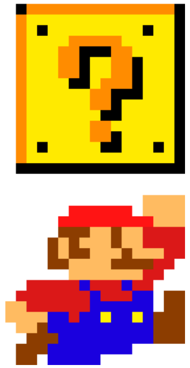
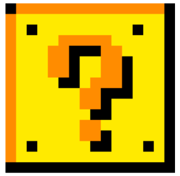
Reinforcement learning

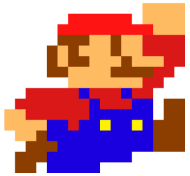
Advanced Machine Learning

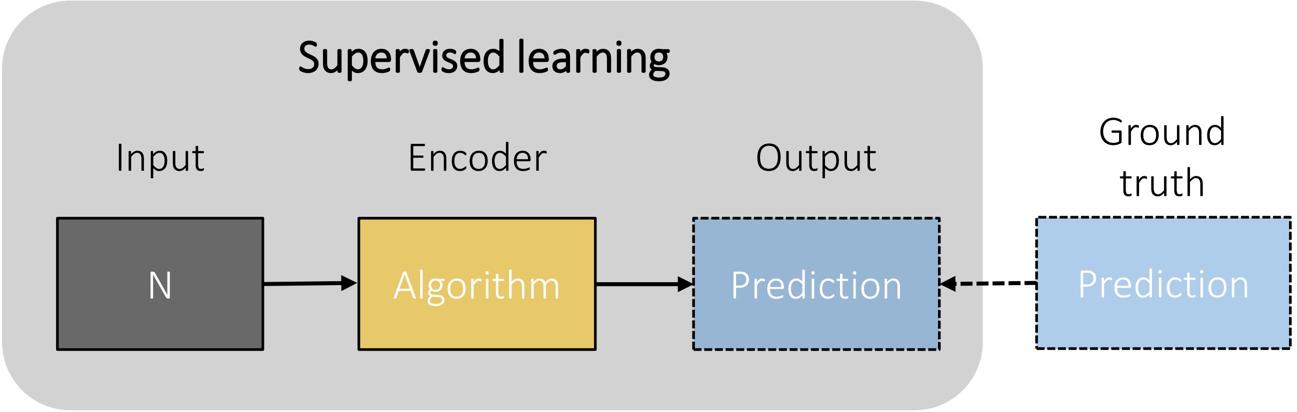
Janosch Bajorath

UNIVERSITÄT OSNABRÜCK



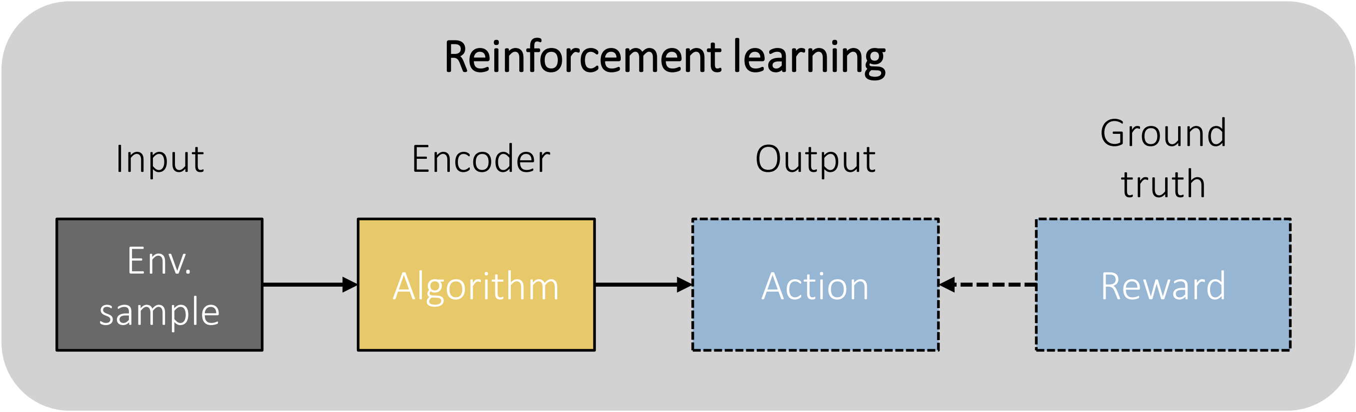






Supervised learner has an informed external supervisor, that provides information about the examples provided

