Introduction to Machine Learning

Reinforcement Learning

Barnabás Póczos

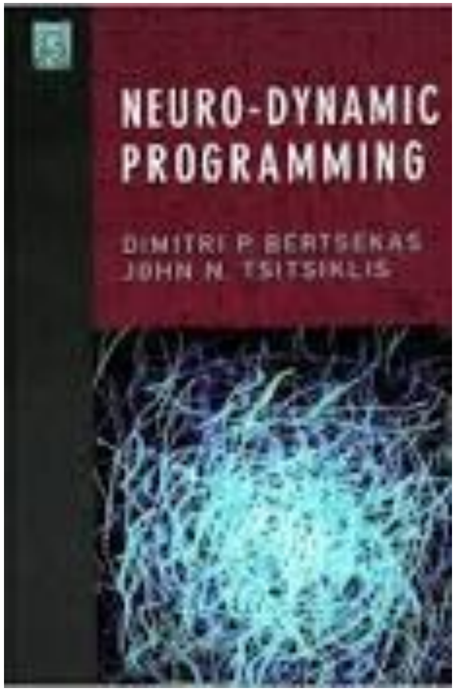
MACHINE LEARNING DEPARTMENT
Carnegie Mellon.
School of Computer Science

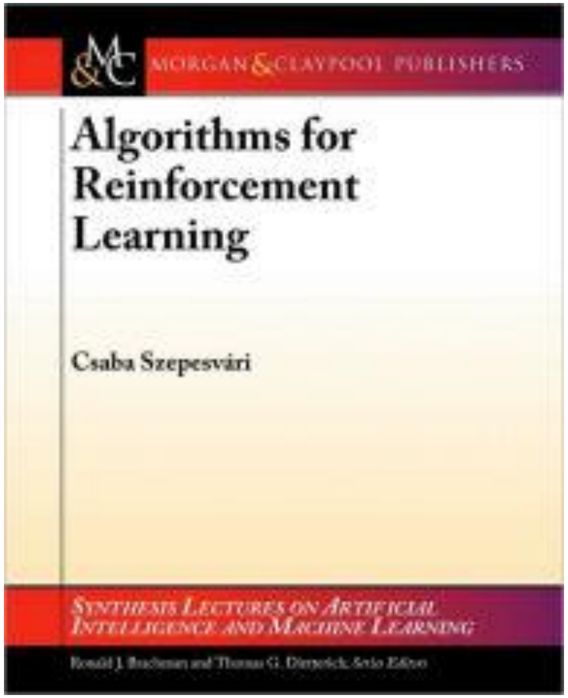
Contents

- **Markov Decision Processes:**
  - State-Value function, Action-Value Function
  - Bellman Equation
  - Policy Evaluation, Policy Improvement, Optimal Policy
- **Dynamical programming:**
  - Policy Iteration
  - Value Iteration
- **Modell Free methods:**
  - MC Tree search
  - TD Learning

RL Books


Reinforcement
Learning
Richard S. Sutton and Andrew G. Barto


NEURO-DYNAMIC
PROGRAMMING
DIMITRI P. BERTSEKAS
JOHN N. TSITSIKLIS

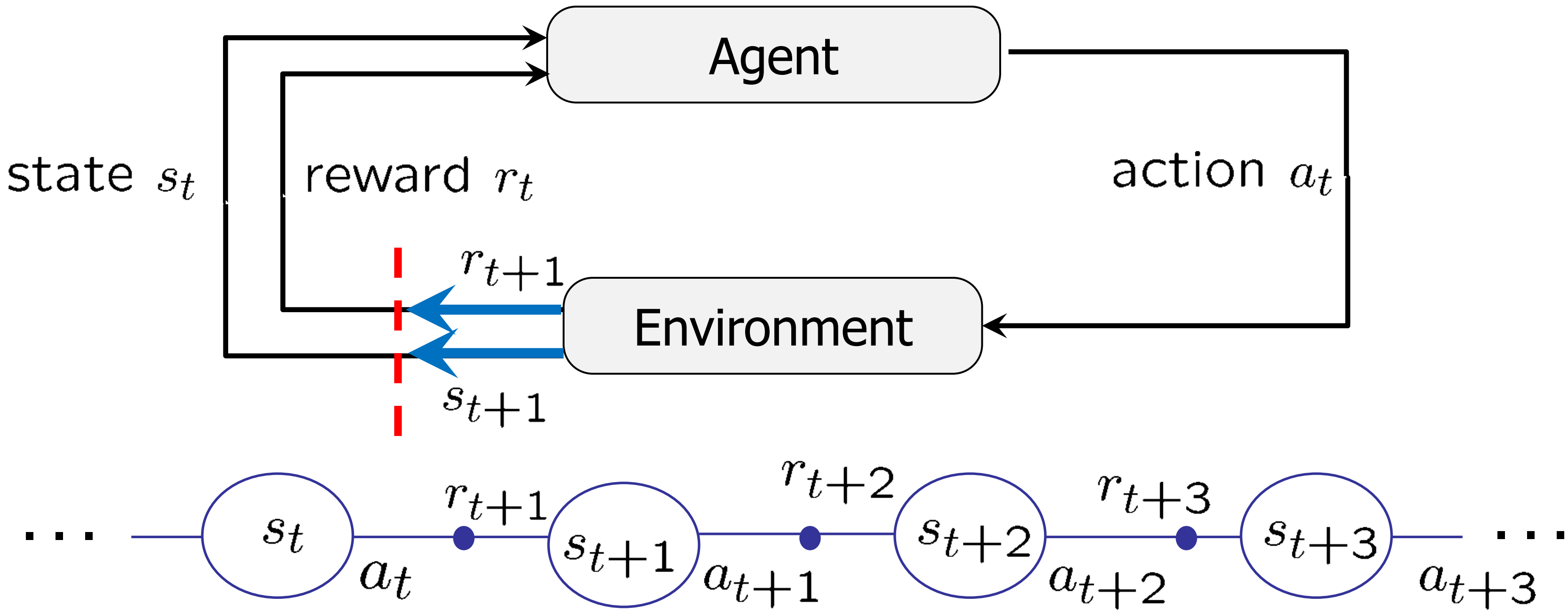

MORGAN & CLAYPOOL PUBLISHERS
Algorithms for
Reinforcement
Learning
Csaba Szepesvári
SYNTHESIS LECTURES ON ARTIFICIAL
INTELLIGENCE AND MACHINE LEARNING

Introduction to Reinforcement Learning

Reinforcement Learning Applications

- Finance
  - Portfolio optimization
  - Trading
- Inventory optimization
- Control
  - Elevator, Air conditioning, power grid, …
- Robotics
- Games
  - Go, Chess, Backgammon
  - Computer games
- Chatbots
- …

Reinforcement Learning Framework



- ⋆ Agent and environment interact in discrete time steps: $t = 0, 1, 2, \ldots$
- ⋆ Agent observes state $s_t \in \mathcal{S}$ in time step t.
- ⋆ Produces action $a_t \in \mathcal{A}(s_t)$ in time step t.
- ⋆ Get resulting reward $r_{t+1} \in \mathbb{R}$
- ⋆ and resulting next state $s_{t+1} \in \mathcal{S}$

Markov Decision Processes

RL Framework + Markov assumption

$$\text{MDP} = (\mathcal{S}, \mathcal{A}, P, R, s_0, \gamma).$$

$\mathcal{S}$: observable state space

$\mathcal{A}$: action space

P: state transition probabilities

R: reward function

s_0: starting state

γ: reward discont rate.

