Model-free Prediction and Control

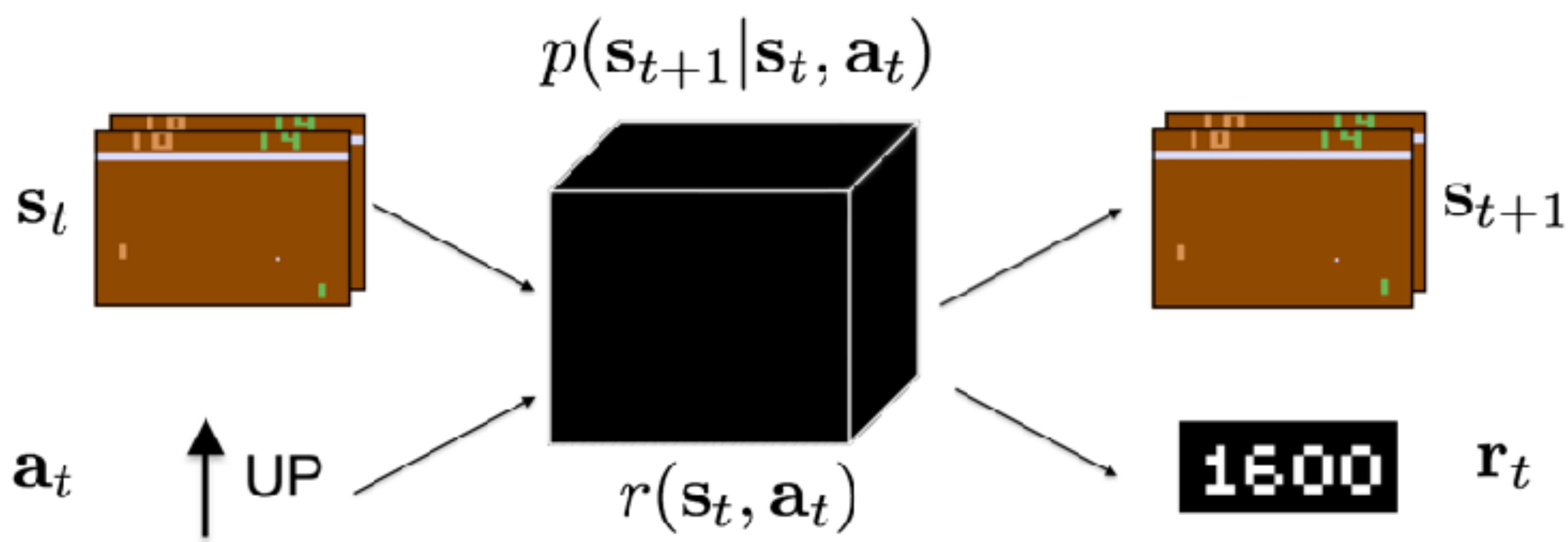
Oleksii Hrinchuk

This lecture is strongly based on David Silver's UCL course on reinforcement learning, lectures 4,5

Model-free prediction and control

- Model-free
  Transition model $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and the reward function $r(\mathbf{s}_t, \mathbf{a}_t)$ are NOT available

- … but we can sample from it!



Monte-Carlo policy evaluation

- Now we are going to learn from trajectories of experience under policy π

$$\mathbf{s}_0, \mathbf{a}_0, \mathbf{r}_0, \mathbf{s}_1, \ldots, \mathbf{a}_{T-1}, \mathbf{r}_{T-1}, \mathbf{s}_T \sim p_{\text{env}}, \pi$$

- Return (reward-to-go) is the total discounted reward

$$Z^\pi(\mathbf{s}_t) = \mathbf{r}_t + \gamma \mathbf{r}_{t+1} + \cdots + \gamma^{T-t-1} \mathbf{r}_{T-1}$$

- Value function is the expected return

$$V^\pi(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \ldots, \mathbf{s}_T} \left[Z^\pi(\mathbf{s}_t) \right]$$

Monte-Carlo policy evaluation

- Evaluate state $\mathbf{s}$ by making rollouts of policy π

- Initialize counter and total return

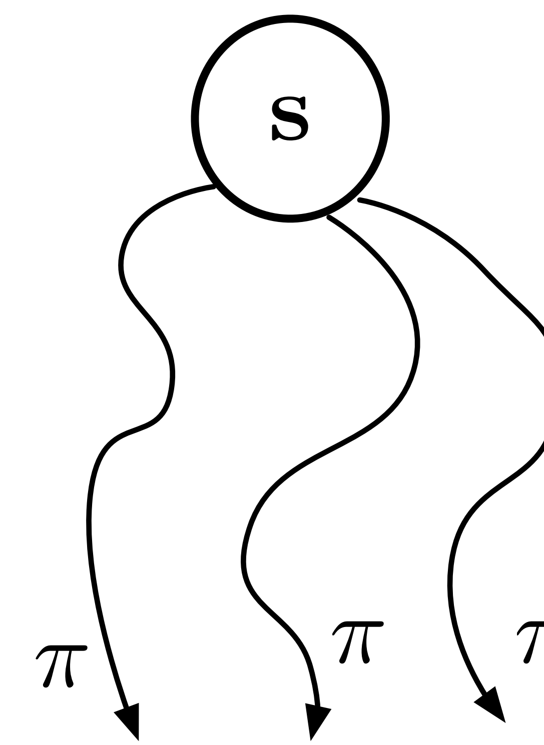
$$N(\mathbf{s}) = 0, \quad \Sigma(\mathbf{s}) = 0$$

