Reinforcement learning

Advanced Machine Learning

Janosch Bajorath

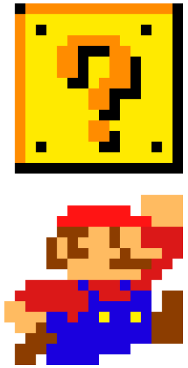
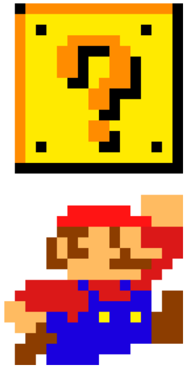


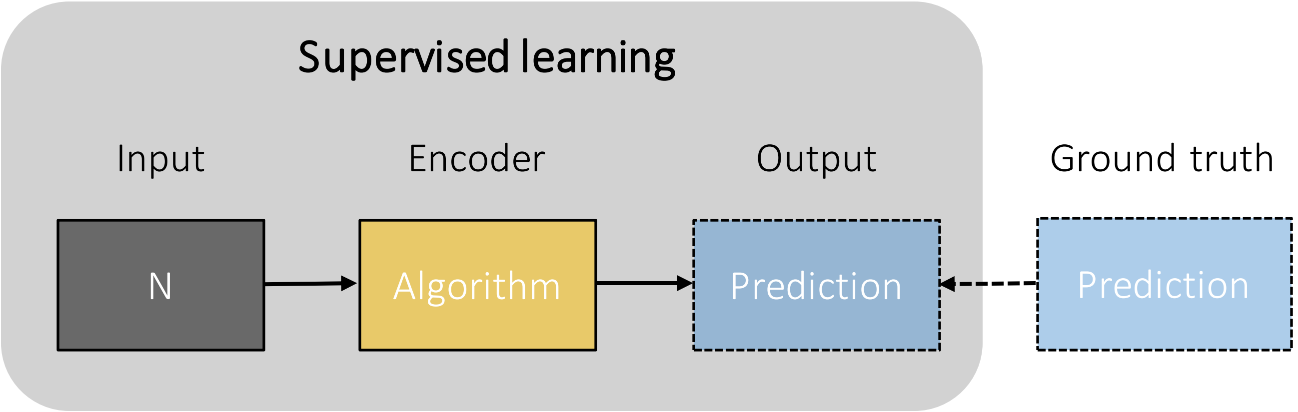
Table of contents

1. Overview of reinforcement learning

2. Basic elements of reinforcement learning
3. Generalized policy iteration

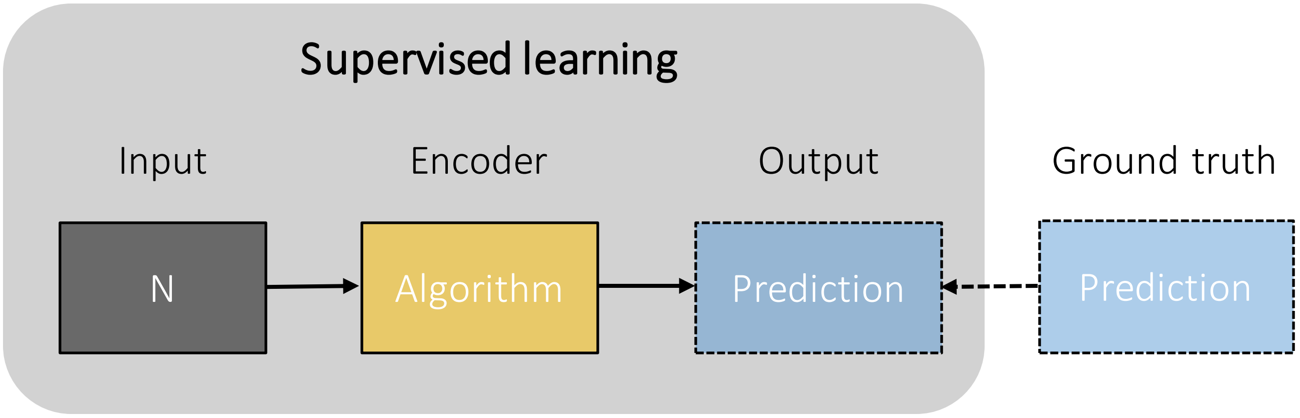
4. Dynamic programming
5. Monte Carlo learning
6. Temporal difference learning – TD(0)
   - Q-learning





Supervised learner has an informed external supervisor, that provides information about the examples provided

→ Examples provided
→ learn patterns from **instruction**



Supervised learning

Input
N

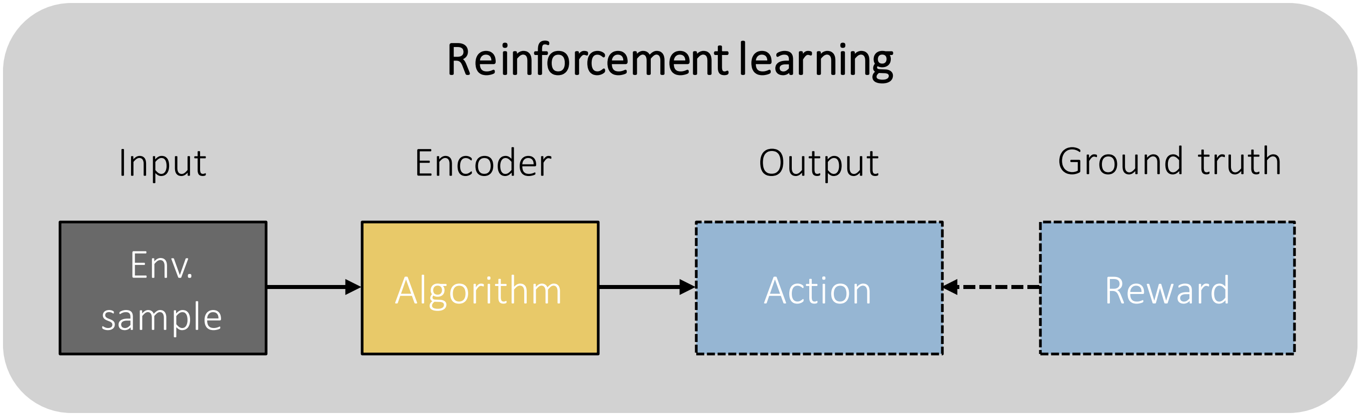
Encoder
Algorithm

Output
Prediction

Ground truth
Prediction

Supervised learner has an informed external supervisor, that provides information about the examples provided

→ Examples provided
→ learn patterns from **instruction**



Reinforcement learning

Input
Env. sample

Encoder
Algorithm

Output
Action

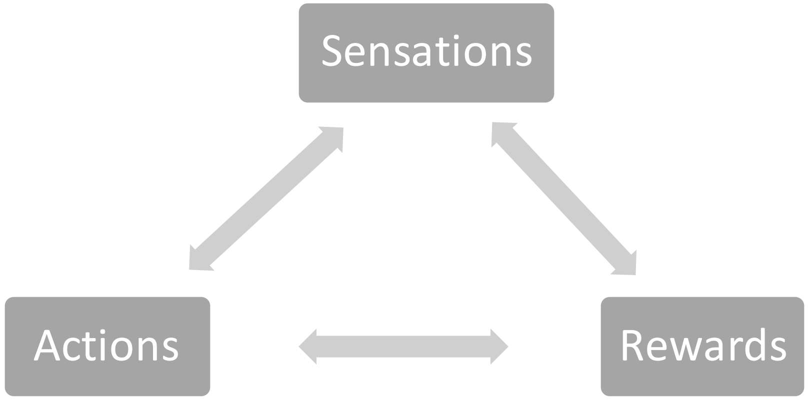
Ground truth
Reward

Reinforced learner (RL agent) must learn from its interaction with the environment

→ Environment provided
→ learn patterns from **exploration** (trial and error search)