Markov assumption: $P(s_{t+1}|s_0, a_0, \ldots, s_t, a_t) = P(s_{t+1}|s_t, a_t)$

Reward assumption: $R(s_0, a_0, \ldots, s_t, a_t, s_{t+1}) = R(s_t, a_t, s_{t+1}) = r_{t+1} \in \mathbb{R}$

Policy: $\pi(s, a) = P(a_t|s_t) \in [0, 1]$, that is $a_t \sim \pi(s_t, \cdot)$

Goal : $\max_{\pi} \mathbb{E}[r_0 + r_1 + \ldots]$

Discount Rates

Goal: $\max_{\pi} \mathbb{E}[r_0 + r_1 + r_2 + \ldots]$

An issue:

$r_0 + r_1 + r_2 + \ldots$ can be infinte...

Solution:

New goal: $\max_{\pi} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, for some $0 < \gamma < 1$ discount rate

RL is different from Supervised/Unsupervised learning

$\star$ Functions to be learned: $\pi : \mathcal{S} \rightarrow \mathcal{A}$

$\star$ However, training examples are not in the form of (s, a) pairs!

$\star$ Training examples are in the form of $\{(s_t, a_t), r_t\}_{t=1}^{T}$

(or $\{(s_t, a_t, s_{t+1}), r_t\}_{t=1}^{T}$)

State-Value Function

For a given state s and policy π, the value of state s:

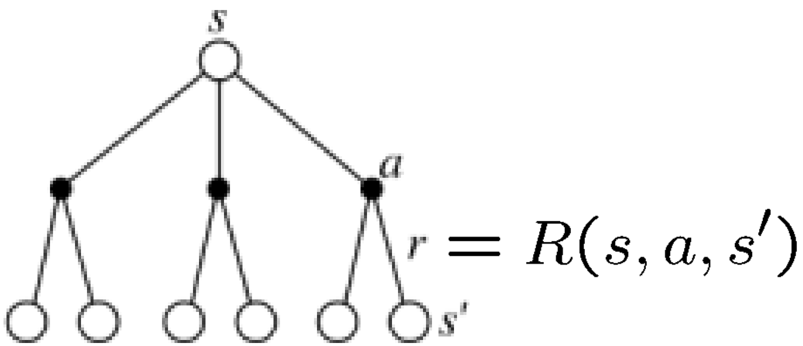
$$V^{\pi}(s) := \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \,\middle|\, s_t = s\right]$$

This is the state-value function of policy π

Bellman Equation of V state-value function:

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V^{\pi}(s')]$$

Backup Diagram:



Bellman Equation

Proof of Bellman Equation:

$$
\begin{aligned}
V^\pi(s) :&= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|\, s_t = s\right] \\
&= \mathbb{E}_\pi\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \,|\, s_t = s\right] \\
&= \sum_{a \in \mathcal{A}} \pi(a_t = a | s_t = s) \sum_{s' \in \mathcal{S}} P(s_{t+1} = s' | s_t = s, a_t = a) \\
&\quad \times \left(R(s, a, s') + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \,|\, s_t = s, a_t = a, s_{t+1} = s'\right) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s, a)[R(s, a, s') + \gamma V^\pi(s')]
\end{aligned}
$$

$$
\Rightarrow V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s, a)[R(s, a) + \gamma V^\pi(s')]
$$

Action-Value Function

Value of state s after taking action a.

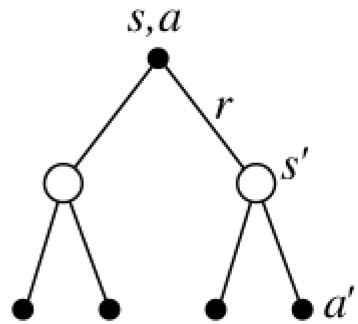
$$Q^\pi(s,a) := \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right]$$

Bellman Equation of the Q Action-Value function:

$$Q^\pi(s,a) = \sum_{s'\in\mathcal{S}} P(s'|s,a)\left[R(s,a,s') + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|s')Q^\pi(s',a')\right]$$

Proof: similar to the proof of the Bellman Equation of V state-value function.

Backup Diagram:



Relation between Q and V Functions

Q from V:

$$Q^{\pi}(s,a) = \sum_{s' \in \mathcal{S}} P(s'|s,a)\left[R(s,a,s') + \gamma V^{\pi}(s')\right]$$

V from Q:

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q^{\pi}(s,a)$$

The Optimal Value Function and Optimal Policy

Partial ordering between policies:

$$\pi_1 \geq \pi_2 \Leftrightarrow V^{\pi_1}(s) \geq V^{\pi_2}(s) \qquad \forall s \in \mathcal{S}$$

Some policies are not comparable!

Optimal policy and optimal state-value function:

$$V^*(s) := \max_\pi V^\pi(s) = V^{\pi^*}(s), \qquad \forall s \in \mathcal{S}$$

π^*: policy whose value function is the maximum out of all policies simultaneously for all states

V*(s) shows the maximum expected discounted reward that one can achieve from state s with optimal play

The Optimal Action-Value Function

Similarly, the **optimal action-value function:**

$$Q^*(s,a) := \max_\pi Q^\pi(s,a)$$

Important Properties:

$$Q^*(s,a) = \mathbb{E}\left[r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\right]$$

$$V^*(s) = \max_{a \in \mathcal{A}} Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

The Existence of the Optimal Policy

Theorem: For any Markov Decision Processes

($\star$) there exists an optimal policy π^* that is at least as good as all other plicies:

$$\pi^* \geq \pi \quad \forall \pi$$

($\star$) There can be many optimal policies, but all optimal policies achieve the optimal value function:

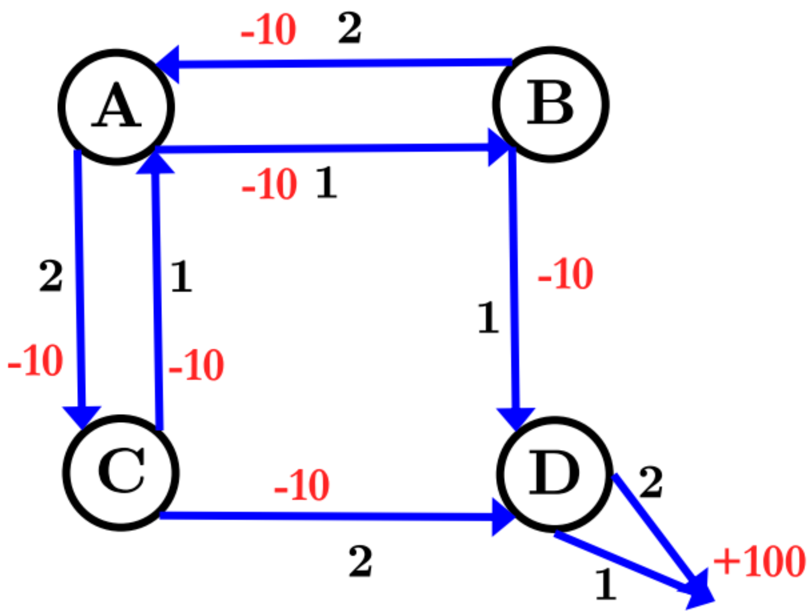
$$V^{\pi^*}(s) = V^*(s) \quad \forall s$$

($\star$) All optimal policies achieve the optimal action-value function,

$$Q^{\pi^*}(s,a) = Q^*(s,a) \quad \forall s,a$$

(*) There is always a deterministic optimal policy for any MDP

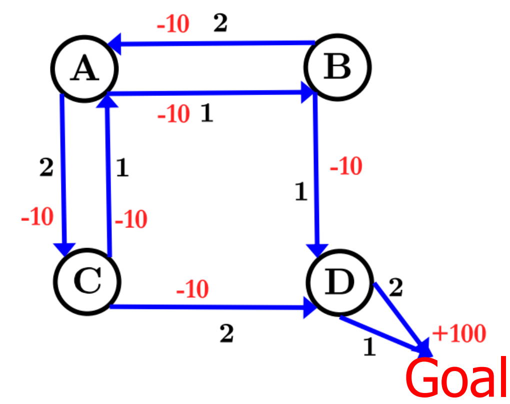
Example



Goal = Terminal state

- 4 states
- 2 possible actions in each state. [E.g in A: 1) go to B or 2) go to C]
- P(s' | s, a) = (0.9 , 0.1) with 10% we go to a wrong direction

Calculating the Value of Policy π



π_1 : always choosing Action 1

$$\pi(1\,|\,A) = 1 \qquad \pi(2\,|\,A) = 0$$
$$\pi(1\,|\,B) = 1 \qquad \pi(2\,|\,B) = 0$$
$$\pi(1\,|\,C) = 1 \qquad \pi(2\,|\,C) = 0$$
$$\pi(1\,|\,D) = 1 \qquad \pi(2\,|\,D) = 0$$

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a) + \gamma V^\pi(s')]$$

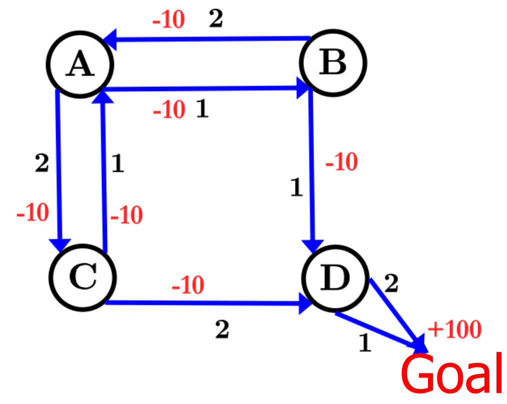
$$V^\pi(D) = 100$$

$$\left.\begin{aligned}
V^\pi(B) = {} & \quad 1 \cdot 0.9 \cdot [-10 + V^\pi(D)] \\
& + 1 \cdot 0.1 \cdot [-10 + V^\pi(A)] \\
V^\pi(C) = {} & \quad 1 \cdot 0.9 \cdot [-10 + V^\pi(A)] \\
& + 1 \cdot 0.1 \cdot [-10 + V^\pi(D)] \\
V^\pi(A) = {} & \quad 1 \cdot 0.9 \cdot [-10 + V^\pi(B)] \\
& + 1 \cdot 0.1 \cdot [-10 + V^\pi(C)]
\end{aligned}\right\} \Rightarrow \left\{\begin{aligned}
V^\pi(A) &= 75.61 \\
V^\pi(B) &= 87.56 \\
V^\pi(C) &= 68.05 \\
V^\pi(D) &= 100
\end{aligned}\right.$$

Calculating the Value of Policy π

π_2 : always choosing Action 2

$$\pi(1 \mid A) = 0 \qquad \pi(2 \mid A) = 1$$
$$\pi(1 \mid B) = 0 \qquad \pi(2 \mid B) = 1$$
$$\pi(1 \mid C) = 0 \qquad \pi(2 \mid C) = 1$$
$$\pi(1 \mid D) = 0 \qquad \pi(2 \mid D) = 1$$

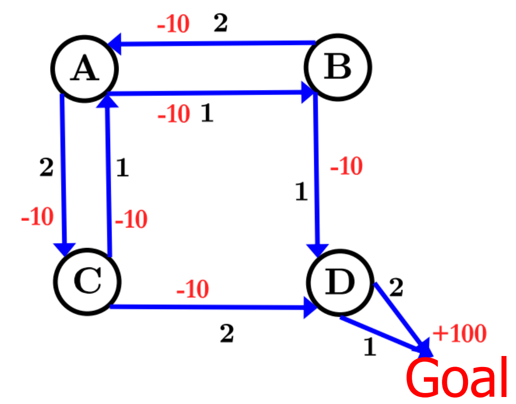


Similarly as before:

$$\begin{aligned} V^{\pi_2}(A) &= 75.61 \\ V^{\pi_2}(B) &= 68.05 \\ V^{\pi_2}(C) &= 87.56 \\ V^{\pi_2}(D) &= 100 \end{aligned}$$