- Every time the state $\mathbf{s}$ is visited, increment counter and total return

$$N(\mathbf{s}) \leftarrow N(\mathbf{s}) + 1, \quad \Sigma(\mathbf{s}) \leftarrow \Sigma(\mathbf{s}) + Z^\pi(\mathbf{s})$$

- Estimate value by mean return

$$V(\mathbf{s}) = \Sigma(\mathbf{s})/N(\mathbf{s})$$



Incremental Mean

- The mean of a sequence $x_1, x_2, \ldots$ can be computed incrementally

$$\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j = \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) =$$

$$= \frac{1}{k}(x_k + (k-1)\mu_{k-1}) = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

Incremental Monte-Carlo

- For each state $\mathbf{s}_t$ with return Z_t

$$N(\mathbf{s}_t) \leftarrow N(\mathbf{s}_t) + 1$$

$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \frac{1}{N(\mathbf{s}_t)} \left[Z_t - V(\mathbf{s}_t) \right]$$

- For non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes

$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha \left[Z_t - V(\mathbf{s}_t) \right]$$

Temporal-difference learning

- Monte-Carlo backup

$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha\left[Z_t - V(\mathbf{s}_t)\right]$$

- Temporal-difference backup

$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha\left[\underbrace{r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V(\mathbf{s}_{t+1})}_{\text{TD-target}} - V(\mathbf{s}_t)\right]$$

- TD-error $\delta_t = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V(\mathbf{s}_{t+1}) - V(\mathbf{s}_t)$

- Do not confuse with Bellman error!

$$\mathrm{BE}_t = \mathbb{E}_{\mathbf{a}_t}\left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V(\mathbf{s}_{t+1})\right] - V(\mathbf{s}_t)$$

Bias / variance trade-off

- Return $Z^\pi(\mathbf{s}_t)$ is unbiased estimate of $V^\pi(\mathbf{s}_t)$
- True TD target $\mathbf{r}_t + \gamma V^\pi(\mathbf{s}_{t+1})$ is unbiased estimate of $V^\pi(\mathbf{s}_t)$
- Our TD target $\mathbf{r}_t + \gamma V(\mathbf{s}_{t+1})$ is biased estimate of $V^\pi(\mathbf{s}_t)$ (TD methods bootstrap)
- TD target is much lower variance than the return, as it depends on one random action, transition and reward instead of many

Comparison of MC and TD

- TD methods can learn before knowing the final outcome, while MC methods wait until the end of episode