Aspects of an RL-agent

Aspects of an RL-Problem

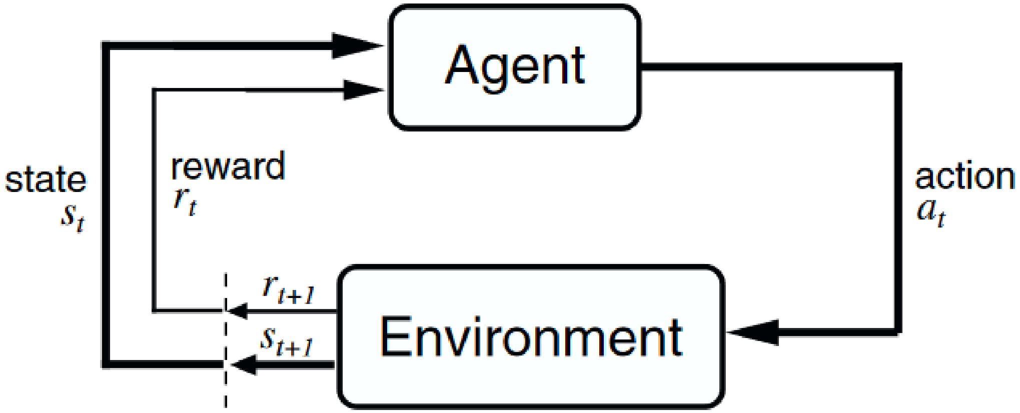


Sensations

Actions

Rewards

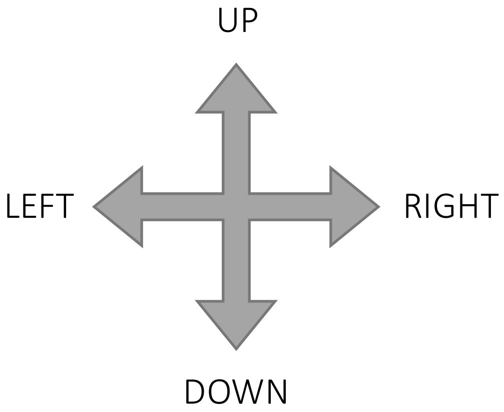
Agent-Environment interface



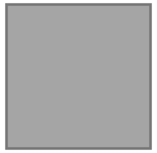
$\boxed{s_0, a_0, r_1,} s_1, a_1, r_2.....r_{T-1}, \boxed{s_{T-1}, a_{T-1}, r_T,} s_T$

- Discrete time steps with delayed reward

Agent

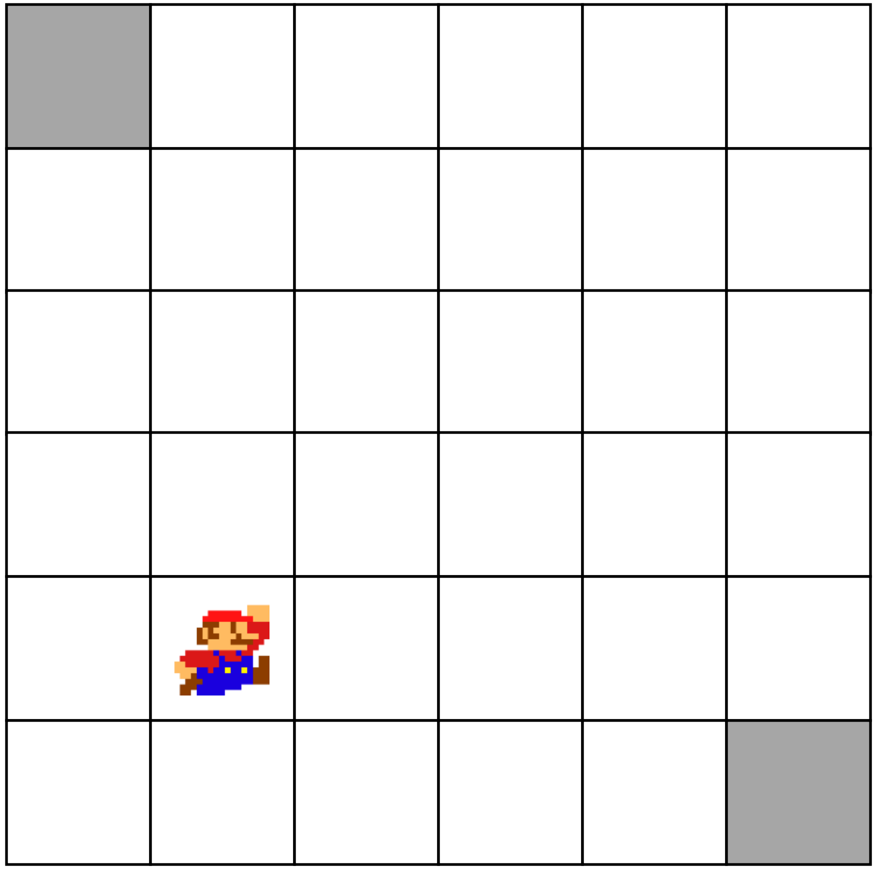


Possible actions - A(s)
(deterministic)



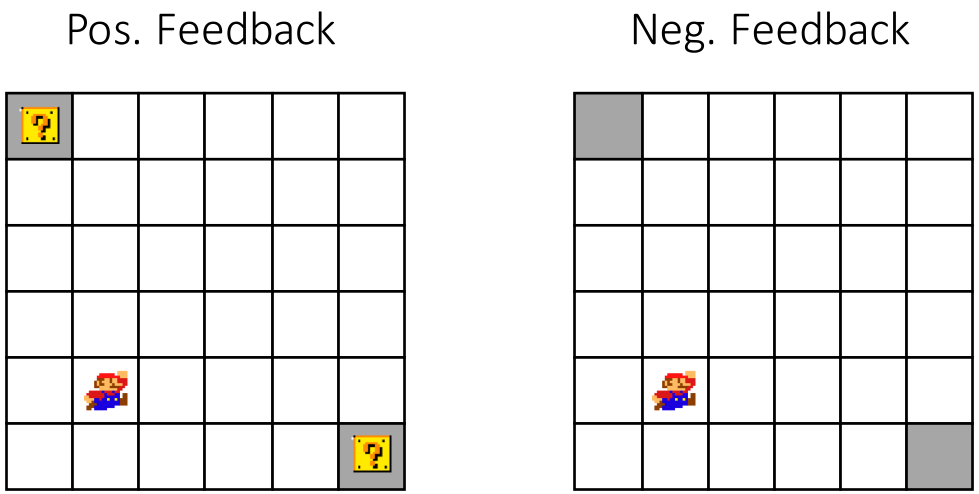
Terminal state – s_T
(Goal)

The Environment
36 states



“Immediate desirability of a state”

- Mapping from perceived *(s, a, s')* tuple to a reward value
- **Goal of a RL-agent**: maximize cumulative Reward



Important: Design of the reward structure is cruical for agent behaviour!