Calculating the Value of Policy π

π_3 : mixed



$\pi(1 \mid A) = 0.4 \qquad \pi(2 \mid A) = 0.6$

$\pi(1 \mid B) = 1 \qquad \pi(2 \mid B) = 0$

$\pi(1 \mid C) = 0 \qquad \pi(2 \mid C) = 1$

$\pi(1 \mid D) = 1 \qquad \pi(2 \mid D) = 0$

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a) + \gamma V^{\pi}(s')]$$

$$\left.\begin{aligned}
V^{\pi}(D) &= 100 \\
V^{\pi}(B) &= 1 \cdot 0.9 \cdot [-10 + V^{\pi}(D)] \\
&+ 1 \cdot 0.1 \cdot [-10 + V^{\pi}(A)] \\
V^{\pi}(C) &= 1 \cdot 0.9 \cdot [-10 + V^{\pi}(D)] \\
&+ 1 \cdot 0.1 \cdot [-10 + V^{\pi}(A)] \\
V^{\pi}(A) &= 0.4 \cdot 0.9 \cdot [-10 + V^{\pi}(B)] \\
&+ 0.4 \cdot 0.1 \cdot [-10 + V^{\pi}(C)] \\
&+ 0.6 \cdot 0.9 \cdot [-10 + V^{\pi}(C)] \\
&+ 0.6 \cdot 0.1 \cdot [-10 + V^{\pi}(B)]
\end{aligned}\right\} \Rightarrow \left\{\begin{aligned}
V^{\pi_3}(A) &= 77.78 \\
V^{\pi_3}(B) &= 87.78 \\
V^{\pi_3}(C) &= 87.78 \\
V^{\pi_3}(D) &= 100
\end{aligned}\right.$$

Comparing the 3 policies

	π_1	π_2	π_3
A	75.61	75.61	77.78
B	87.56	68.05	87.78
C	68.05	87.56	87.78
D	100	100	100

$\pi_1 \leq \pi_3$ and $\pi_2 \leq \pi_3 \Rightarrow \pi_3$ is optimal among these 3 policies

π_1 and π_2 are not comparable

Bellman optimality equation for V*

Similarly, as we derived Bellman Equation for V and Q,
we can derive Bellman Equations for V* and Q* as well

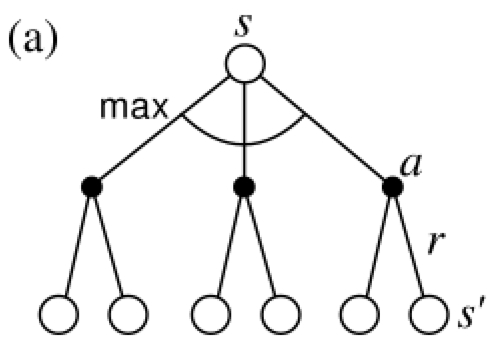
We proved this for V:

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V^{\pi}(s')]$$

Theorem: Bellman optimality equation for V*:

$$V^{*}(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V^{*}(s')]$$

Backup Diagram:



Bellman optimality equation for V*

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V^\pi(s')]$$

Proof of Bellman optimality equation for V*:

$$\begin{aligned}
V^*(s) &:= \max_{a \in \mathcal{A}(s)} Q^*(s,a) \\
&= \max_a \mathbb{E}_{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right] \\
&= \max_a \mathbb{E}_{\pi^*}\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \,\middle|\, s_t = s, a_t = a\right] \\
&= \max_a \mathbb{E}_{\pi^*}\left[r_{t+1} + \gamma V^*(s_{t+1}) \,\middle|\, s_t = s, a_t = a\right] \\
&= \max_a \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V^*(s')]
\end{aligned}$$

Bellman optimality equation for Q*

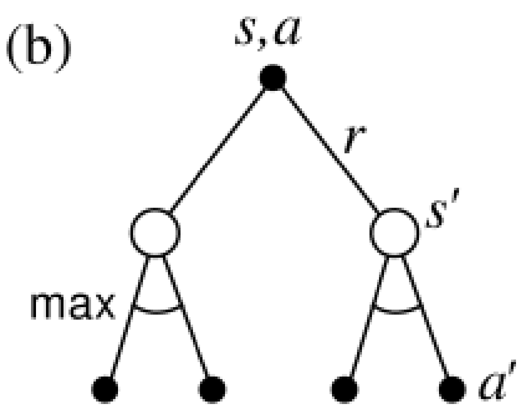
$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

Bellman optimality equation for Q*:

$$\begin{aligned} Q^*(s,a) &= \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1},a') \mid s_t = s, a_t = a\right] \\ &= \sum_{s' \in \mathcal{S}} P(s'|s,a)\left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right] \end{aligned}$$

Proof: similar to the proof of the Bellman Equation of V*.

Backup Diagram:



Greedy Policy for V

Definition: Greedy policy for a given $Q(s,a)$ function:

$$\pi(s,a) = \begin{cases} 1, & \text{if } a = \arg\max_a Q(s,a) \\ 0, & \text{otherwise;} \end{cases}$$

Equivalently, (Greedy policy for a given V(s) function):

$$\pi(s,a) = \begin{cases} 1, & \text{if } a = \arg\max_a P(s' \mid s,a)(R(s,a,s') + \gamma V(s')) \\ 0, & \text{otherwise;} \end{cases}$$

The Optimal Value Function and Optimal Policy

$$V^*(s) = \max_{\pi} V^{\pi}(s) = V^{\pi^*}(s), \qquad \forall s \in S$$

Bellman optimality equation for V:*

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s,a') \left[R(s,a,s') + \gamma V^*(s') \right]$$

This is a nonlinear equation!

Theorem: A greedy policy for V* is an optimal policy. Let us denote it with π*

Theorem: A greedy optimal policy from the optimal Value function:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \left[R(s,a,s') + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \right]$$

RL Tasks

- **Policy evaluation:**

  Given policy π, what is $V^{\pi}(s)$ and $Q^{\pi}(s,a)$?

- **Policy improvement**

  Given policy π, can we create another policy π' such that $\pi' \geq \pi$, that is $V^{\pi'}(s) \geq V^{\pi}(s)\ \forall s$?

- **Finding an optimal policy**

  How can we find an optimal policy π^*?

Policy Evaluation

Policy Evaluation with Bellman Operator

Bellman equation:

$$V^\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \sum_{s'\in\mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V^\pi(s')]$$

This equation can be used as a fix point equation to evaluate policy π

Bellman operator: (one step with π, then using V)

$$(T_\pi V)(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \sum_{s'\in\mathcal{S}} P(s' \mid s,a) \left[R(s,a,s') + \gamma V(s')\right]$$

Iteration:

$$\begin{aligned} V_0 &:= \text{arbitrary} \\ V_{k+1} &:= (T_\pi V_k) \end{aligned}$$

$$V_{k+1}(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \sum_{s'\in\mathcal{S}} P(s' \mid s,a) \left[R(s,a,s') + \gamma V_k(s')\right]$$

Theorem: then $V_k \to V^\pi$

Policy Improvement

Policy Improvement

Given π and V^π.

Theorem:

If there is a (deterministic) policy π' such that for all $s \in S$

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s)$$

then $V^{\pi'} \geq V^\pi$

That is, if we improve one step everywhere (then π), then we improve the whole policy.

