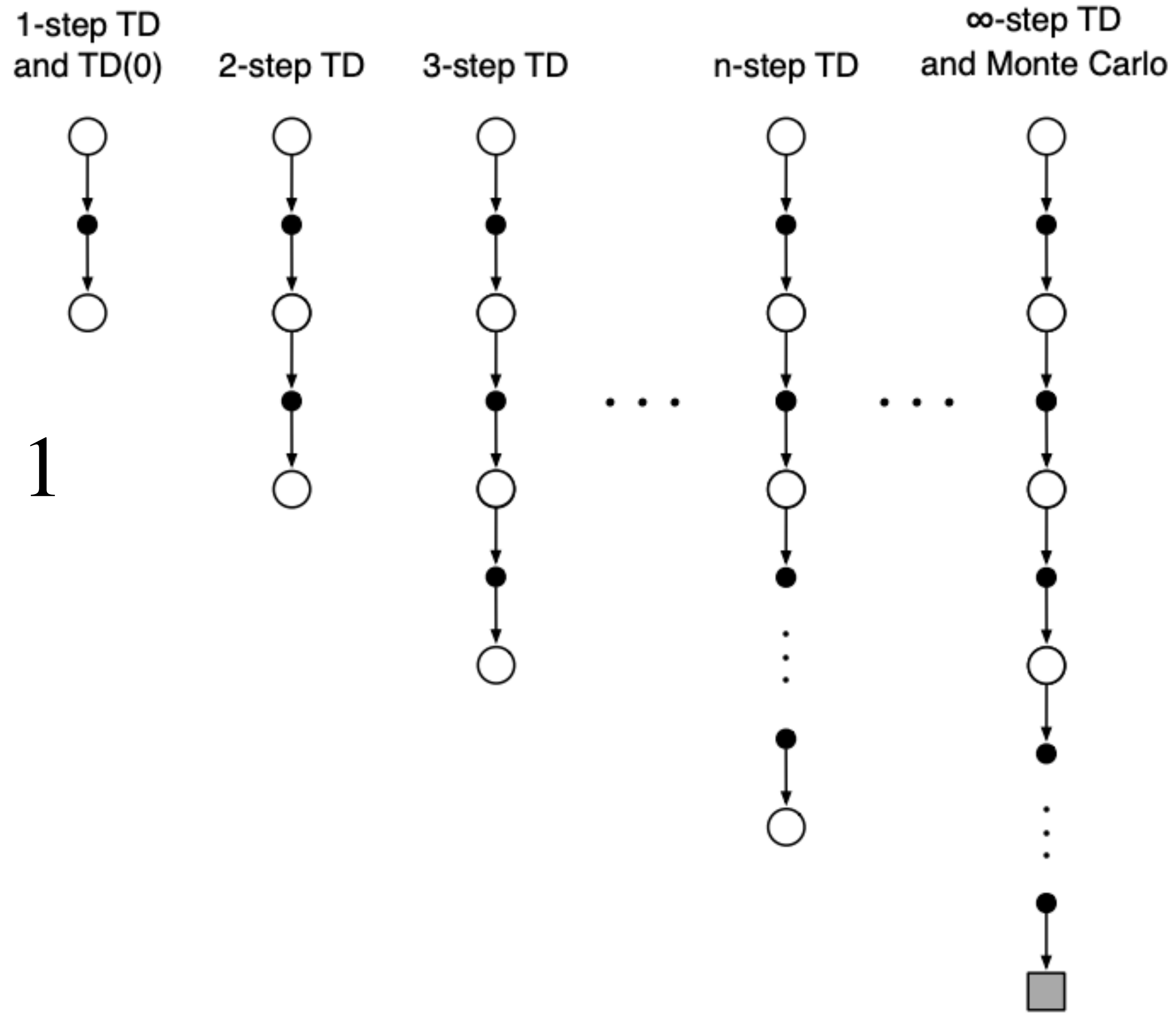
N-step SARSA

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha(y_Q^N - Q_k(s,a))$$

$$y_Q^N = r + \gamma r' + \gamma^2 r'' + \ldots + \gamma^N Q_k(s^{(N)}, a^{(N)})$$

$$s^{(n)} \sim p(\,.\,|\,s^{(n-1)}, a^{(n-1)}),\, a^{(n)} \sim \mu(a\,|\,s), n \geq 1$$

Ranging N we can control the trade-off between bias and variance.