Return

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$$

Discounted Return

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Discount factor

γ = 0 -> myopic
γ = 1 -> far sighted

Connetion between returns

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots.$$
$$= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \ldots.)$$
$$= r_{t+1} + \gamma R_{t+1}$$

Policy function

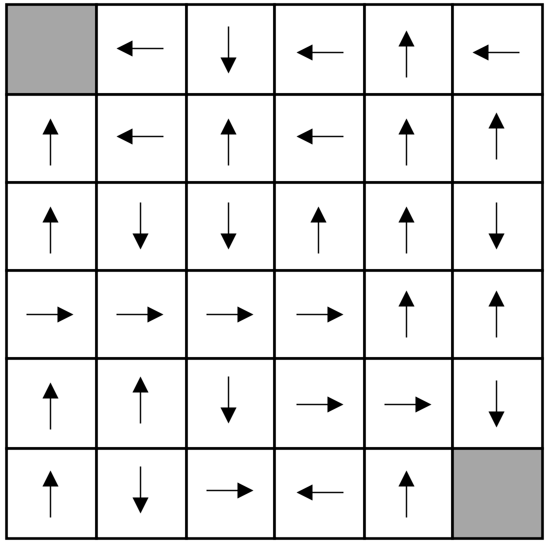
$$\pi_t(s)$$

- Defines behavior (action) of an agent
- Function that maps from perceived states of the environment to actions taken (stimulus-response rules)
- **Optimal policy:**

“Which action to take for the greatest reward”

$$\pi_t{}^*(s)$$

Deterministic policy



$$\pi_t(s) = a$$

Stochastic policy

$$\pi_t(a, s) = P[a_t = a | s_t = s]$$

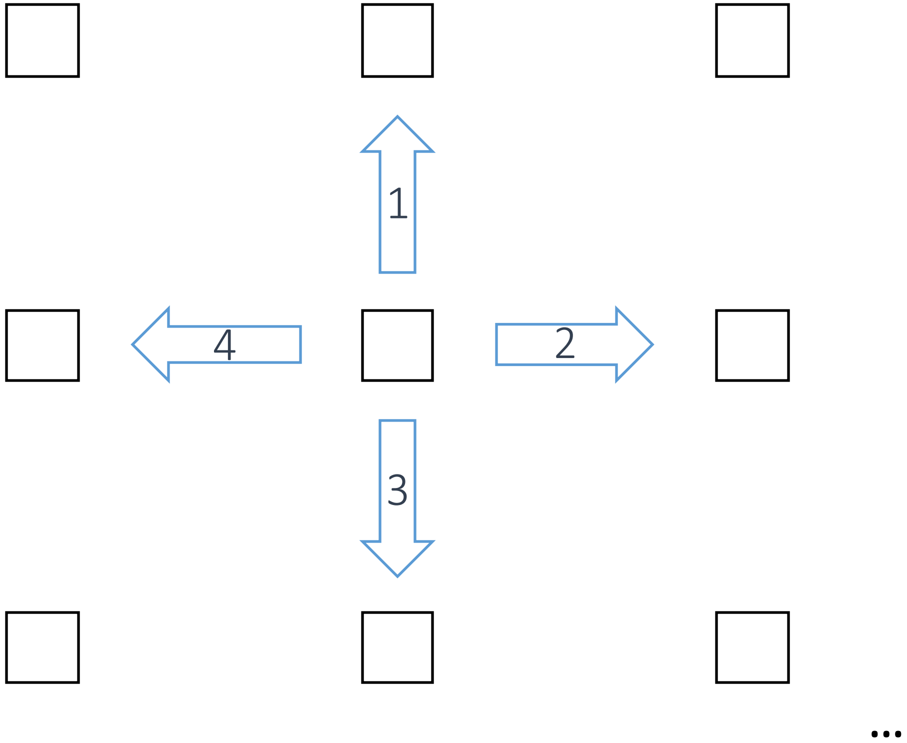
Value Tables

$V^{\pi}(s)$ State-Value

1	2	3
4	5	6
7	8	9

-> Vector with *s ε S* entries

$Q^{\pi}(s, a)$ Action-Value

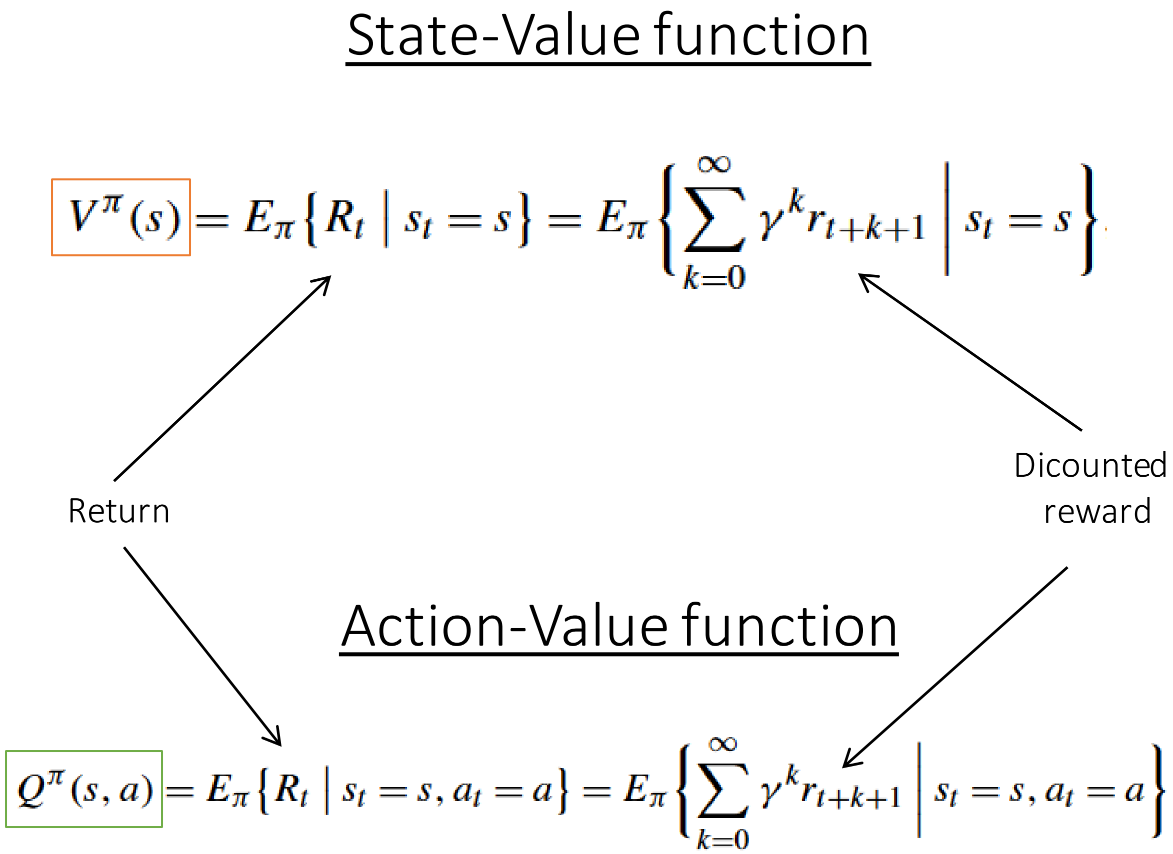


-> Matrix with *(a ε A)* x *(s ε S)* entries

Value functions and -tables

The “Goodness of a state”

- Expected Return an agent can accumulate when starting from a specific state and complying with the policy (π)
- Long-term desirability of a state, taking possible following states and their reward into account
- Bellman Expectation Equation



State-Value function

$$\boxed{V^{\pi}(s)} = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s\right\}$$

Return

Dicounted reward

Action-Value function

$$\boxed{Q^{\pi}(s,a)} = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right\}$$

Representation matters – Curse of dimensionality

Breakout

State-Space:

Image ratio – 210, 160, 3

Intensity range – 256

$256^{210x160x3}$ possible states

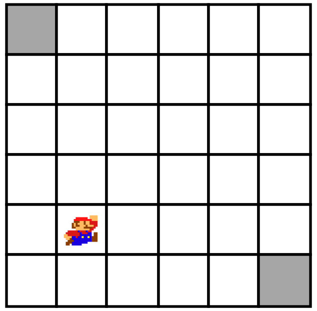
$\approx 10^{182063} >> 10^{82}$ atoms in the universe

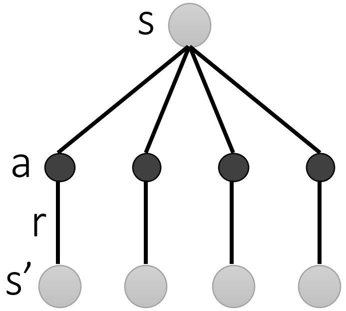
Model of the environment

Environment Dynamics

- Function that mimics the behavior of the environment
- Given a state and action, model predicts resulting next state and reward