Proof of Policy Improvement

Proof:

$$
\begin{aligned}
V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s)) \\
&= \mathbb{E}_{\pi'}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s] \\
&\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, \pi'(s_{t+1})) \mid s_t = s] \\
&= \mathbb{E}_{\pi'}\left[r_{t+1} + \gamma \mathbb{E}_{\pi'}[r_{t+2} + \gamma V^{\pi}(s_{t+2})] \mid s_t = s\right] \\
&= \mathbb{E}_{\pi'}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 V^{\pi}(s_{t+2}) \mid s_t = s\right] \\
&= \vdots \\
&= \mathbb{E}_{\pi'}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots \mid s_t = s\right] \\
&= V^{\pi'}(s)
\end{aligned}
$$

If there is at least one $<$ above, then $V^{\pi} \neq V^{\pi'}$

If we can't improve the policy this way, then the policy is optimal.

Finding the Optimal Policy

Finding the Optimal Policy

Model based approaches:

First we will discuss methods that need to know the model:

$P(s' \mid s, a)$ and $R(s, a, s')$.

- **Policy Iteration**
- **Value Iteration**

Model-free approaches:

Then we will discuss "model-free" methods that do NOT need to know the model: $P(s' \mid s, a)$ and $R(s, a, s')$.

- **Monte Carlo Method**
- **TD Learning**

Policy Iteration

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}$.

$\pi(s)$ is a deterministic policy.

$\delta > 0$ is a small threshold parameter.

2. Policy Evaluation

repeat

$\Delta \leftarrow 0$

for all $s \in \mathcal{S}$ **do**:

$v \leftarrow V(s)$

$a \leftarrow \pi(s)$

$V(s) \leftarrow \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

end for

until $\Delta < \delta$

Policy Iteration

3. Policy Improvement

$policyStable \leftarrow true$

for all $s \in \mathcal{S}$ **do**:

$b \leftarrow \pi(s)$

$$\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$$

if $b \neq \pi(s)$ **then**

$policyStable \leftarrow false$

end if

end for

if $policyStable$ **then**

$STOP$

else

Go to 2 (Policy Evaluation)

end if

One drawback of policy iteration is that each iteration involves policy evaluation

Value Iteration

Main idea:

Use the Bellman equation of V^* instead of V^π

The greedy operator:

$$[T^*V](s) := \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$$

V^* is the solution of $V = T^*(V)$ fixpoint iteration.

The value iteration update:

$k = 0$ and $V_0(s) \in \mathbb{R}$ for all $s \in \mathcal{S}$

repeat

  for all $s \in \mathcal{S}$ **do**:

    $$V_{k+1}(s) \leftarrow \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')]$$

  end for

  $k \leftarrow k + 1$

until $V_k(\cdot)$ converged

Model Free Methods

The previous approaches need to know the model: $P(s, a, s')$ and $R(s, a, s')$. In practice, we often don't know them.

Monte Carlo Policy Evaluation

Monte Carlo Policy Evaluation Without knowing the model

⋆ Let $R(s)$ be the reward that can be achieved from state s following policy π.

⋆ It is a random variable with expected value $V^{\pi}(s)$.

$$\begin{aligned} V^{\pi}(s) &:= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|\, s_t = s\right] \\ &= \mathbb{E}_{\pi}[R(s)] \end{aligned}$$

Monte Carlo Estimation of $V^{\pi}(s)$

- **Empirical average**: Let us use N simulations starting from state s following policy π. The observed rewards are:

$$R_1(s), R_2(s), \ldots, R_N(s)$$

- Let $\hat{V}(s) := \frac{1}{N} \sum_{k=1}^{N} R_k(s)$ $\qquad \hat{V}(s) \rightarrow V^{\pi}(s)$
- This is the so-called „Monte Carlo" method.
- MC can estimate $V^{\pi}(s)$ without knowing the model

Online Averages (=Running averages)

- **If we don't want to store the N sample points:**

$$\begin{aligned}\hat{X}_N &= \frac{\sum_{k=1}^{N} x_k}{N} = \frac{\sum_{k=1}^{N-1} x_k + x_N}{N} \\ &= \frac{N-1}{N} \cdot \frac{\sum_{k=1}^{N-1} x_k}{N-1} + \frac{1}{N} x_N \\ &= \left(1 - \frac{1}{N}\right) \hat{X}_{N-1} + \frac{1}{N} x_N \\ &= \hat{X}_{N-1} + \frac{1}{N}(x_N - \hat{X}_{N-1})\end{aligned}$$

- $\alpha_k := 1/k$

$$\hat{X}_N = \hat{X}_{N-1} + \alpha_N \cdot (x_N - \hat{X}_{N-1})$$

Similarly,

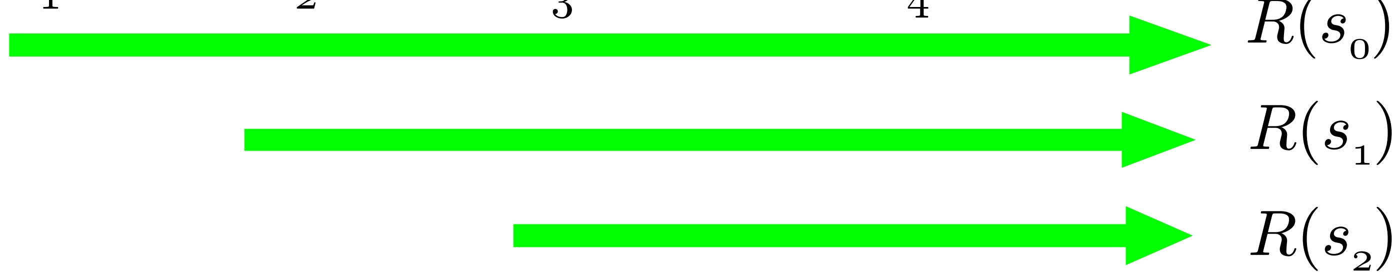
$$V_k(s_t) := V_{k-1}(s_t) + \alpha_k \cdot (R_k(s_t) - V_{k-1}(s_t))$$

A better MC method

- From one single trajectory we can get lots of R estimates:

$$s_0 \quad \rightarrow \quad s_1 \quad \rightarrow \quad s_2 \quad \rightarrow \quad \ldots \quad \rightarrow \quad s_T$$

$$r_1 \qquad\qquad r_2 \qquad\qquad r_3 \qquad\qquad r_4$$



- Warning: These $R(s_i)$ random variables might be dependent!