Credit Assignment

At the time step t:

$$\delta_t^1 = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

$$\delta_t^2 = r_t + \gamma r_{t+1} + \gamma^2 Q(s_{t+2}, a_{t+2}) - Q(s_t, a_t)$$

$$\delta_t^N = r_t + \gamma r_{t+1} + \ldots + \gamma^N Q(s_{t+N}, a_{t+N}) - Q(s_t, a_t)$$

$$\delta_t^N = \sum_{n=0}^{N-1} \gamma^n \delta_{t+n}^1$$

Credit Assignment

At the time step t:

$$\delta_t^1 = r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

$$\delta_t^2 = r_t + \gamma r_{t+1} + \gamma^2 Q(s_{t+2}, a_{t+2}) - Q(s_t, a_t)$$

$$\delta_t^N = r_t + \gamma r_{t+1} + \ldots + \gamma^N Q(s_{t+N}, a_{t+N}) - Q(s_t, a_t)$$

$$\delta_t^N = \sum_{n=0}^{N-1} \gamma^n \delta_{t+n}^1$$

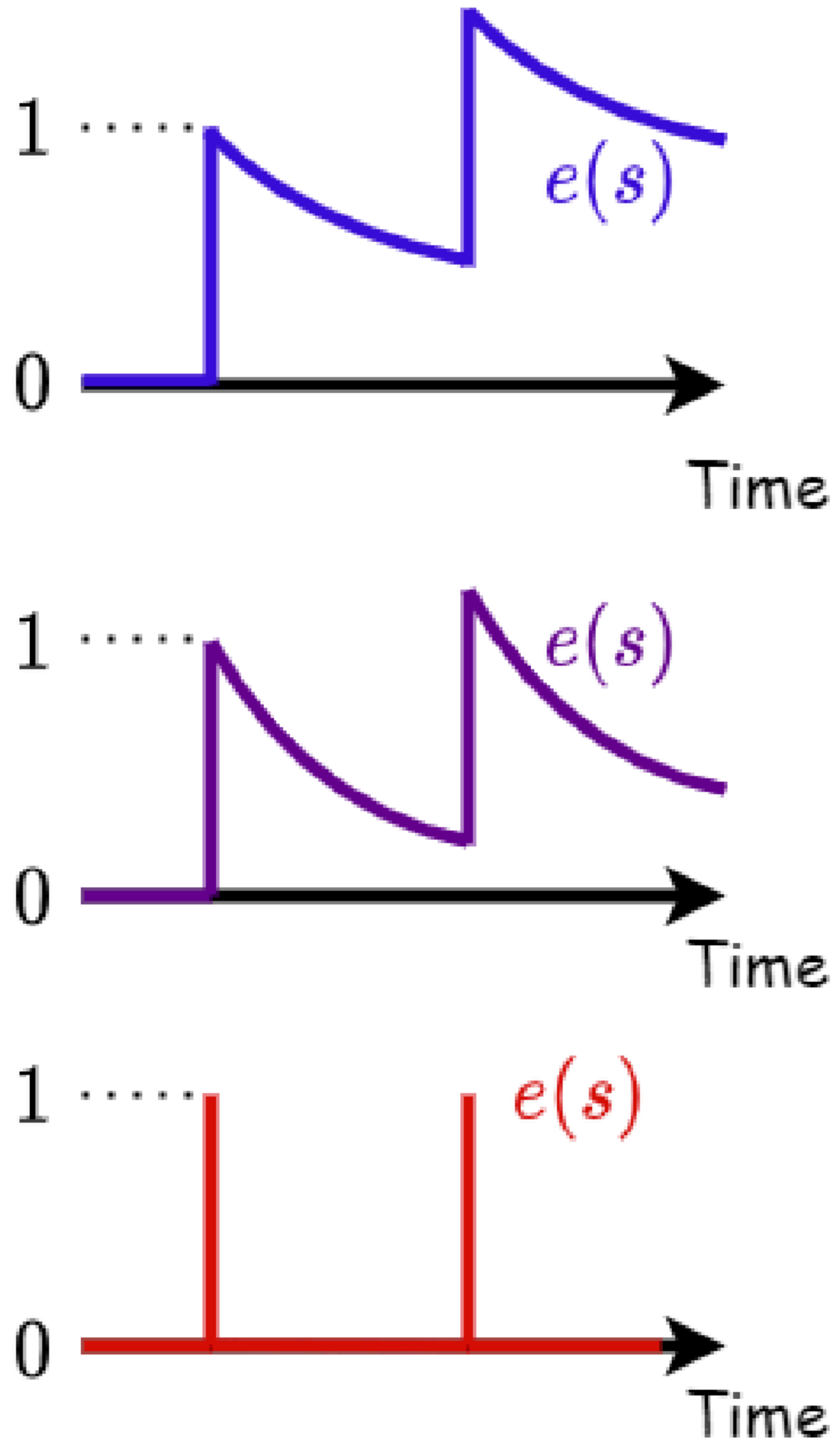
N-step SARSA: $Q(s, a) \leftarrow Q(s, a) + \alpha \delta_t^N$