- TD methods can work in continuing (non-terminating) environments

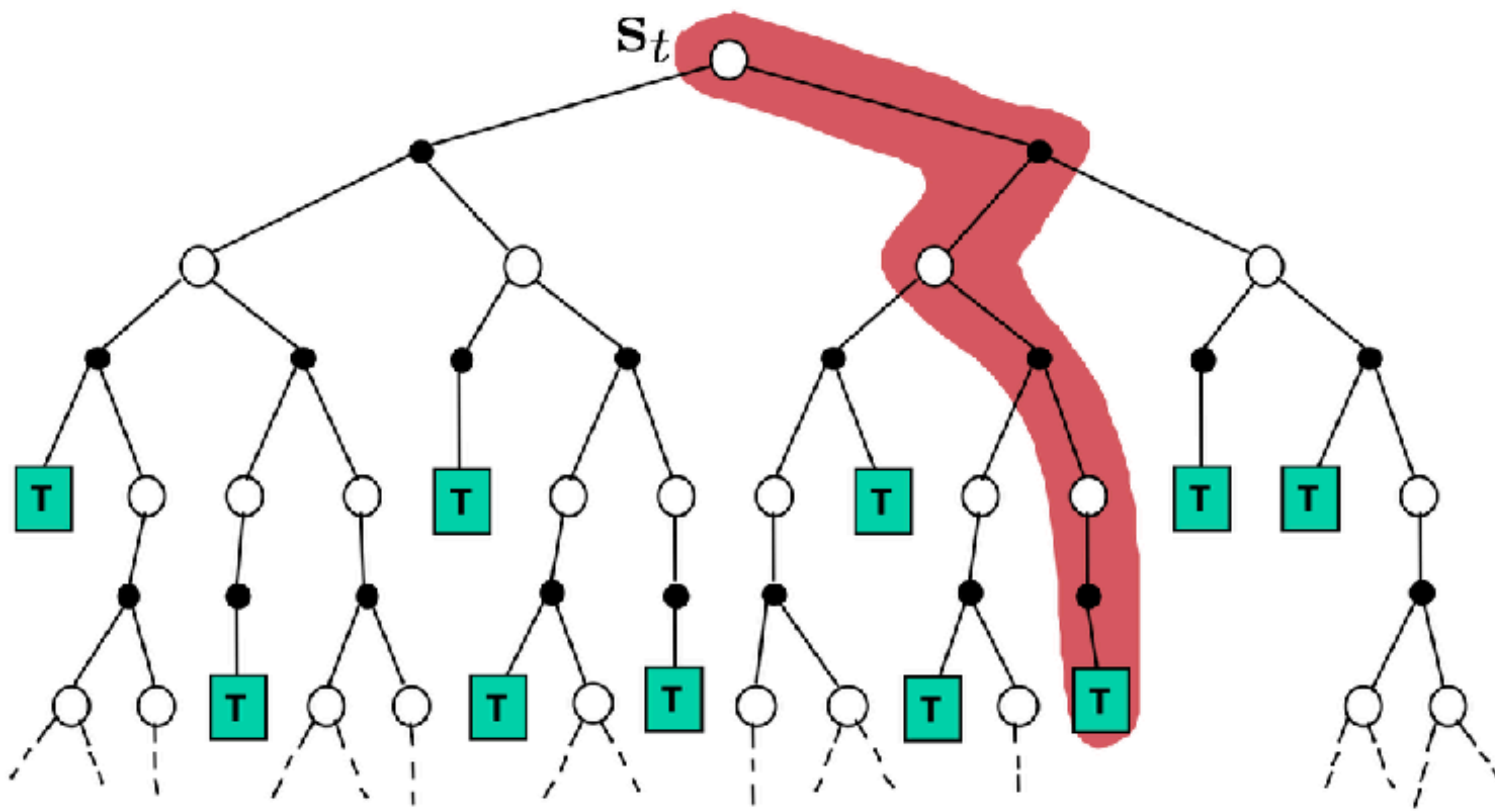
- TD methods exhibit much lower variance, but they introduce some bias to the estimate of $V^{\pi}(\mathbf{s}_t)$

Comparison of MC and TD

- MC has high variance, zero bias
  - Good convergence properties
  - (even with function approximation)
  - Not very sensitive to the initial value
  - Very simple to understand and use

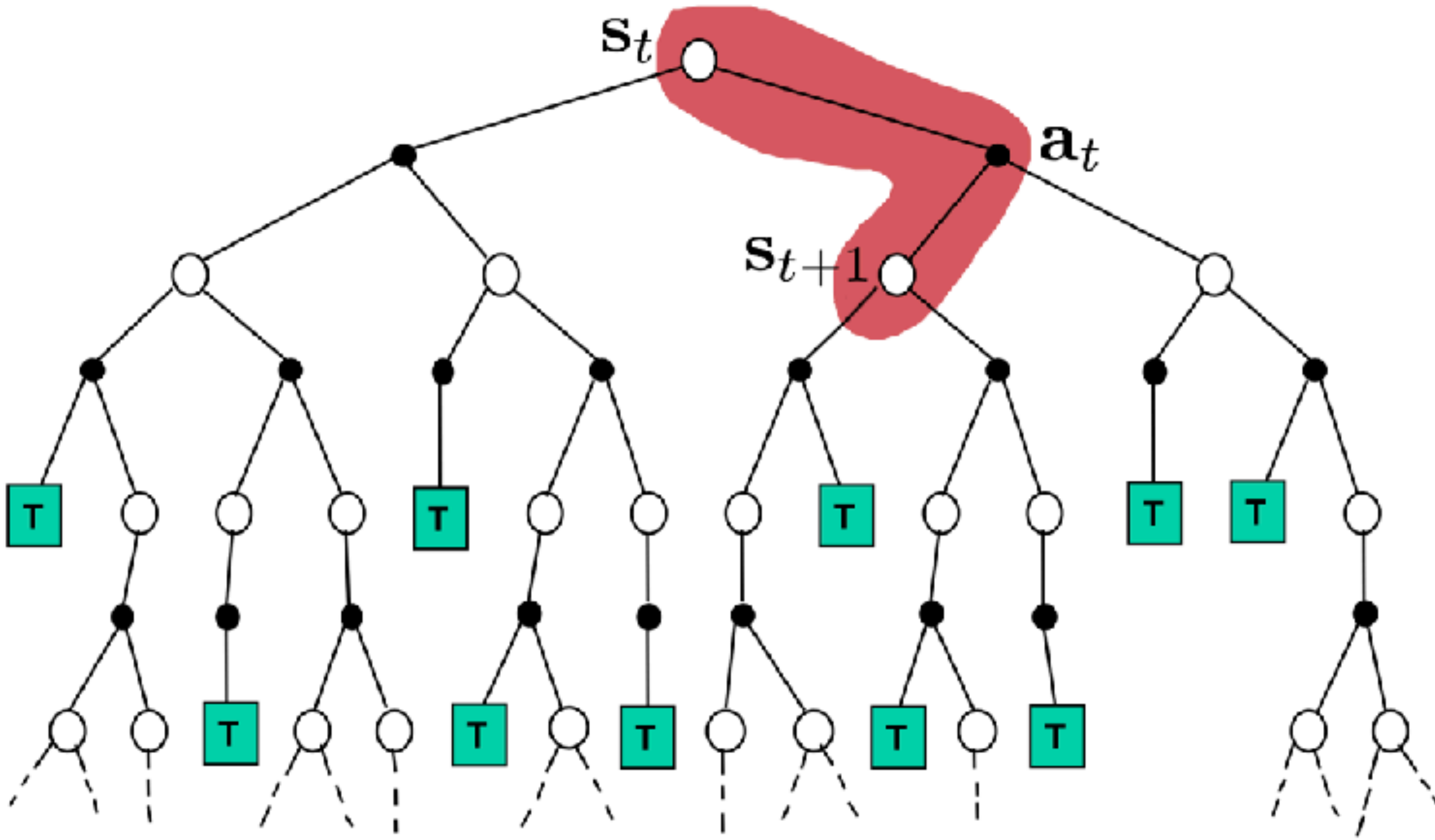
- TD has low variance, some bias
  - Usually more efficient than MC
  - Converge to true value function in tabular case
  - (but not always with function approximation)
  - More sensitive to the initial value