Temporal Differences method

We already know the MC estimation of V^π:

$$V_k(s_t) := V_{k-1}(s_t) + \alpha_k \cdot (R_k(s_t) - V_{k-1}(s_t))$$

Here is an other estimate:

$$\begin{aligned} V^\pi(s_t) &\approx R_k(s_t) \\ &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \\ &= r_{t+1} + \gamma \left(r_{t+2} + \gamma r_{t+3} + \dots \right) \\ &\approx r_{t+1} + \gamma V_{k-1}(s_{t+1}) \end{aligned}$$

So let us use $r_{t+1} + \gamma V_{k-1}(s_{t+1})$, instead of $R_k(s_t)$

Temporal Differences method

MC estimate: $V_k(s_t) := V_{k-1}(s_t) + \alpha_k \cdot (R_k(s_t) - V_{k-1}(s_t))$

TD estimate: $V_k(s_t) := V_{k-1}(s_t) + \alpha_k \cdot \Big(\boxed{(r_{t+1} + \gamma V_{k-1}(s_{t+1}))} - V_{k-1}(s_t)\Big)$

Instead of waiting for R_k, we estimate it using V_{k-1}

$$V_k(s_t) := (1 - \alpha_k) \cdot V_{k-1}(s_t) + \alpha_k \cdot \Big(r_t + \gamma V_{k-1}(s_{t+1})\Big)$$

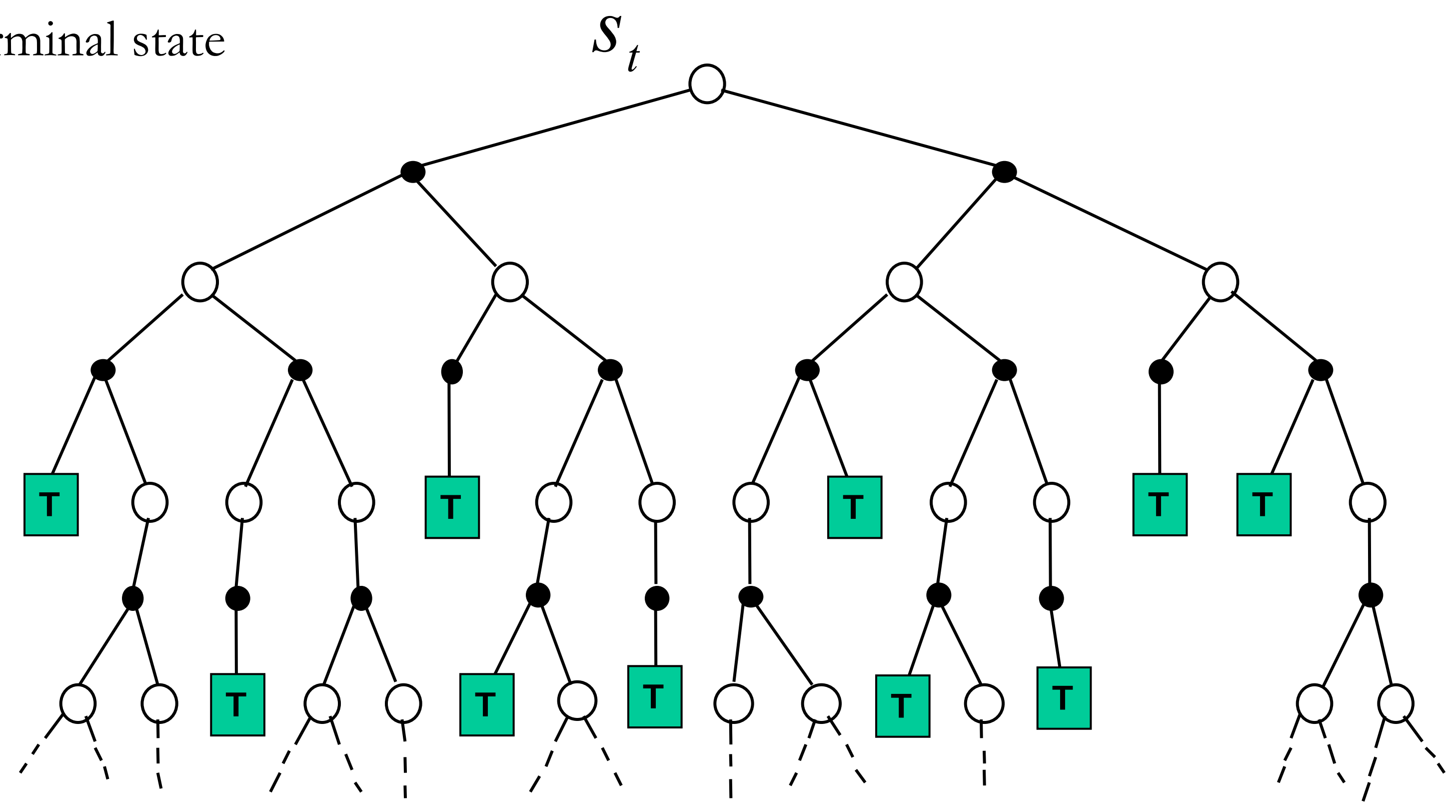
- Temporal difference: $\Big(r_{t+1} + \gamma V_{k-1}(s_{t+1})\Big) - V_{k-1}(s_t)$
- Benefits
  - No need for model! (Dynamic Programming with Bellman operators need them!)
  - No need to wait for the end of the episode! (MC methods need them)
- We use an estimator for creating another estimator (=bootstrapping) … still it works