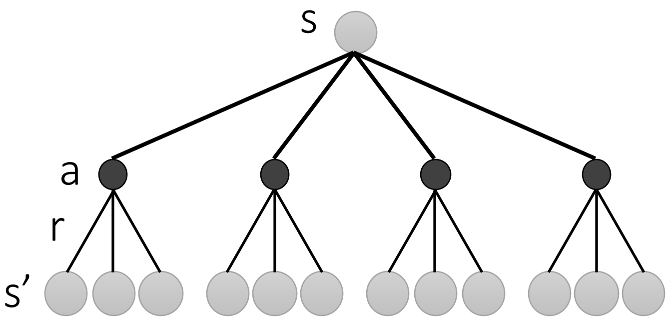
Deterministic Model





Fixed successor state
Fixed reward

Stochastic Model



$$\mathcal{P}^{a}_{ss'} = \Pr\left\{ s_{t+1} = s' \;\middle|\; s_t = s, a_t = a \right\}$$

$$\mathcal{R}^{a}_{ss'} = E\left\{ r_{t+1} \;\middle|\; s_t = s, a_t = a, s_{t+1} = s' \right\}$$

State representation

State signal = information available to the agent on which he makes decisions

- Physical world evolves through time from step to step with specific dynamics:

$$\Pr\left\{s_{t+1}=s', r_{t+1}=r \,\middle|\, s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\right\}$$

- Independence of path property:

$$\Pr\left\{s_{t+1}=s', r_{t+1}=r \,\middle|\, s_t, a_t\right\}$$

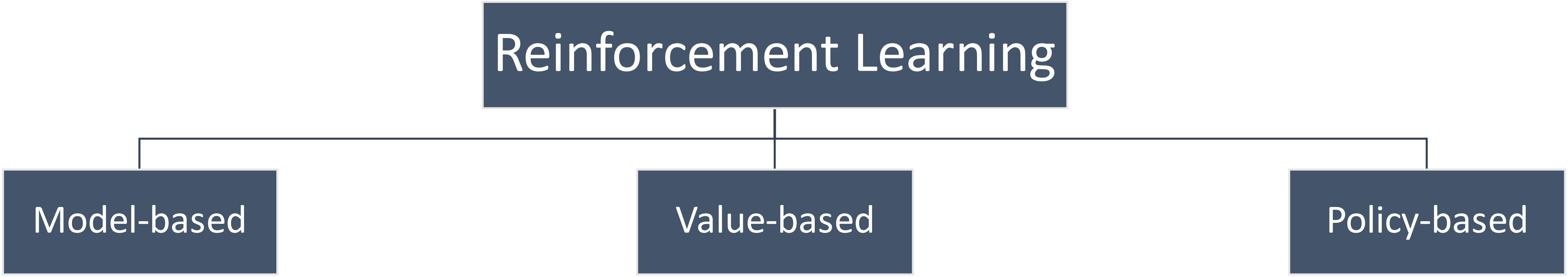
Reward

$$\mathcal{R}^a_{ss'} = E\left\{r_{t+1} \,\middle|\, s_t=s, a_t=a, s_{t+1}=s'\right\}$$

Transition Dynamics

$$\mathcal{P}^a_{ss'} = \Pr\left\{s_{t+1}=s' \,\middle|\, s_t=s, a_t=a\right\}$$

-> State signal that retains all relevant information is said to be Markov



Reinforcement Learning

Model-based

- Learn the model of the environment, then use the model for planning
- many re-planning calculations to generate optimal policy
- Learned model needs to be updated frequently

Value-based

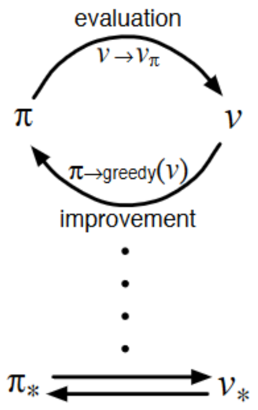
- Learn the state- or action-value function of an environment
- Decide among actions based on the value function
- Algorithms need to take active exploration into account

Policy-based

- Learn the imminent behavior of the agent, by approximating the optimal policy
- Samples from the policy space
- Exploration as inherent feature
- evolutionary methods

Generalized Policy Iteration

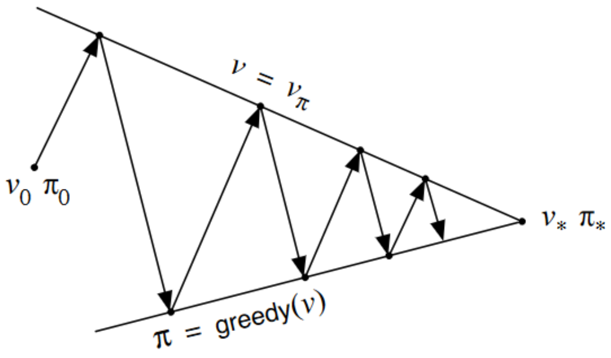
A universal approach to solve a RL-Problem



-> Two simultaneous interacting processes

1. Predicting desired and undesired states – **Policy evaluation**
   → Making value-function consistent with the current policy
2. Improve the control of the agent – **Policy improvement**
   → Making the policy greedy with respect to the current value-function

---



- Interaction between policy and value functions until both stabilize and optimal functions are reached

Idea: Dividing a problem into sub-problems (**divide and conquer**) while using these solutions to solve similar sub-problems (**bootstrapping**)

Update policy by greedily evaluating the obtained Value function

Perquisites:

- Assumes that the problem is defined by a finite MDP <s, P, a, R, γ>
- Dynamics of the environment need to be known (transition probabilities and immediate rewards)

- Basically, the simplest implementation of GPI
- No need to sample from the environment as Model is known

Bellman expectation equation – self-consistency condition

- Recursive relationship between the value function of the current state s and the successor state s’

$$\boxed{V^\pi(s)} = E_\pi\{R_t \mid s_t = s\}$$

$$= E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \,\middle|\, s_t = s\right\}$$

Recurrency of Return

$$= E_\pi\left\{r_{t+1} + \gamma \underbrace{\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}}_{\gamma V^\pi(s')} \,\middle|\, s_t = s\right\}$$

Immediate Reward

Future discounted Return

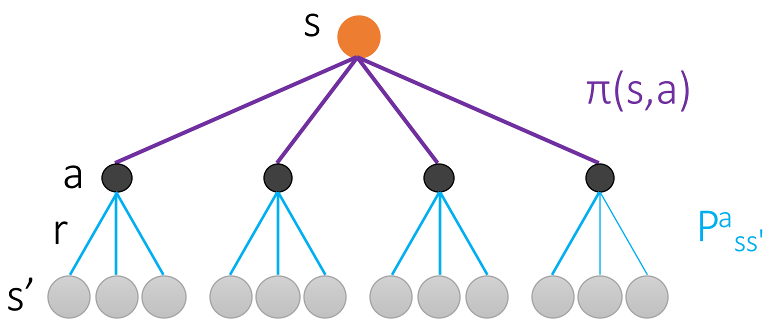
$$= E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s\}$$

$$= \boxed{\sum_a \pi(s,a)}\boxed{\sum_{s'} \mathcal{P}^a_{ss'}}\left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right]$$

Action probability

Transition probability

Backup diagramm for DP



Solving the prediction problem with Bellmann

Turning the Bellman expectation equation into an iterative update equation

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s\} \\ &= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right] \end{aligned}$$

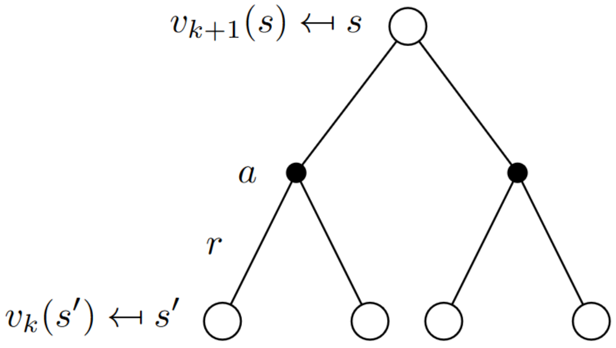
$$\longrightarrow$$

new estimate

Old estimate

$$\begin{aligned} V_{k+1}(s) &= E_{\pi}\{r_{t+1} + \gamma V_k(s_{t+1}) \mid s_t = s\} \\ &= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V_k(s')\right] \end{aligned}$$