Monte-Carlo backup



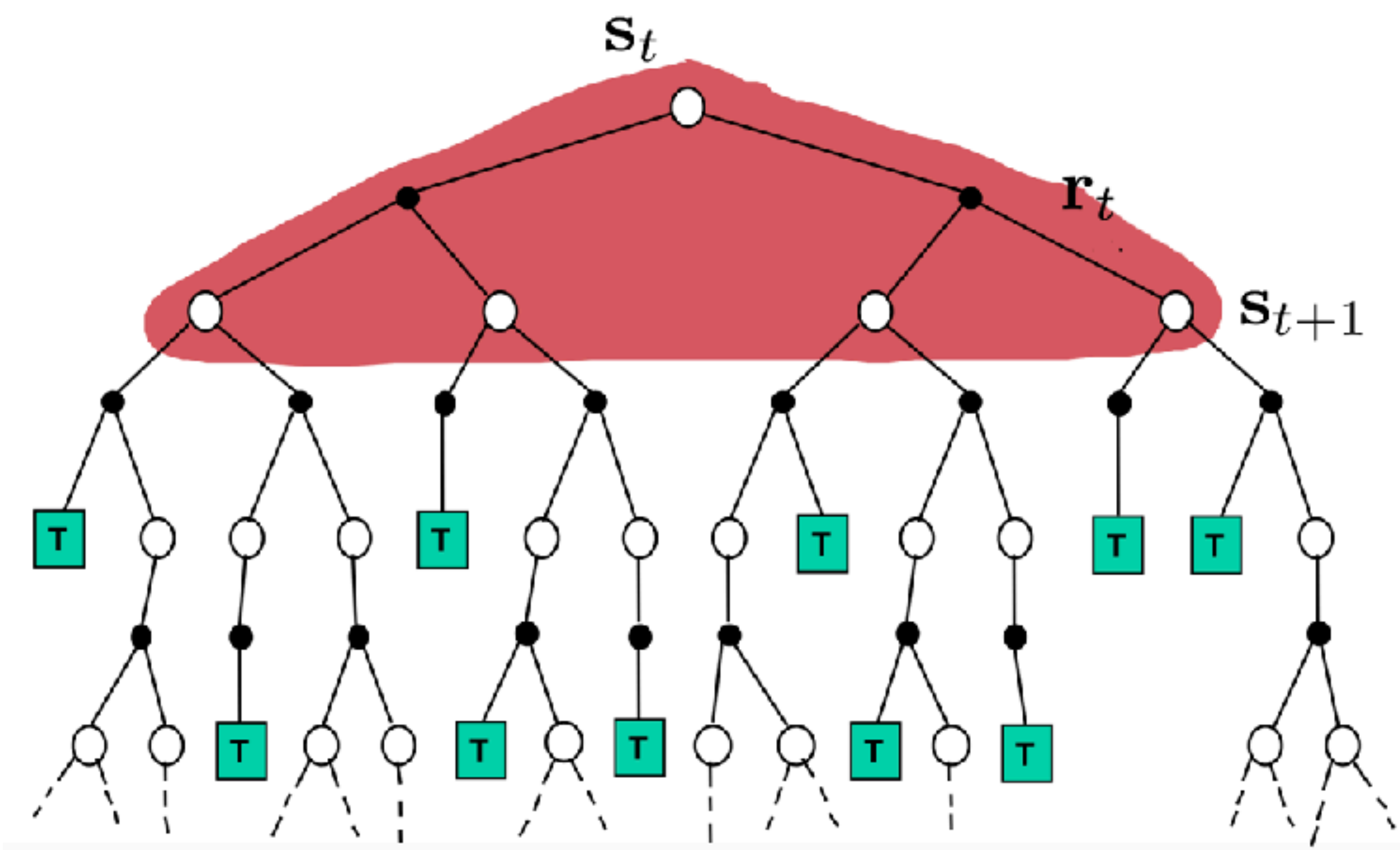
$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha[Z(\mathbf{s}_t) - V(\mathbf{s}_t)]$$