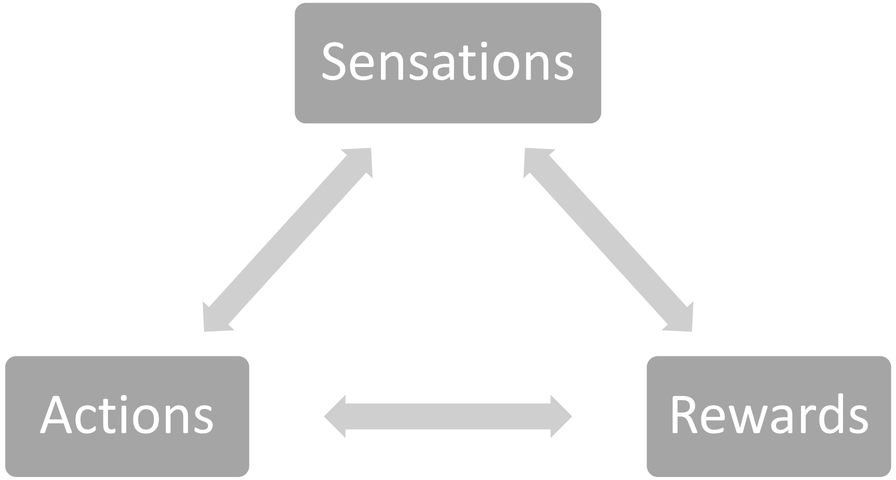
→ Examples provided, learn patterns from them



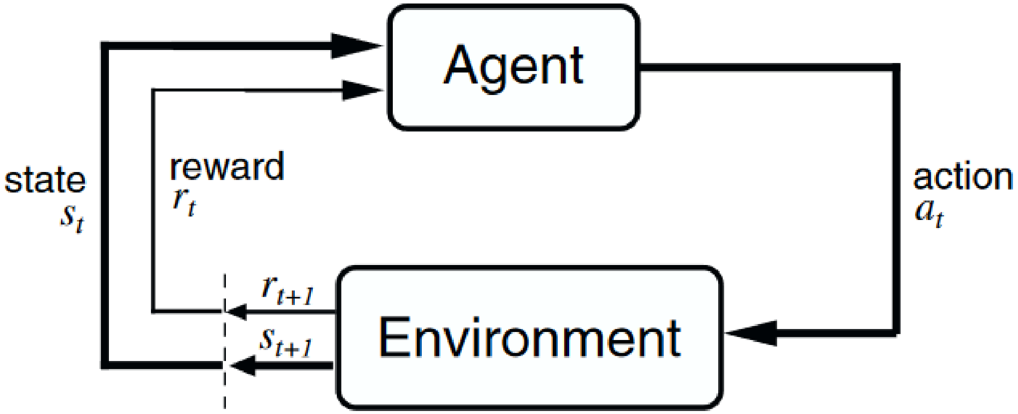
Reinforced learner (RL agent) must learn from its interaction with the environment

→ World provided, learn patterns by exploring

Aspects of an RL-agent



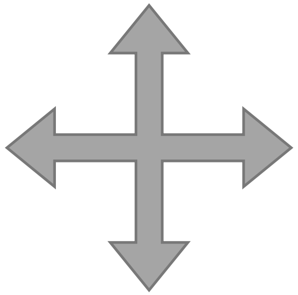
Agent-Environment interface



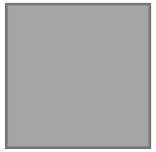
$\boxed{s_0, a_0, r_1,}\ s_1, a_1, r_2.....r_{T-1}, \boxed{s_{T-1}, a_{T-1}, r_T,}\ s_T$

- Discrete time steps with delayed reward (R_{t+1})
- Trial and error search for optimal behavior
- Defined by characterizing learning problem (not method)