- Updating exectation estimates based on expectation estimates
- Synchronous full backups in a step-by-step approach



Policy Evaluation

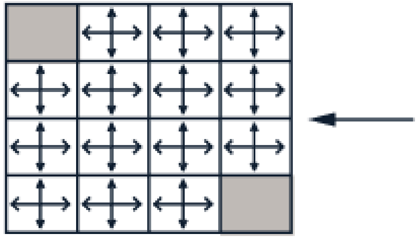
Reward: -1 per step
Discount factor: 1

$$V_{k+1}(s) = \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V_k(s') \right]$$

v_k for the Random Policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



random policy

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

v_k for the Random Policy

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Optimal value functions

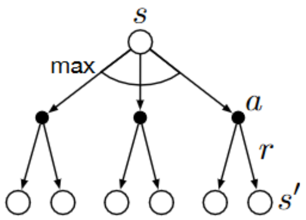
- Defines partial ordering over policies

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

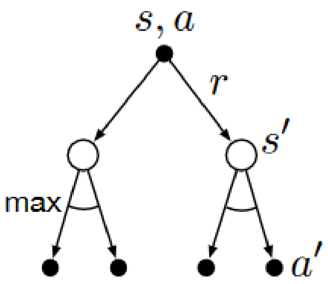
$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

Bellman optimality equation

$$\begin{aligned} V^*(s) &= \max_a E_{\pi^*}\{R_t \mid s_t = s, a_t = a\} \\ &= \max_a E_{\pi^*}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \,\middle|\, s_t = s, a_t = a\right\} \\ &= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\} \\ &\boxed{= \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right].} \end{aligned}$$



$$Q^*(s,a) = \boxed{E\left\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \,\middle|\, s_t = s, a_t = a\right\}}$$



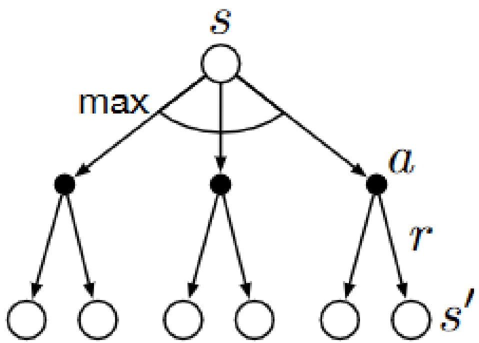
- No knowledge about the environment's dynamics as well as states needed

The Control problem – how to improve the old policy?

- Policy encoded in the state-value function $V^{\pi}(s)$
- deterministic policy - Acting **greedy** in respect to the state-value function

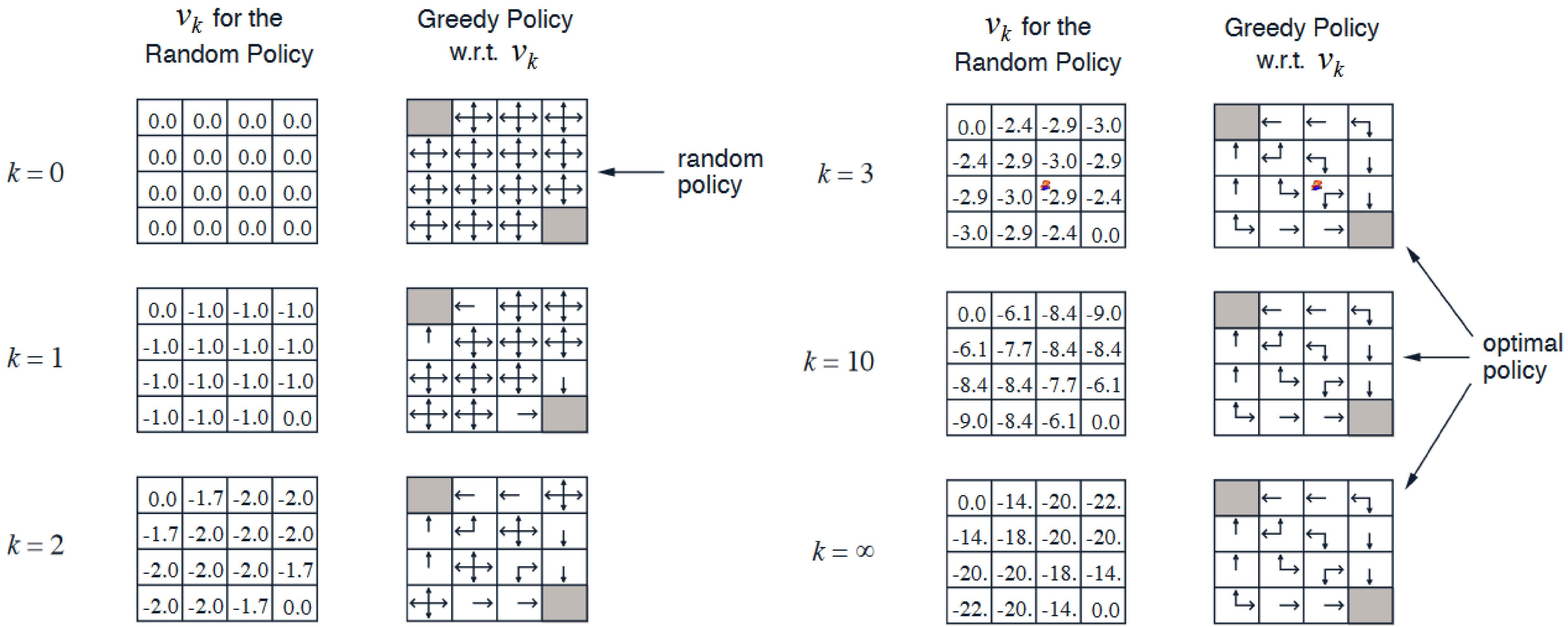
$$\pi'(s) = \arg\max_a E\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \,\middle|\, s_t = s, a_t = a\right\}$$

$$\boxed{= \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right],}$$



Policy Improvement

$$\pi'(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right]$$



$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

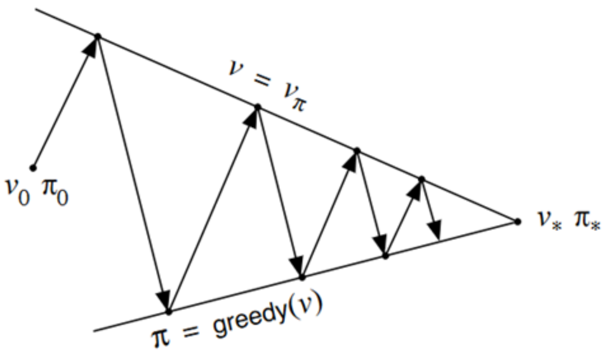
0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