Temporal-difference backup



$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha[\mathbf{r}_t + \gamma V(\mathbf{s}_{t+1}) - V(\mathbf{s}_t)]$$

Dynamic programming backup



$$V(\mathbf{s}_t) \leftarrow \mathbb{E}_{\mathbf{a}_t}[\mathbf{r}_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V(\mathbf{s}_{t+1})]$$

Model-free control

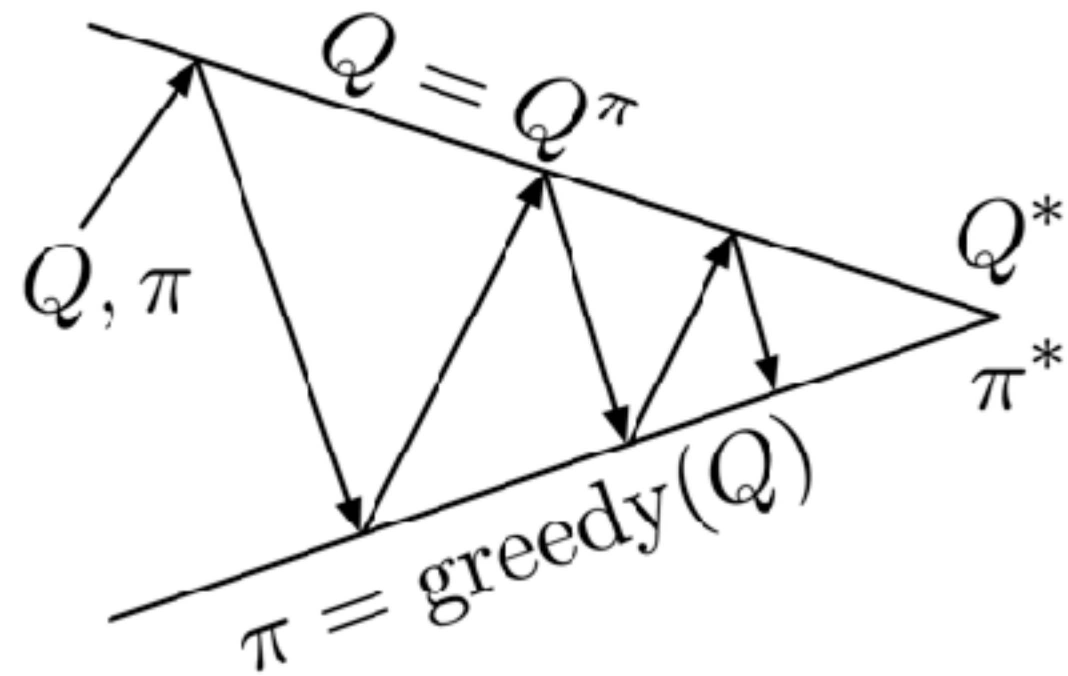
- Monte-Carlo policy evaluation $V = V^{\pi}$
- Greedy policy improvement?

  $$\pi(\mathbf{s}_t) = \arg\max_{\mathbf{a_t} \in \mathcal{A}} \left[\mathbf{r}_t + \gamma \mathbb{E}_{\mathbf{s}_{t+1}} V(\mathbf{s}_{t+1})\right]$$

  requires model of MDP

- Use Q-function!

  $$\pi(\mathbf{s}_t) = \arg\max_{\mathbf{a_t} \in \mathcal{A}} Q(\mathbf{s}_t, \mathbf{a}_t)$$



Epsilon-greedy policy improvement

- Greedy policy can lead to poor exploration

$$\pi(\mathbf{a}|\mathbf{s}) = \begin{cases} 1, & \mathbf{a} = \arg\max_{\mathbf{a}\in\mathcal{A}} Q(\mathbf{s},\mathbf{a}) \\ 0, & \text{otherwise} \end{cases}$$