Agent

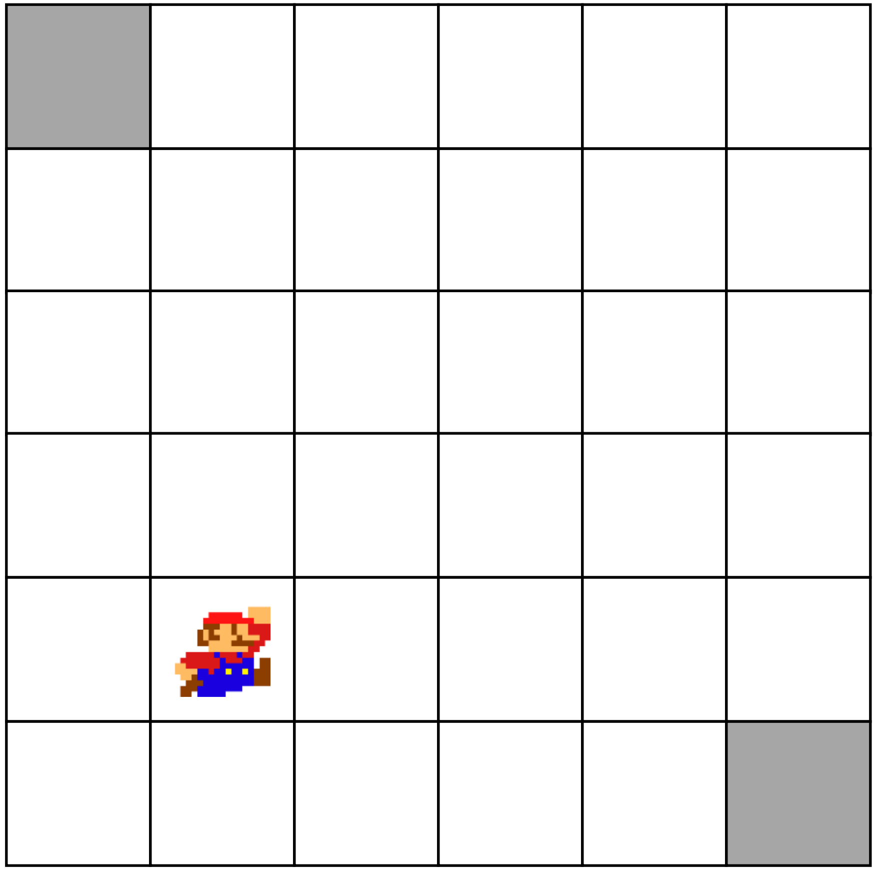


Possible actions - A(s)
(deterministic or stochastic)



Terminal state - s_T

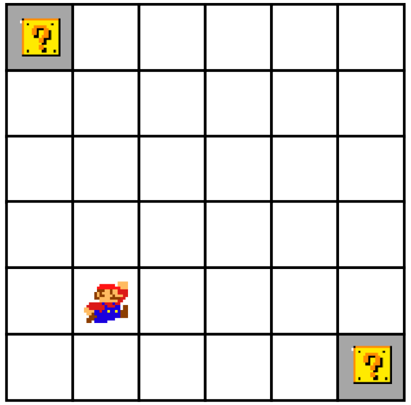
The Environment – $s \varepsilon S$



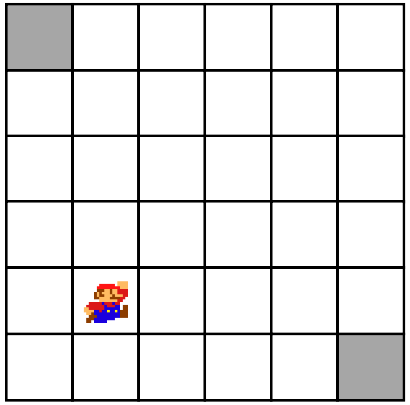
"Immediate, intrinsic desirability of a state or state-action"

- Mapping from perceived state *(s)* or state action pair *(s, a)* to a reward value
- Goal of a RL-agent: maximize cumulative Reward *R*

Pos. Feedback



Neg. Feedback



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

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots \\ &= r_{t+1} + \gamma (r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \ldots) \\ &= r_{t+1} + \gamma R_{t+1} \end{aligned}$$

[2]