Comparisons: DP, MC, TD

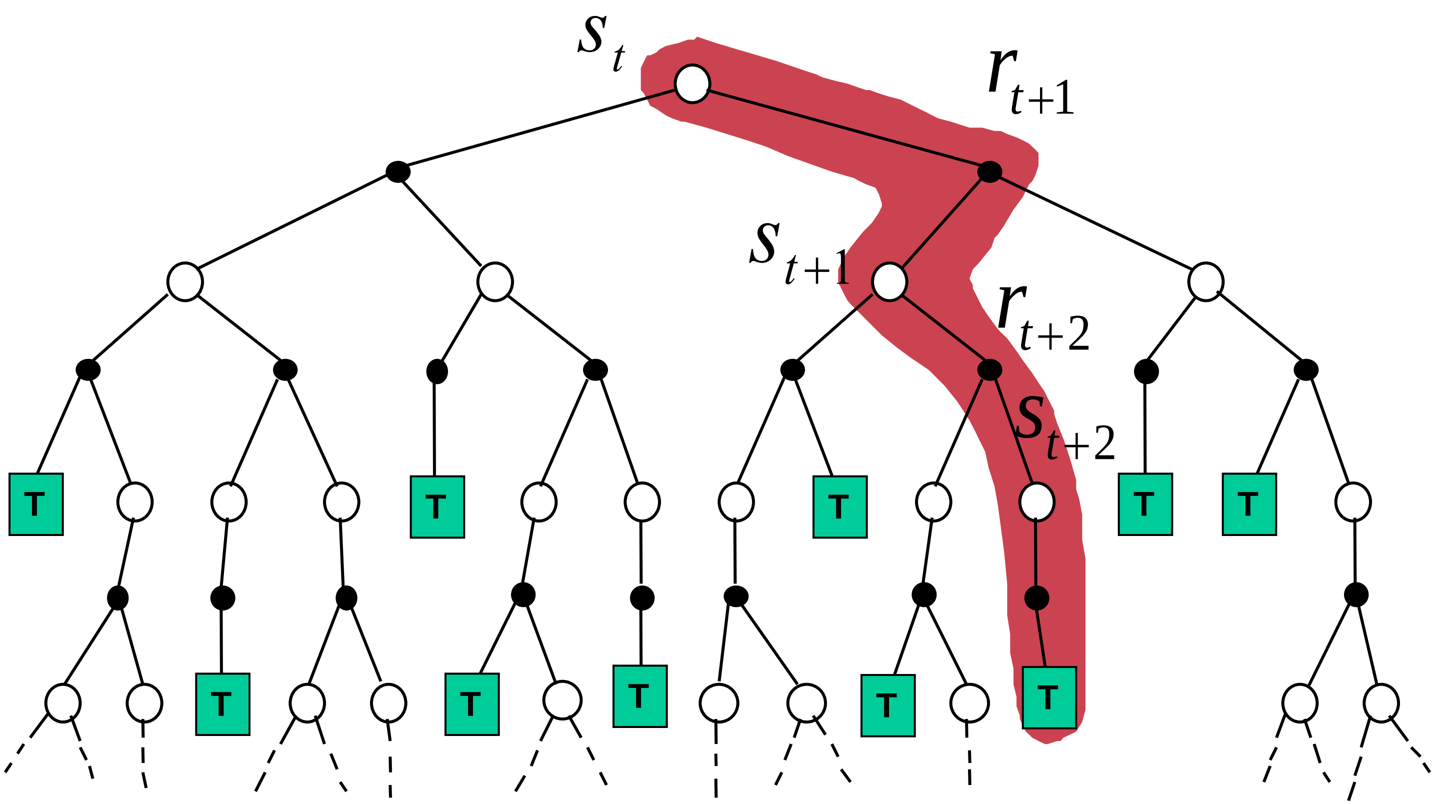
- They all estimate V^{π}
- DP: $V_k(s_t) \approx E_\pi \left(r_{t+1} + \gamma V_{k-1}(s_{t+1}) \mid s_t\right)$
  - Estimate comes from the Bellman equation
  - It needs to know the model
- TD: $V_k(s_t) \approx \left(r_{t+1} + \gamma V_{k-1}(s_{t+1})\right)$
  - Expectation is approximated with random samples
  - Doesn't need to wait for the end of the episodes.
- MC: $V_k(s_t) \approx R_t(s_t)$
  - Expectation is approximated with random samples
  - It needs to wait for the end of the episodes