- Epsilon-greedy policy ensures continual exploration

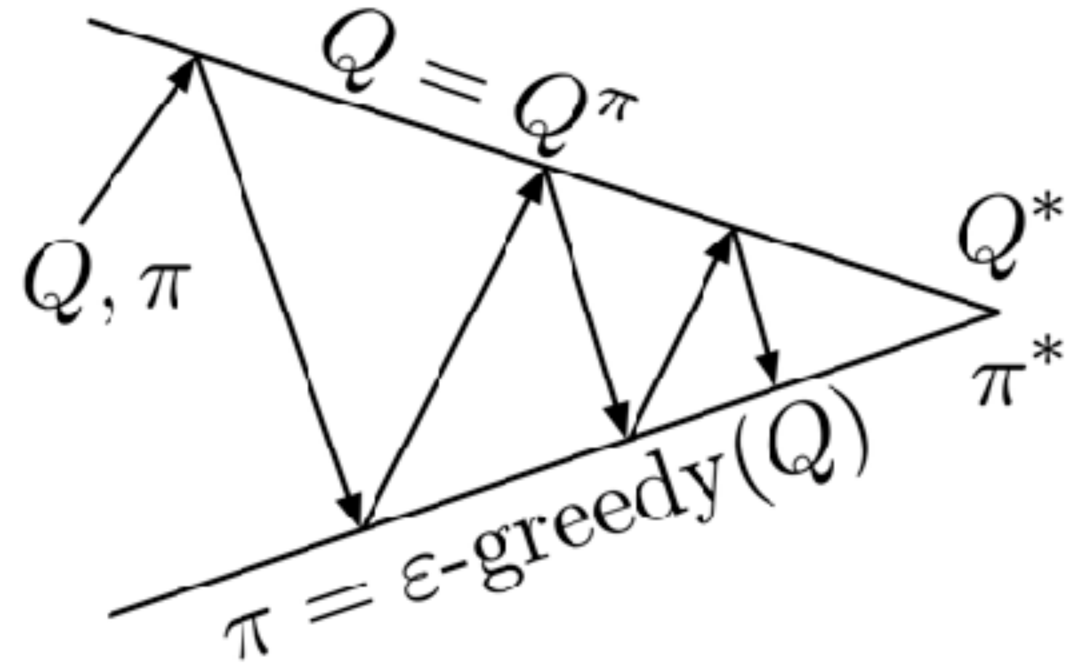
$$\pi(\mathbf{a}|\mathbf{s}) = \begin{cases} 1-\varepsilon, & \mathbf{a} = \arg\max_{\mathbf{a}\in\mathcal{A}} Q(\mathbf{s},\mathbf{a}) \\ \varepsilon/(|\mathcal{A}|-1), & \text{otherwise} \end{cases}$$

- All actions are tried with non-zero probability

Monte-Carlo policy iteration

- Policy evaluation $\pi \rightarrow Q^{\pi}$
  Monte-Carlo policy evaluation algorithm

- Policy improvement $\pi, Q^{\pi} \rightarrow \hat{\pi}$
  Epsilon-greedy policy improvement algorithm



What about TD?

- Temporal-difference learning has several advantages over Monte-Carlo
  — Lower variance
  — Learn on the fly

- Idea: use TD instead of MC in our control loop

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha[Z^\pi(\mathbf{s}_t, \mathbf{a}_t) - Q(\mathbf{s}_t, \mathbf{a}_t)]$$

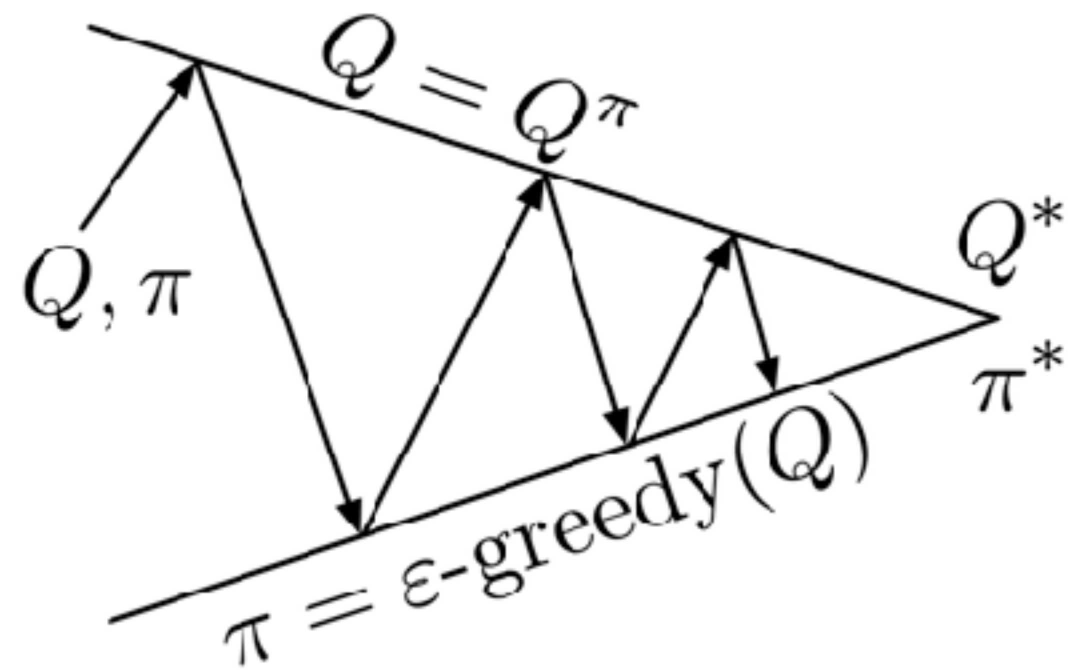
$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha[\mathbf{r}_t + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - Q(\mathbf{s}_t, \mathbf{a}_t)]$$