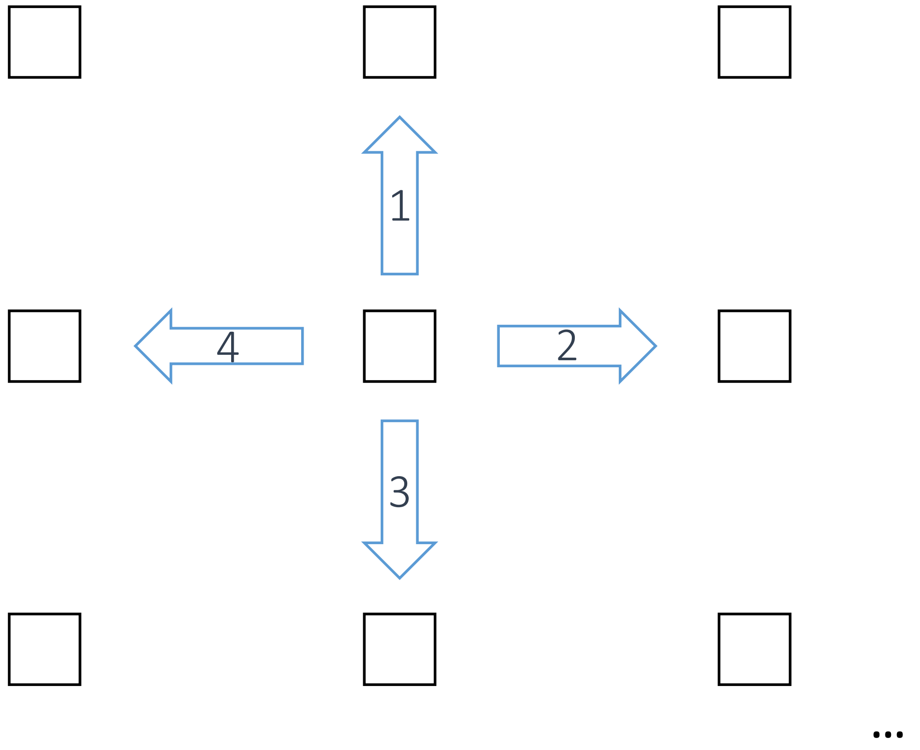
Value Tables

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

1	2	3
4	5	6
7	8	9

→ Vector with *s ε S* entries

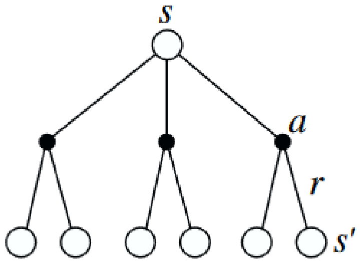
$Q^{\pi}(s, a)$ Action-Value Table



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

The “Goodness of a state”

- Expected reward an agent can accumulate when starting from that state and complying with the policy (π)
- Long-term desirability of a state, taking possible following states and their reward into account
- Used in value-estimation based learning



State-Value function

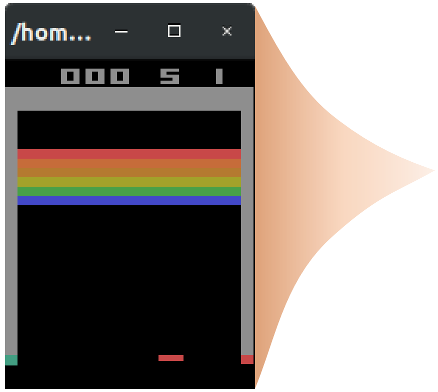
$$\boxed{V^{\pi}(s)} = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s\right\}$$

Return

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



State:
- Image ratio – 210, 160, 3
- Intensity range – 256

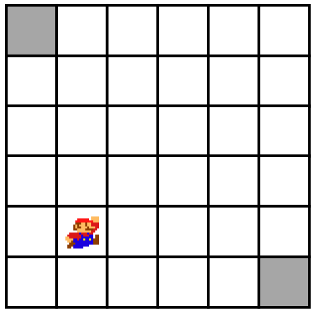
$256^{210x160x3}$ possible states

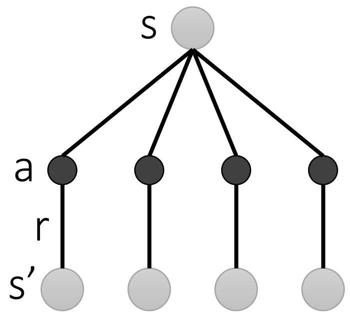
≈ 10^{182063} >> 10^{82} atoms in the universe

Environment Dynamics

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

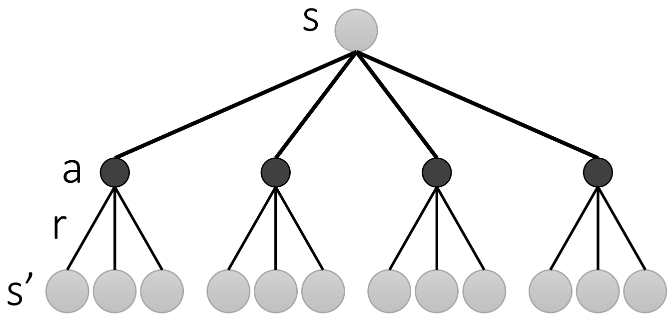
Deterministic Model





Stochastic Model

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



State representation

- Physical world evolves through time from step to step with specific dynamics: $\Pr\left\{s_{t+1}=s', r_{t+1}=r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\right\}$