1. $Q(s, a) \leftarrow Q(s, a) + \alpha \delta_t^1$
2. $Q(s, a) \leftarrow Q(s, a) + \alpha \gamma \delta_{t+1}^1$
3. ....

Eligibility Traces

$$e_0(s, a) = 0$$

$$e_t(s, a) = \gamma \lambda e_{t-1}(s, a) + \mathbb{I}[S_t = s, A_t = a]$$



$\lambda = 1$

$0 < \lambda < 1$

$\lambda = 0$

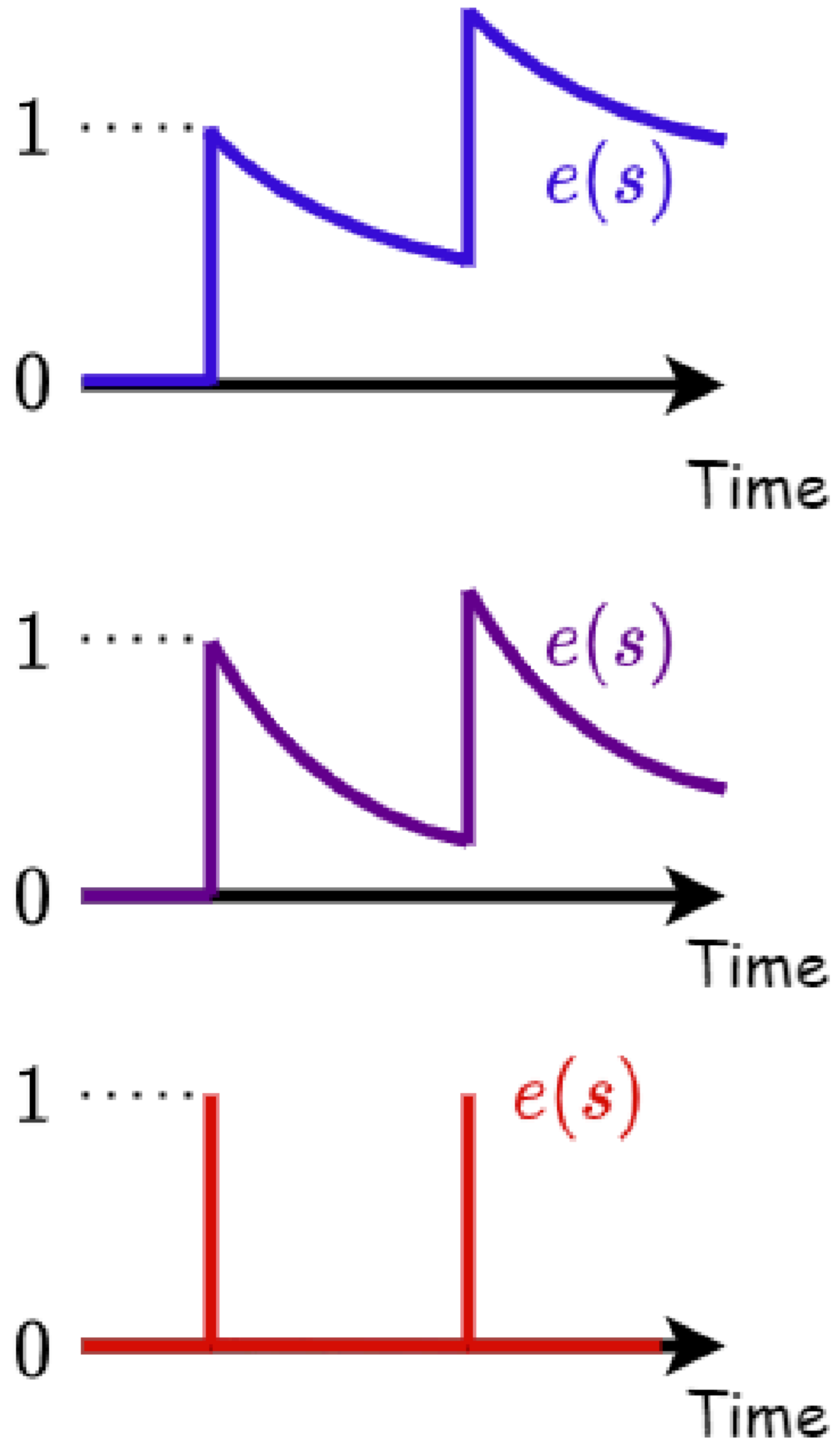
SARSA(λ)