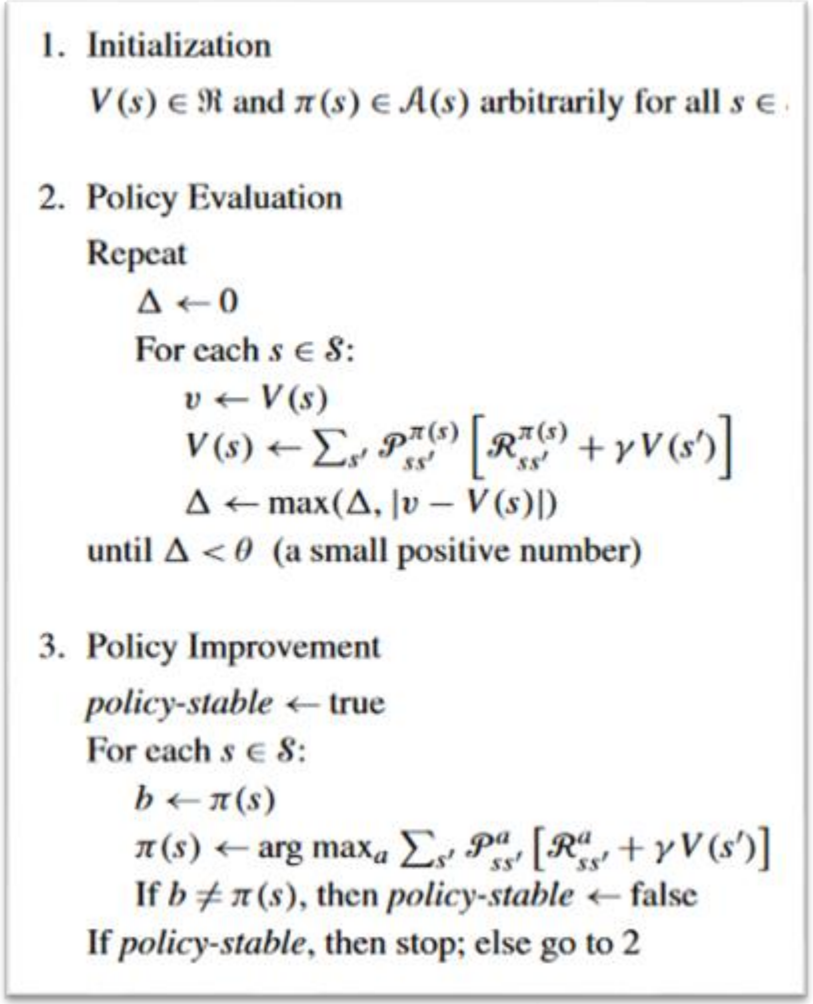
$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

$$\pi^0 \xrightarrow{E} V^{\pi 0} \xrightarrow{I} \pi^1 \xrightarrow{E} V^{\pi 1} \xrightarrow{I} \ldots\ldots \xrightarrow{E} V^{\pi *} \xrightarrow{I} \pi^*$$

- Application of policy evaluation and policy improvement in a GPI pattern until value and policy function converge to their respective optima
- Finite MDP has finite set of policies, hence convergence is guaranteed in finite steps





```
1. Initialization
   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in$

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[ \mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ true
   For each $s \in \mathcal{S}$:
      $b \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[ \mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$
      If $b \neq \pi(s)$, then policy-stable $\leftarrow$ false
   If policy-stable, then stop; else go to 2
```

Idea

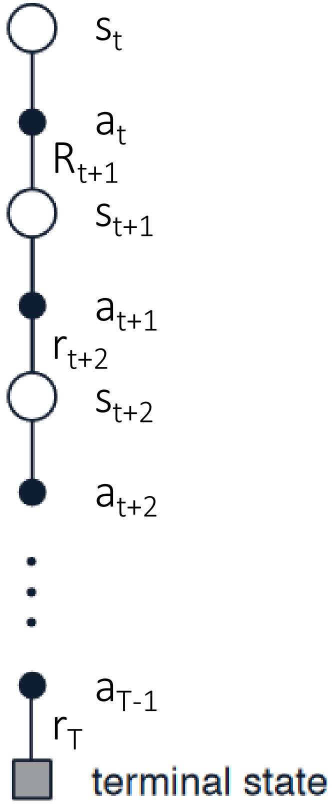
$$\xcancel{\begin{gathered}\mathcal{P}_{ss'}^{a} = \Pr\left\{s_{t+1} = s' \,\middle|\, s_t = s, a_t = a\right\} \\ \mathcal{R}_{ss'}^{a} = E\left\{r_{t+1} \,\middle|\, s_t = s, a_t = a, s_{t+1} = s'\right\}\end{gathered}}$$

Problem: Knowledge of entire dynamics most of the time not given

→ Estimate action-value function and discover policies based on average returns

Update Value estimate after (s,a) visit

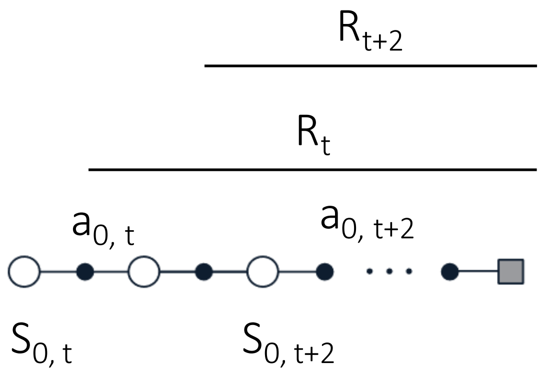
$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t \,\middle|\, s_t = s, a_t = a\right\} \quad \longleftarrow \quad Q^{\pi}(s,a) \leftarrow \frac{1}{k}\sum_{k=0}^{\infty} R_k$$



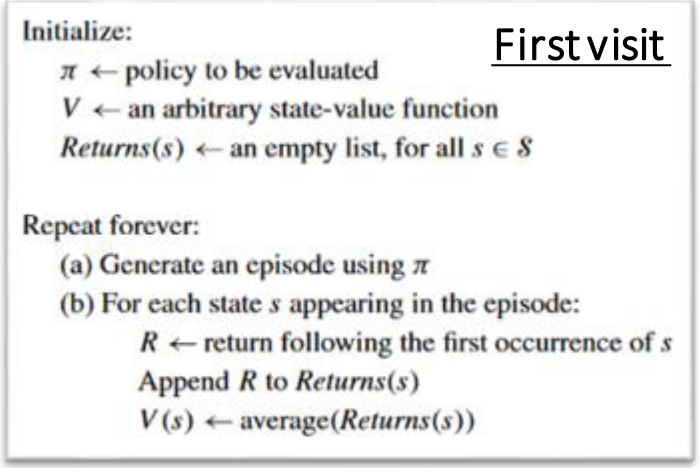
- Only suited for episodic tasks, as Return needs to be well defined
- Extension of DP where only sample experience is available

When to update the Value function?

1. Every visit MC prediction
   - Update $Q^{\pi}(s)$ every time we encounter state s
2. First visit MC prediction
   - Update $Q^{\pi}(s)$ the first time we encounter state s



- Updates are made in an episode-by-episode sense
- No bootstrapping, estimated Value functions are independent of each other



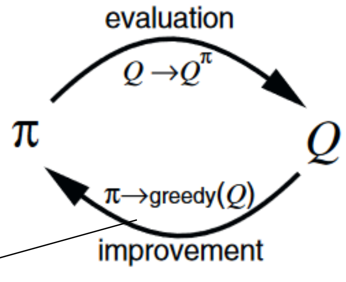
First visit

```
Initialize:
    $\pi \leftarrow$ policy to be evaluated
    $V \leftarrow$ an arbitrary state-value function
    $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:
    (a) Generate an episode using $\pi$
    (b) For each state $s$ appearing in the episode:
            $R \leftarrow$ return following the first occurrence of $s$
            Append $R$ to $Returns(s)$
            $V(s) \leftarrow$ average($Returns(s)$)
```

Problem: no guarantee of exploration with greedy policy improvement

→ How to keep exploring while we sample with a policy, as well as optimize that policy?