- Independence of path property: $\Pr\left\{s_{t+1}=s', r_{t+1}=r \mid s_t, a_t\right\}$

Reward

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

Transition Dynamics

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

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

Bellman expectation equation

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

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

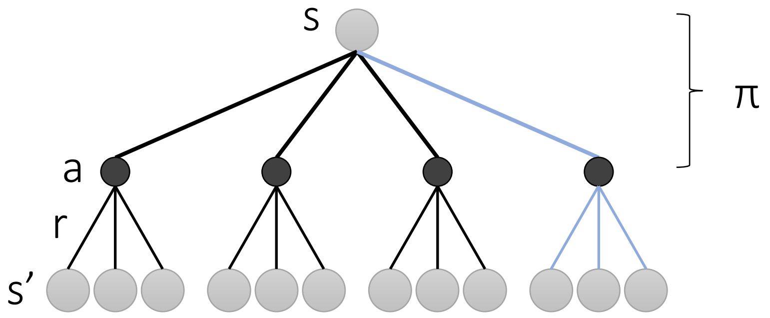
$$= E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s\right\}$$

Recurrency of Return

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

Immediate Reward

Future Reward



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

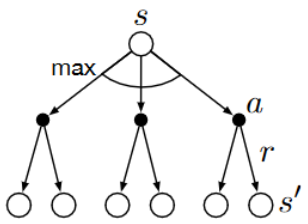
Optimal value functions

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

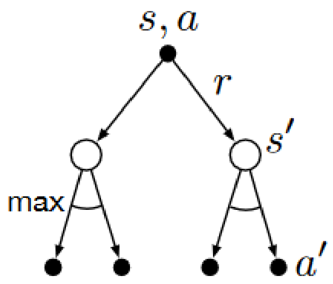
$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

Bellman optimality equation

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



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

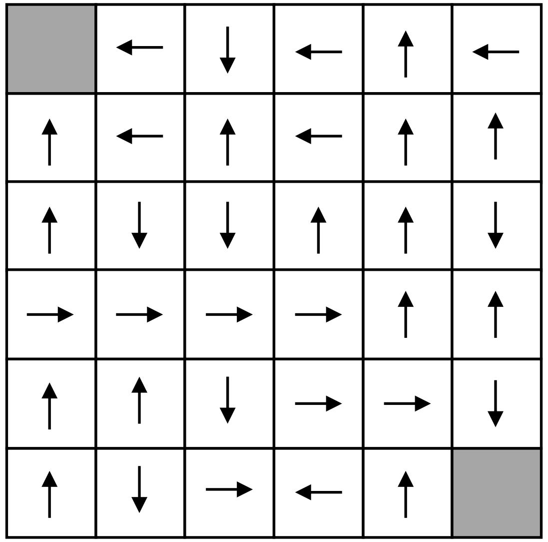


- No knowledge about the environment's dynamics as well as states needed

- Defines behavior/action of an agent at a given state and timestep
- Function that maps from perceived states of the environment to actions taken (stimulus-response rules)
- Depending on the RL-Method policy is derived differently (Value-based, evolutionary)
- **Optimal policy:**

  "Which action to take for the greatest reward"

Random deterministic policy



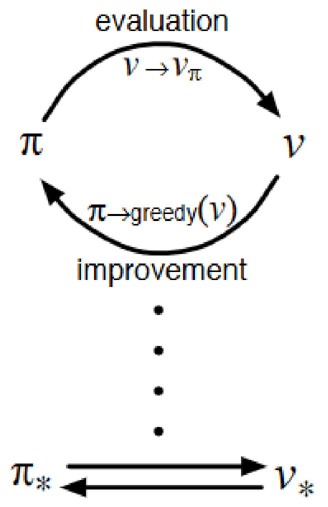
$$\pi_t(s) = a$$

Stochastic policy

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

Generalized Policy Iteration

A universal approach to solve a Markov decision process (RL-Problem)