$$e_0(s,a) = 0$$

$$e_t(s,a) = \gamma\lambda e_{t-1}(s,a) + \mathbb{1}[S_t = s, A_t = a]$$

1. $Q(s,a) \leftarrow Q(s,a) + \alpha e_1(s,a)\delta_0^1$
2. $Q(s,a) \leftarrow Q(s,a) + \alpha e_2(s,a)\delta_1^1$
3. ....

$$Q(s,a) \leftarrow Q(s,a) + \alpha \sum_{t\geq 0} (\gamma\lambda)^t \delta_t^1$$



$\lambda = 1$

$0 < \lambda < 1$

$\lambda = 0$

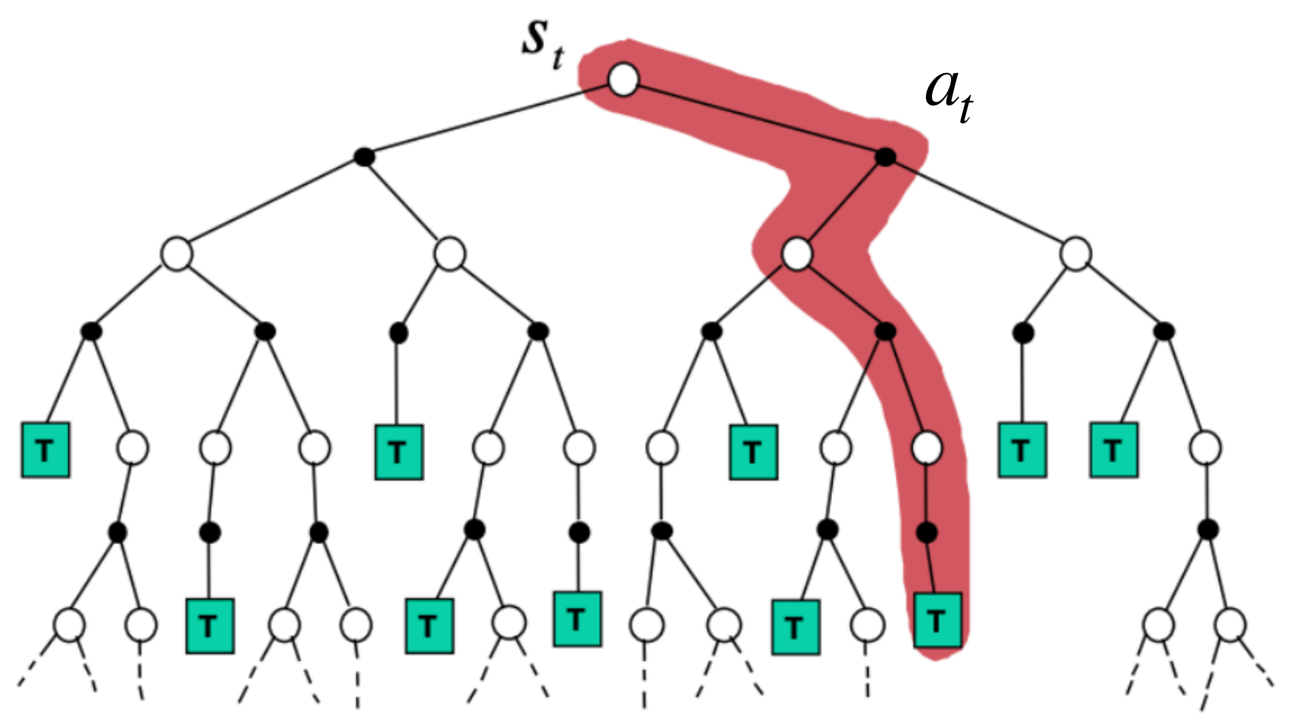
SARSA(λ)

$$\sum_{t\geq 0} (\gamma\lambda)^t \delta_t^1 = (1-\lambda) \sum_{N\geq 1} \lambda^{N-1} \delta_0^N$$