MDP Backup Diagrams

- White circle: state
- Black circle: action
- T: terminal state

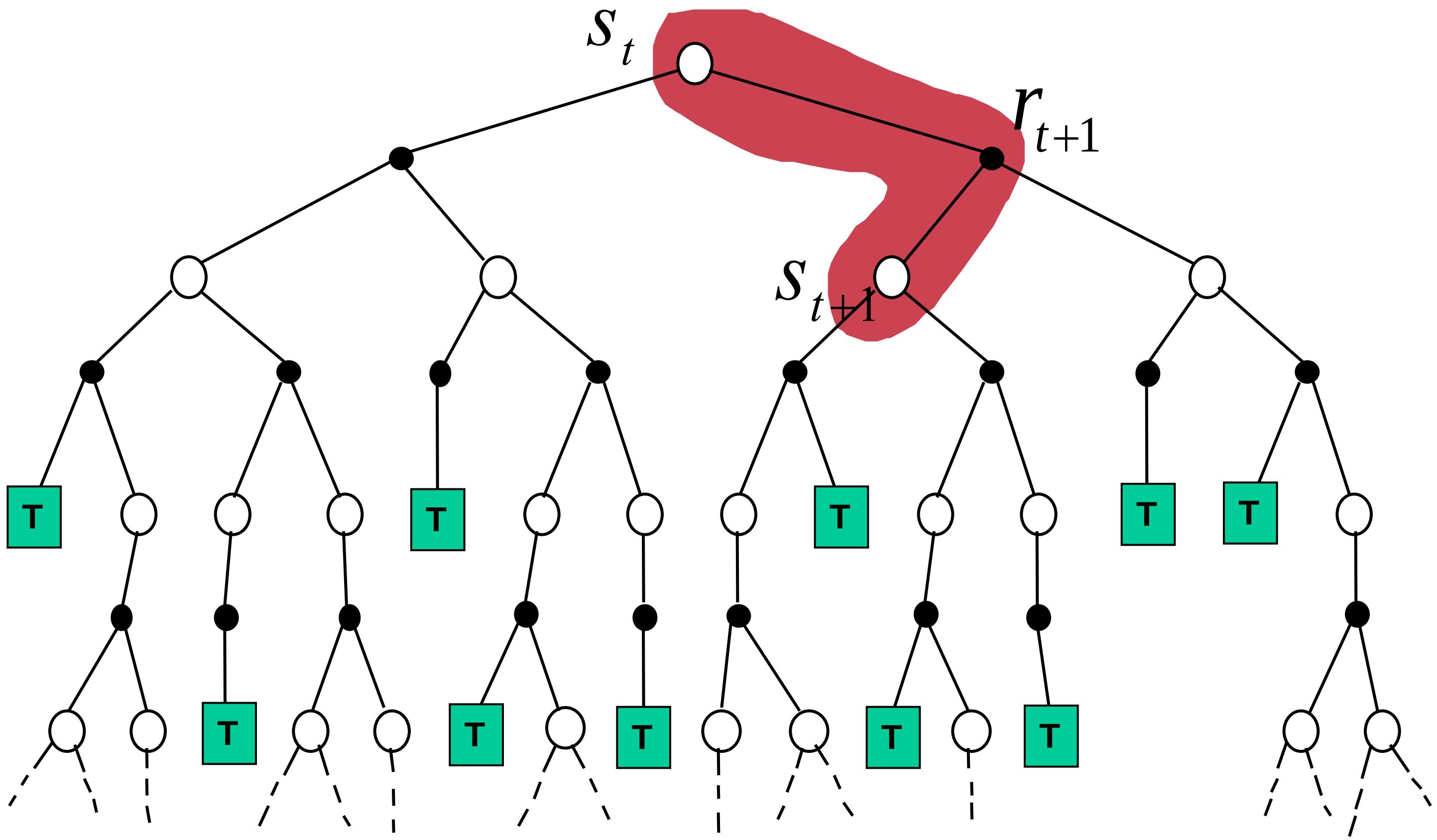


Monte Carlo Backup Diagram



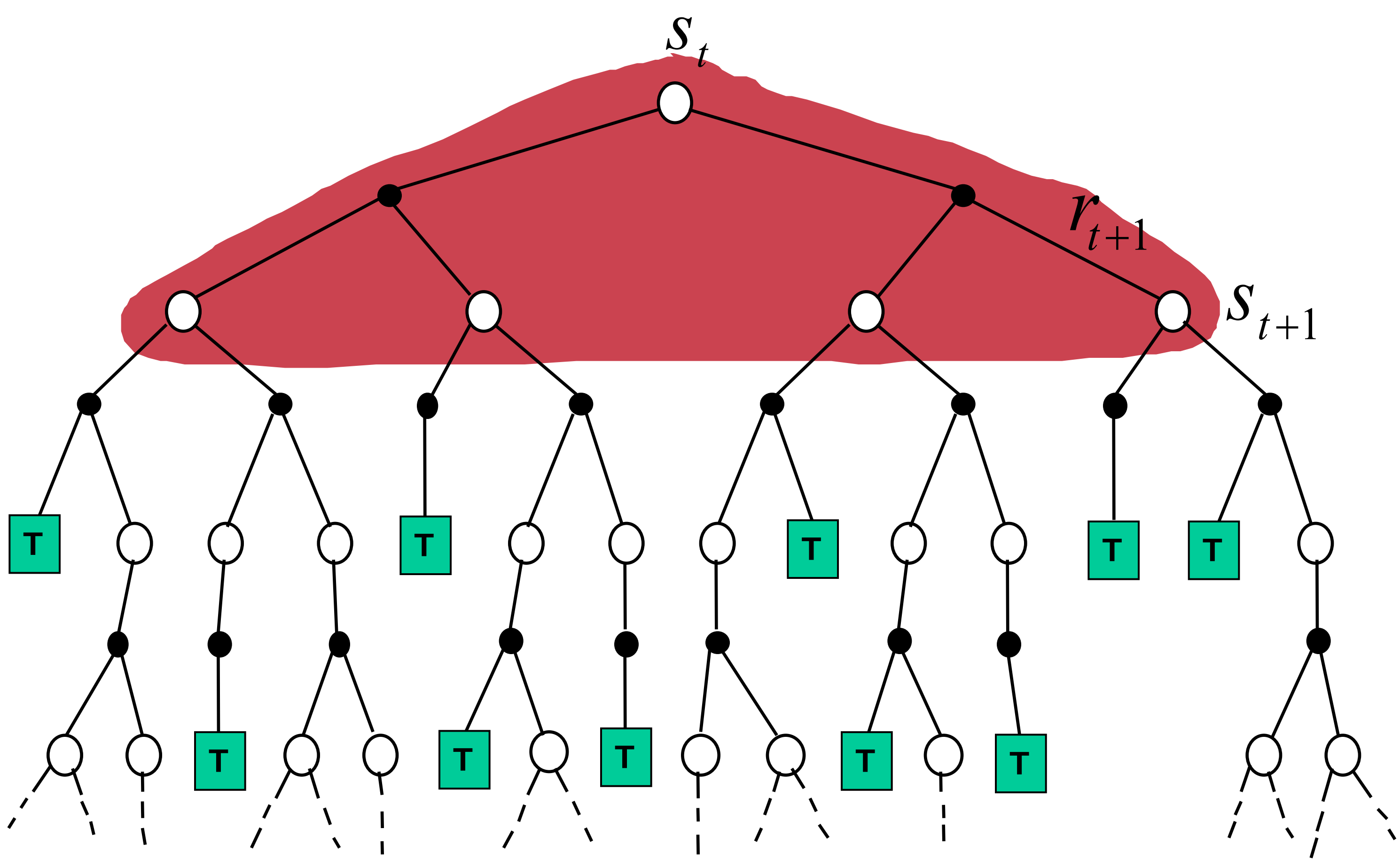
MC estimate: $V_k(s_t) := V_{k-1}(s_t) + \alpha_k \cdot (R_k(s_t) - V_{k-1}(s_t))$

Temporal Differences Backup Diagram



TD estimate: $V_k(s_t) := V_{k-1}(s_t) + \alpha_k \cdot \Big((r_{t+1} + \gamma V_{k-1}(s_{t+1})) - V_{k-1}(s_t) \Big)$

Dynamic Programming Backup Diagram



$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s,a)[R(s,a,s') + \gamma V^\pi(s')]$$

TD for function Q

This was our TD estimate for V:

TD estimate: $V_k(s_t) := V_{k-1}(s_t) + \alpha_k \cdot \Big((r_{t+1} + \gamma V_{k-1}(s_{t+1})) - V_{k-1}(s_t) \Big)$

$$V_k(s_t) := (1 - \alpha_k) \cdot V_{k-1}(s_t) + \alpha_k \cdot \Big((r_{t+1} + \gamma V_{k-1}(s_{t+1})) \Big)$$

We can use the same for Q(s,a):

$$\begin{aligned} Q^\pi(s,a) &= R(s,a,s') + \gamma \sum_{s'} P(s,a,s') \sum_{a'} \pi(s',a') Q^\pi(s',a') \\ &= R(s,a,s') + \gamma E_\pi \left[Q^\pi(s',a') \right] \end{aligned}$$

$$Q_k(s,a) = (1 - \alpha_k) \cdot Q_{k-1}(s,a) + \alpha_k \cdot \left[R(s,a,s') + \gamma Q_{k-1}(s',a') \right]$$

Finding The Optimal Policy with TD

Finding The Optimal Policy with TD

- We already know the Bellman-equation for Q*:

$$\begin{aligned} Q^*(s,a) &= R(s,a,s') + \gamma \sum_{s'} P(s,a,s') \max_{a'} Q^*(s',a') \\ &= R(s,a,s') + \gamma \mathbb{E}\left[\max_{a'} Q^*(s',a')\right] \end{aligned}$$

- **DP update:**

$$Q_k(s,a) = R(s,a,s') + \gamma \sum_{s'} P(s,a,s') \max_{a'} Q_{k-1}(s',a')$$

- **TD update for Q [= Q Learning]**

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

Q Learning Algorithm

- $Q(s,a)$ arbitrary
- For each episode
  - $s{:=}s_0$; $t{:=}0$
  - For each time step t in the actual episode
    - $t{:=}t{+}1$
    - Choose action a according to a policy π e.g. (epsilon-greedy)
    - Execute action a
    - Observer reward r and new state s'
    - $Q(s,a) \;=\; (1-\alpha_t)\cdot Q(s,a) + \alpha_t \cdot \left[r + \gamma \max_{a'} Q(s',a')\right]$
    - $s{:=}s$'
  - End For
- End For

Q Learning Algorithm

- **Q-learning learns an optimal policy** no matter which policy the agent is actually following (i.e., which action *a* it selects for any state *s*)

  as long as there is no bound on the number of times it tries an action in any state (i.e., it does not always do the same subset of actions in a state).

- Because it learns an optimal policy no matter which policy it is carrying out, it is called an **off-policy** method.