TD policy evaluation $\Leftrightarrow$ SARSA algorithm

TD policy iteration

- Policy evaluation $\pi \rightarrow Q^{\pi}$
  SARSA policy evaluation algorithm

- Policy improvement $\pi, Q^{\pi} \rightarrow \hat{\pi}$
  Epsilon-greedy policy improvement algorithm



Off-policy learning

- Evaluate target policy $\pi(\mathbf{a}|\mathbf{s})$ while following some behavior policy $\mu(\mathbf{a}|\mathbf{s})$

$$\mathbf{s}_0, \mathbf{a}_0, \mathbf{r}_0, \mathbf{s}_1, \ldots, \mathbf{a}_{T-1}, \mathbf{r}_{T-1}, \mathbf{s}_T \sim p_{\text{env}}, \mu$$

- Why is this important?
  — Learn from observing humans or other agents
  — Re-use experience generated from old policies
  — Learn about optimal policy while following exploratory policy

Importance sampling

- Estimate the expectation of a different distribution

$$\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a})}\left[f(\mathbf{a})\right] = \sum_{\mathbf{a}\in\mathcal{A}} \pi(\mathbf{a})f(\mathbf{a}) = \sum_{\mathbf{a}\in\mathcal{A}} \mu(\mathbf{a})\frac{\pi(\mathbf{a})}{\mu(\mathbf{a})}f(\mathbf{a})$$

$$= \mathbb{E}_{\mathbf{a}\sim\mu(\mathbf{a})}\left[\frac{\pi(\mathbf{a})}{\mu(\mathbf{a})}f(\mathbf{a})\right]$$

IS for off-policy Monte-Carlo

- Weight return $Z^{\mu}(\mathbf{s}_t)$ according to IS estimator

$$Z^{\pi}(\mathbf{s}_t) = \frac{\pi(\mathbf{a}_t|\mathbf{s}_t)}{\mu(\mathbf{a}_t|\mathbf{s}_t)} \frac{\pi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})}{\mu(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})} \dots \frac{\pi(\mathbf{a}_T|\mathbf{s}_T)}{\mu(\mathbf{a}_T|\mathbf{s}_T)} Z^{\mu}(\mathbf{s}_t)$$

- Update value towards corrected returns

$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha \left[Z^{\pi}(\mathbf{s}_t) - V(\mathbf{s}_t) \right]$$

- Can not be used if μ is zero when π is non-zero

- Can dramatically increase variance

IS for off-policy TD

- Weight TD-target $y^{\mu}(\mathbf{s}_t) = \mathbf{r}_t + \gamma V(\mathbf{s}_{t+1})$ according to IS estimator

$$y^{\pi}(\mathbf{s}_t) = \frac{\pi(\mathbf{a}_t|\mathbf{s}_t)}{\mu(\mathbf{a}_t|\mathbf{s}_t)} y^{\mu}(\mathbf{s}_t)$$

- Update value toward corrected TD-target

$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha[y^{\pi}(\mathbf{s}_t) - V(\mathbf{s}_t)]$$

- Much lower variance than Monte-Carlo IS

Q-learning

- We now consider off-policy learning of $Q(\mathbf{s}_t, \mathbf{a}_t)$
- No importance sampling is required
- Next action is chosen using behavior policy $\mathbf{a}_t \sim \mu$ but we consider alternative successor action $\mathbf{a}' \sim \pi$
- And update q-values towards value of alternative action

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha[\mathbf{r}_t + \gamma Q(\mathbf{s}_{t+1}, \mathbf{a}') - Q(\mathbf{s}_t, \mathbf{a}_t)]$$