Monte Carlo with exploring starts:

$$\pi(s) = \arg\max_a Q(s, a) \longrightarrow$$ Deterministic policy

Monte Carlo with ε-soft policy improvement

- No need for exploring starts, as exploration is encoded in policy

Suboptimal/non greedy action: $\pi(s, a) = \frac{\varepsilon}{A(s)}$

Probability of taking an action under policy π

$$\pi(s, a) \geq \frac{\epsilon}{|\mathcal{A}(s)|}$$

Optimal/greedy action: $\pi(s, a) = 1 - \varepsilon + \frac{\varepsilon}{A(s)}$

Idea

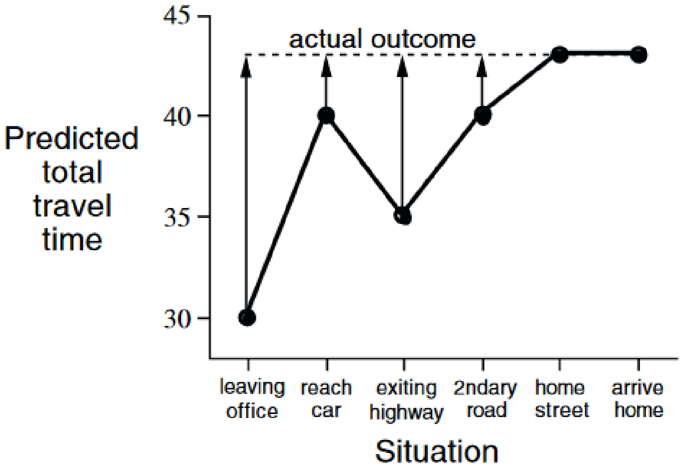
Bootstrapping

→ update estimated based on other learned estimates (like DP-Methods)

Episode sampling

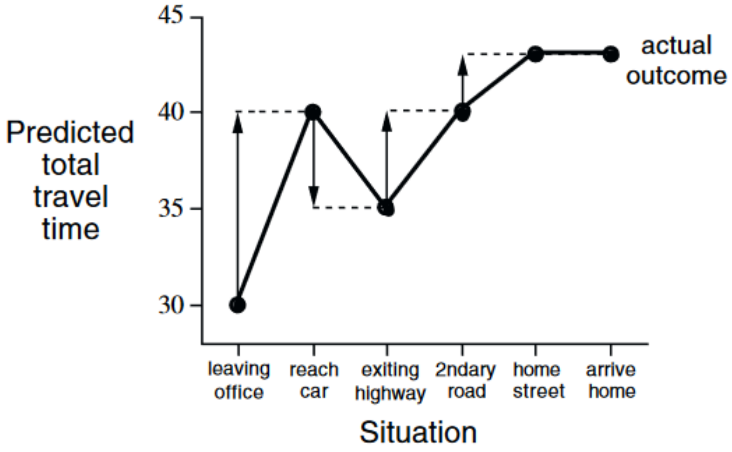
→ no prior knowledge of the environment dynamics (like MC-Methods)

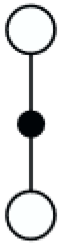
MC learning



Adjust expectation based on the actual Return

TD learning





Adjust expectation based on the immediate reward and expected Return

Recency weighted estimate

- Tracking non-stationary problems

new estimate step size old estimate Target value

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i .
\end{aligned}
$$

Step-by-step approach

new estimate old estimate

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$

step size TD error

TD target Value

$0 > \alpha \geq 1$

α close to 0 -> higher weighting of old errors

α close to 1 -> higher weighting of new errors

- Bellman optimality equation, where Value of next state $V_k(s_{t+1})$ is adjusted by the previous estimation of $V_{k-1}(s_{t+1})$ and the received reward

Off policy approach to solve the RL control problem by the TD-learning

Behavior policy – used to sample from the environment (e.g., ε-soft policy)

Target policy – used for control after training (greedy policy)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \boxed{\max_a Q(s_{t+1}, a)} - Q(s_t, a_t) \right]$$

- Both policies derived from the action value Q(s, a)
- Exploration encoded in behavior policy s. t. target policy can be optimal/greedy with respect to Q (s, a)

ε-soft policy

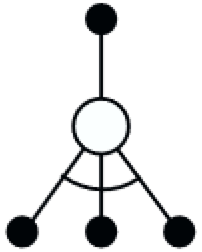
$$\boxed{\pi(s,a) = \frac{\varepsilon}{A(s)} \qquad \pi(s,a) = 1 - \varepsilon + \frac{\varepsilon}{A(s)}}$$

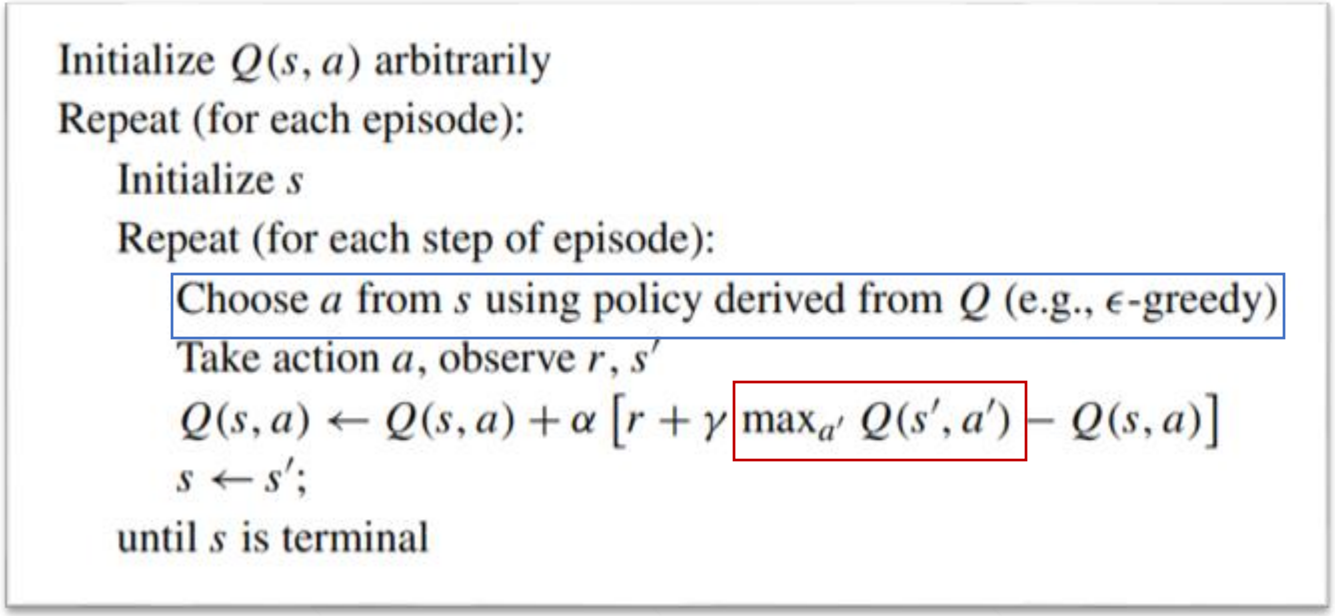
Greedy policy

$$\boxed{\pi'(s) = \arg\max_a Q^{\pi}(s,a)}$$

Off-policy TD-Control – Q-learning

Q-learning backup diagram





```
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
    Initialize $s$
    Repeat (for each step of episode):
        Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
        Take action $a$, observe $r$, $s'$
        $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \boxed{\max_{a'} Q(s', a')} - Q(s, a)\right]$
        $s \leftarrow s'$;
    until $s$ is terminal
```

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \boxed{\max_a Q(s_{t+1}, a)} - Q(s_t, a_t) \right]$$