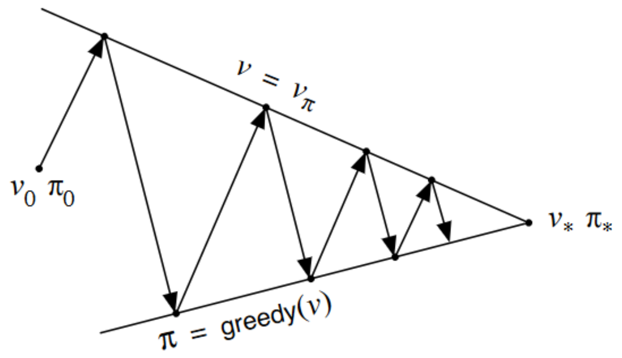


Two simultaneous interacting processes

1. Predicting the value function for a policy – **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

Reinforcement Learning

Model-based	Value-based	Policy-based
• Know/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	• Learn the state- or action-value function of an environment • Decide among actions based on the value of a state / action • Algorithms need to take active exploration into account	• Learn the imminent behavior of the agent, by approximating the optimal policy • Samples from the policy space • Exploration as inherent feature

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

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

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

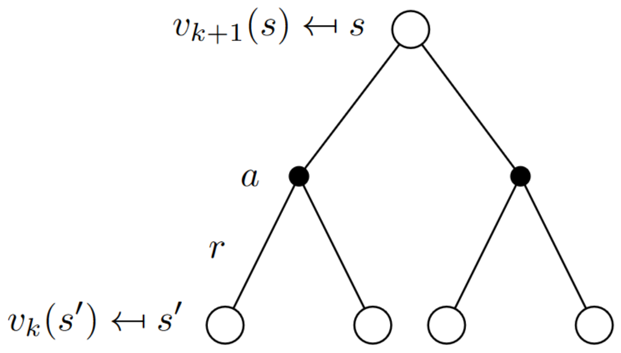
The prediction problem – how to consider whether a policy is good?

Turning the Bellman expectation equation into an iterative update equation

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid s_t = s\right\} \\ &= E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s\right\} \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right] \end{aligned}$$

$$\longrightarrow$$

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



- Updating expectations based on expectations
- Synchronous full backups in a step-by-step approach

$$V_{k+1}(s) = E_\pi \left\{ r_{t+1} + \gamma V_k(s_{t+1}) \,\middle|\, s_t = s \right\}$$

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

$k = 1$

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

The Control problem – how to improve the old policy?

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

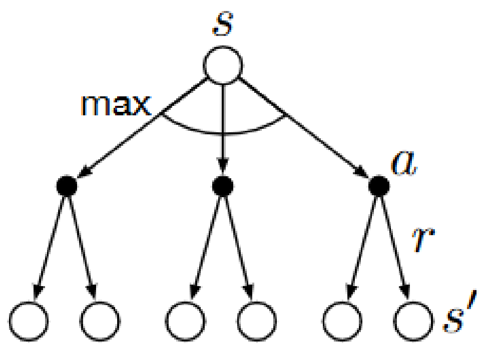
$$\begin{aligned}\pi'(s) &= \arg\max_a Q^{\pi}(s,a)\\ &= \arg\max_a E\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s, a_t = a\}\\ &= \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right],\end{aligned}$$

Policy improvement theorem

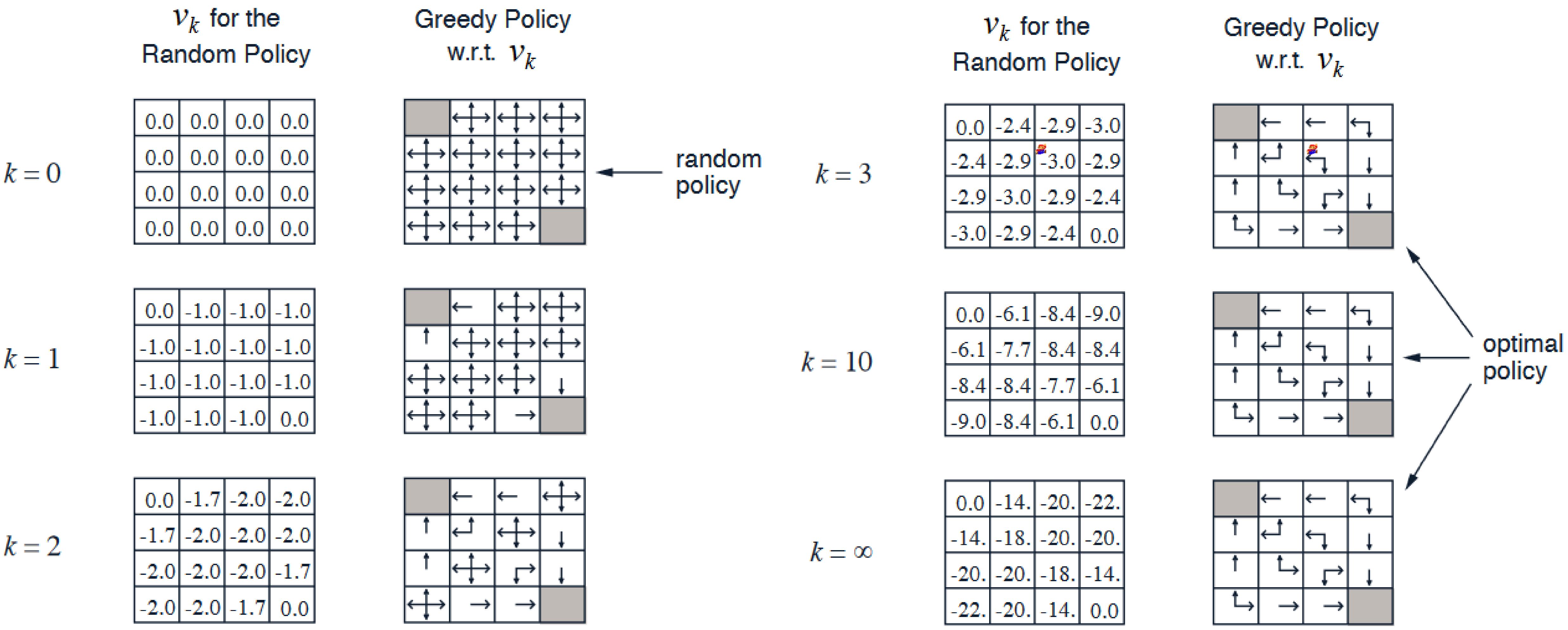
$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s) \quad ,\forall\ s \in S$$