Off-policy control with Q-learning

- We now allow both behavior and target policies to improve

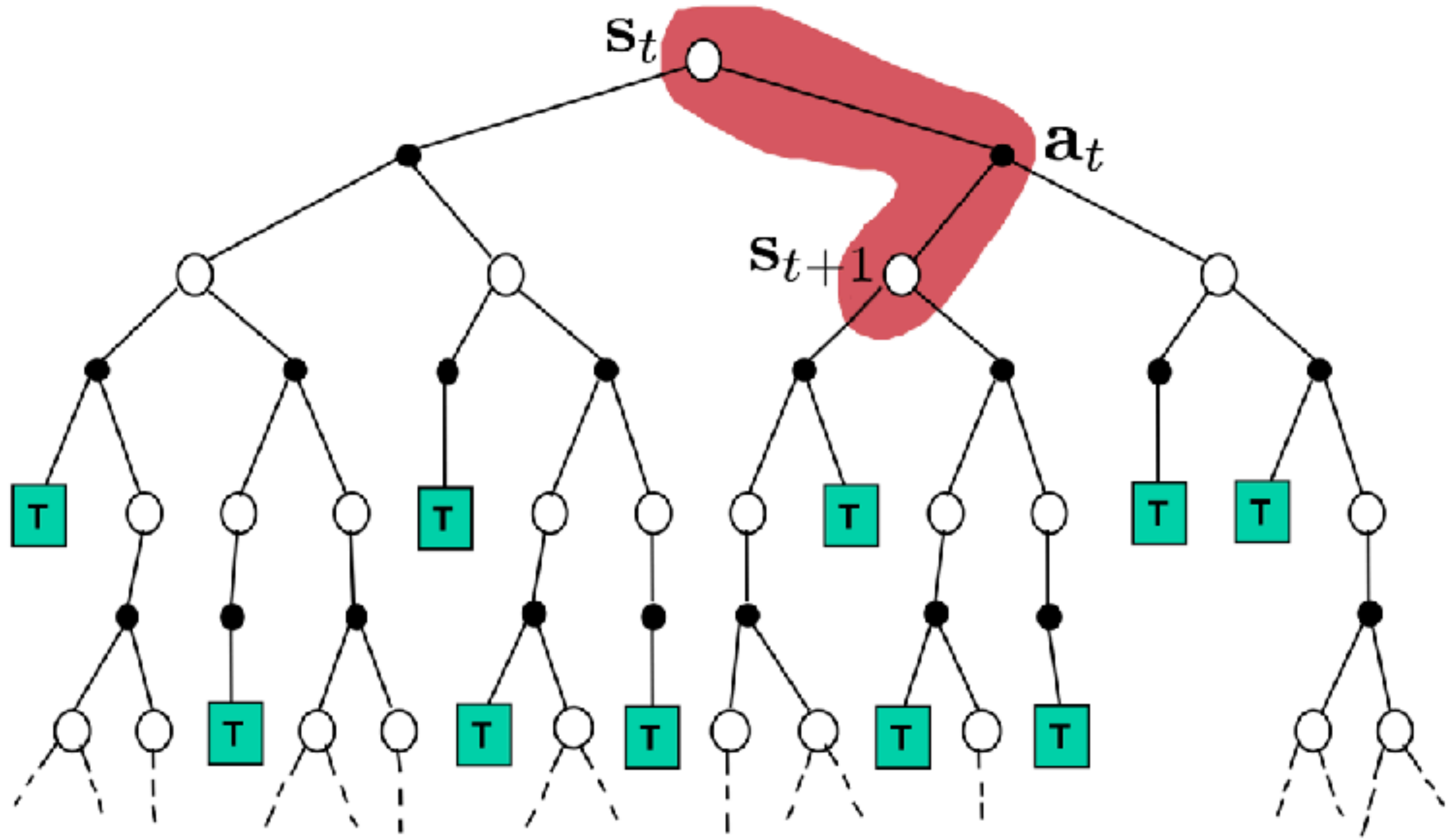
- The target-policy is greedy w.r.t. $Q(\mathbf{s}, \mathbf{a})$

- The behavior policy is epsilon-greedy w.r.t. $Q(\mathbf{s}, \mathbf{a})$

- The Q-learning target then simplifies

$$\begin{aligned}\mathbf{r}_t + \gamma Q(\mathbf{s}_{t+1}, \mathbf{a}') &= \mathbf{r}_t + \gamma Q(\mathbf{s}_{t+1}, \arg\max_{\mathbf{a}' \in \mathcal{A}} Q(\mathbf{s}_{t+1}, \mathbf{a}')) \\ &= \mathbf{r}_t + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q(\mathbf{s}_{t+1}, \mathbf{a}')\end{aligned}$$

Q-learning control algorithm



$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha[\mathbf{r}_t + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q(\mathbf{s}_{t+1}, \mathbf{a}') - Q(\mathbf{s}_t, \mathbf{a}_t)]$$