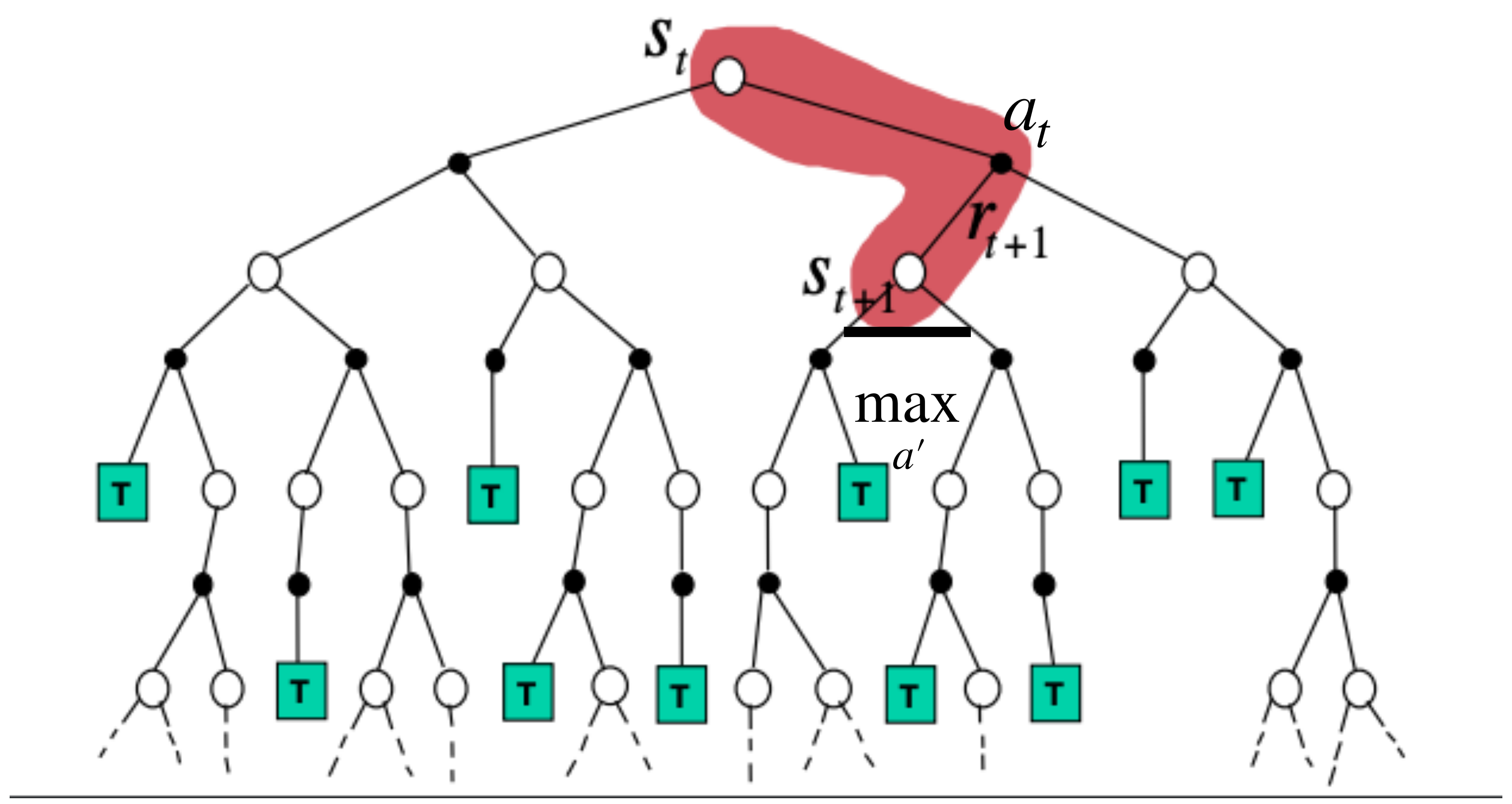
We ensemble all N-step updates.