If True, then:

$$V^{\pi'}(s) \geq V^{\pi}(s).$$



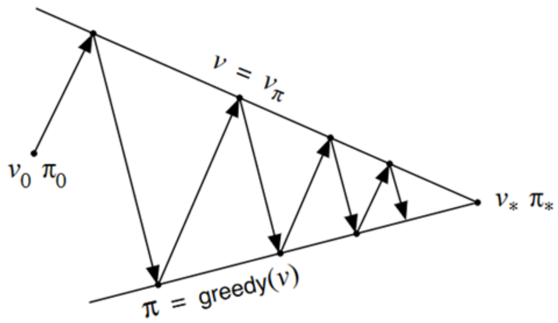
$$\pi'(s) = \arg\max_{a} Q^{\pi}(s, a)$$



- Application of policy evaluation and policy improvement until value and policy function converge to their respective optima

- Finite MDP has finite set of policies, hence convergence is guaranteed in finite steps

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



1. **Initialization**

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

2. **Policy Evaluation**

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

3. **Policy Improvement**

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

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

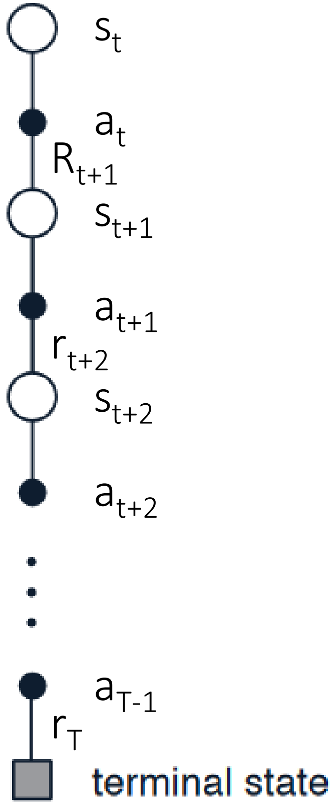
$$\mathcal{R}^a_{ss'} = E\left\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\right\}$$

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

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

Estimate

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



- Only suited for episodic MDPs, as Return needs to be well defined
- Updates are made in an episode-by-episode sense
- Extension of DP where only sample experience is available

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

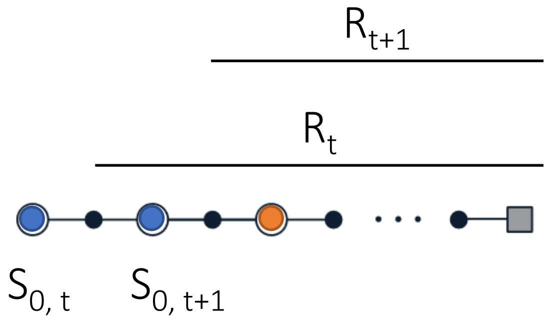
$$V(s_t) = V(s_t) + \alpha(R_t - V(S_t))$$

→ No bootstrapping, estimated Value functions are independent of each other

Problem: no guarantee of exploration

Solution: Exploring starts

When to update the expected return?