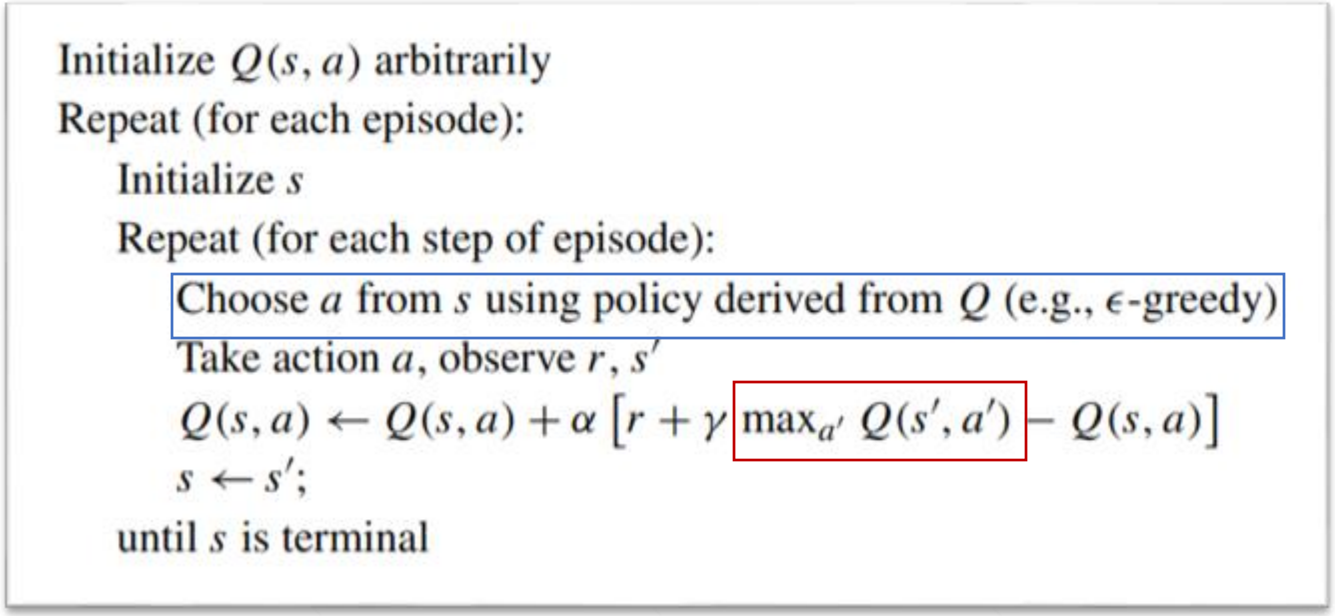
ε-soft policy

$$\boxed{\pi(s,a) = \frac{\varepsilon}{A(s)} \qquad \pi(s,a) = 1 - \varepsilon + \frac{\varepsilon}{A(s)}}$$

Greedy policy

$$\boxed{\pi'(s) = \arg\max_a Q^{\pi}(s, a)}$$

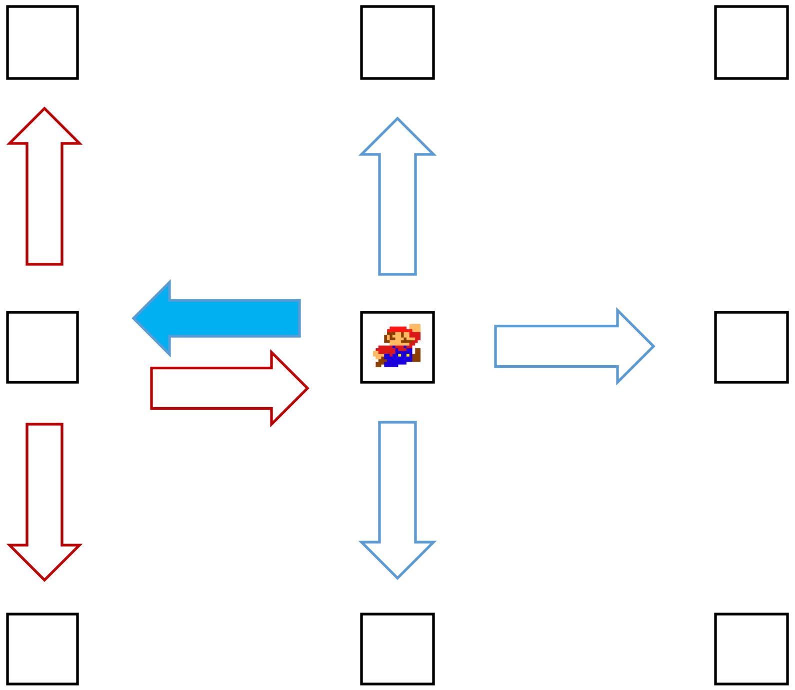
Off-policy TD-Control – Q-learning



```
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
    Initialize $s$
    Repeat (for each step of episode):
        $\boxed{\text{Choose } a \text{ from } s \text{ using policy derived from } Q \text{ (e.g., } \epsilon\text{-greedy)}}$
        Take action $a$, observe $r$, $s'$
        $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \boxed{\max_{a'} Q(s', a')} - Q(s, a)\right]$
        $s \leftarrow s'$;
    until $s$ is terminal
```

$\boxed{Q^{\pi}(s, a)}$

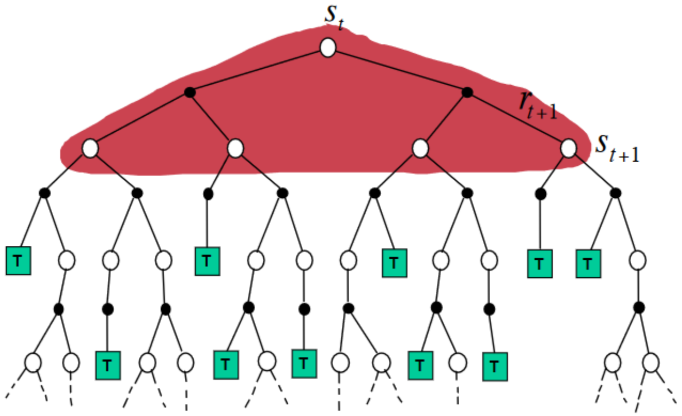
Action-Value



Backup diagrams

Dynamic programming

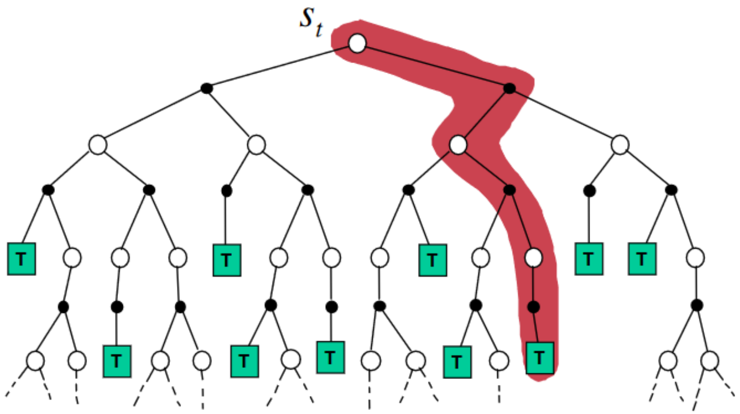
$$V(s_t) \leftarrow E_\pi[R_{t+1} + V(S_{t+1})]$$



- Full backup
- Bootstrapping

Monte Carlo learning

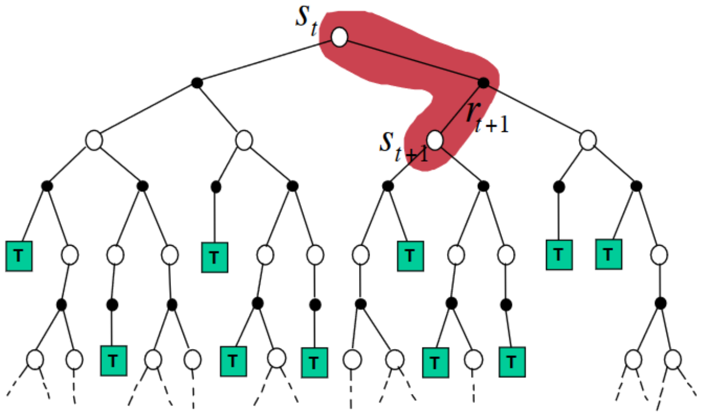
$$V(s_t) \leftarrow V(s_t) + \alpha[Rt - V(St)]$$



- sample backup
- no Bootstrapping

Temporal difference learning TD(0)

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$



- sample backup
- Bootstrapping

1. Introduction to Reinforcement learning by A. Barto and R. Sutton, 1st edition
2. Reinforcement learning lecture by D. Silva (University College London)
3. Introduction to Deep Reinforcement Learning lecture by Lex Fridman (MIT 6.S09)
4. Machine Learning, Ethem Alpaydin (Chapter 16)
5. Machine Learning, Tom M. Mitchell (Chapter 19)

Code examples: https://marcinbogdanski.github.io/reinforcement-learning.html