Policy Evaluation

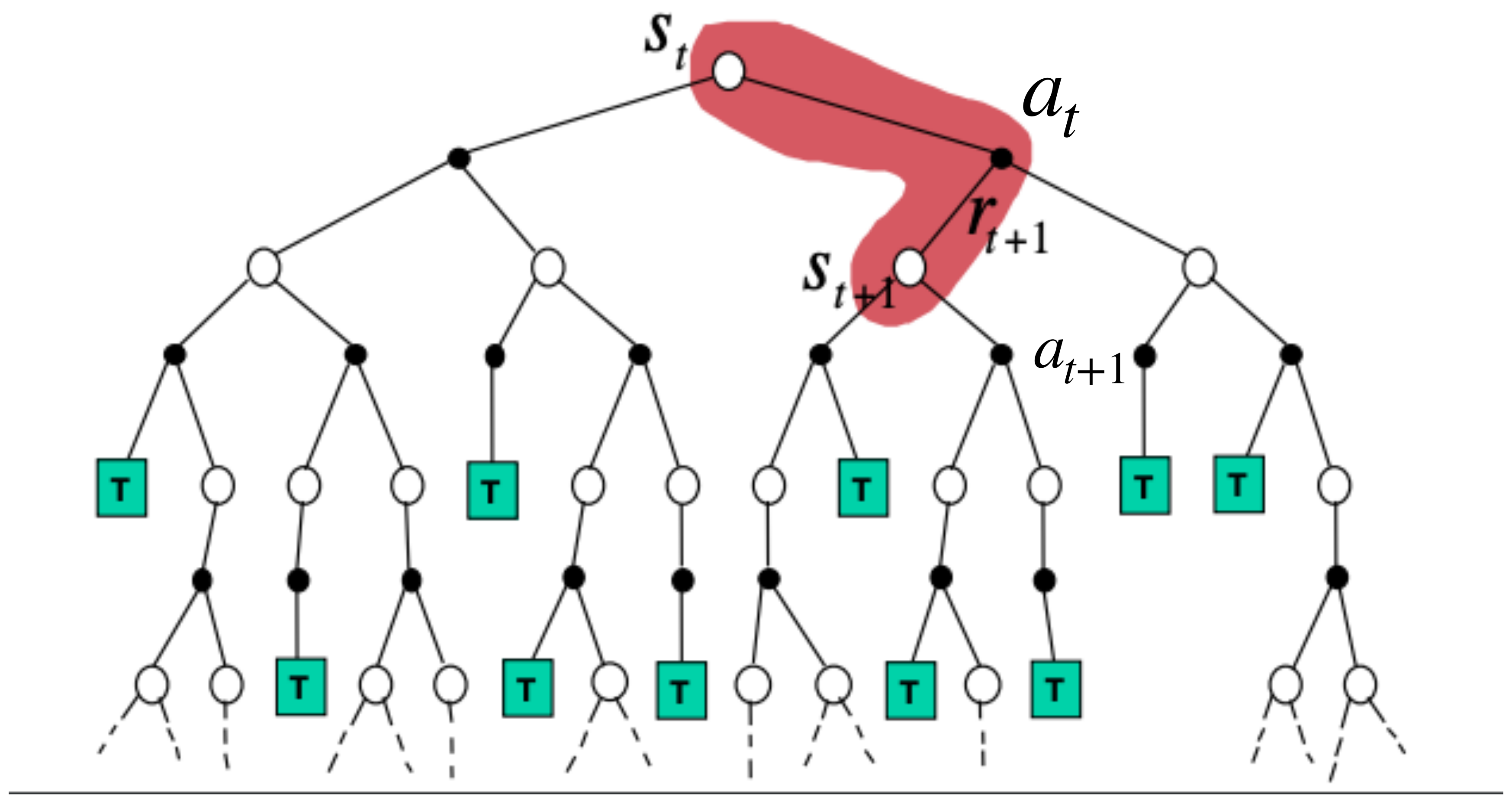
$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(G_k - Q_k(s,a))$$



$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(r + \gamma \max_{a'} Q_k(s',a') - Q_k(s,a))$$



$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha_k(s,a)(r + \gamma Q_k(s',a') - Q_k(s,a))$$



Background

1. Reinforcement Learning Textbook (in Russian): 3.4 - 3.5
2. Sutton & Barto, Chapter 5 + 6 + 7*
3. Practical RL course by YSDA, week 3
4. DeepMind course, lectures 5 + 6

Thank you for your attention!