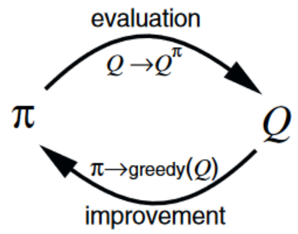
Every visit

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 π
- (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$(Returns(s))$

→ Make policy greedy with respect to the current value function

- Update policy that was used for sampling from the environment



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

$\epsilon > 0$

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

Probability of taking an action under policy π

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

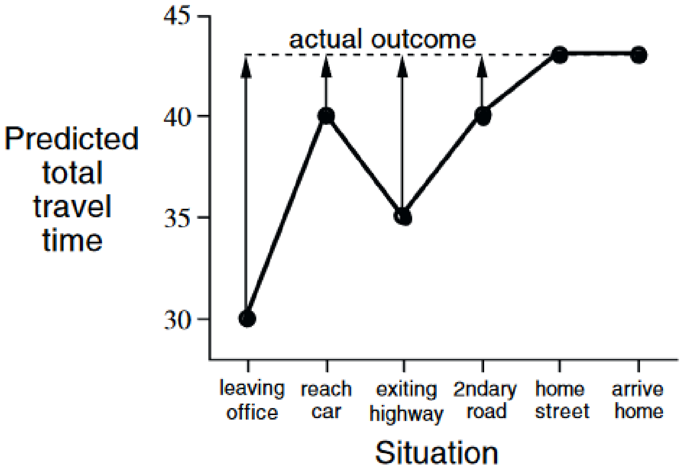
Suboptimal/non greedy action: $$\pi(s,a) = 1 - \varepsilon + \frac{\varepsilon}{A(s)}$$

Bootstrapping

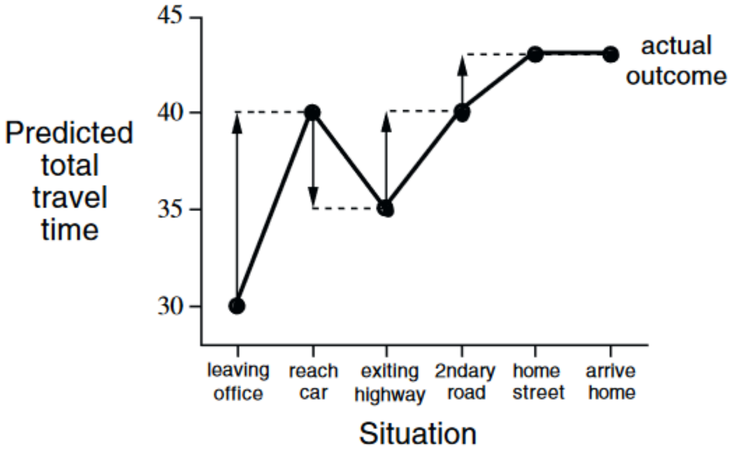
→ update estimated based on other learned estimates (like DP-Methods)

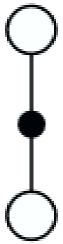
Episode sampling

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



Adjust expectation based on the actual Return





Adjust expectation based on the estimated Return

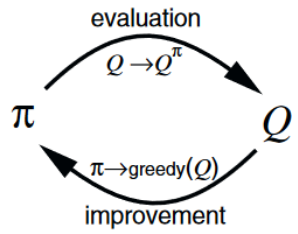
- one step lookahead search

$$\underbrace{V(s_t)}_{\text{new estimate}} \leftarrow V(s_t) + \underbrace{\alpha}_{\text{step size}} \left[\underbrace{r_{t+1} + \gamma V(s_{t+1})}_{\text{Target}} - V(s_t)\right]$$

old estimate

Implementation:
iterative procedure, where Value of next state $V_k(s_{t+1})$ is estimate of previous calculation $V_{k-1}(s_{t+1})$

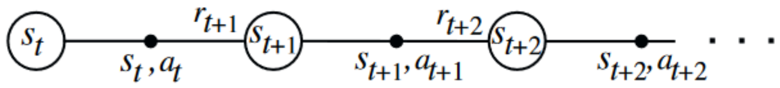
Iterative approach which encapsules GPI idea:



- Apply TD to **action-value function** (policy evaluation)
- Change policy towards greediness with respect to Q^{π} (e.g. e-greedy)

Action-value function updates

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



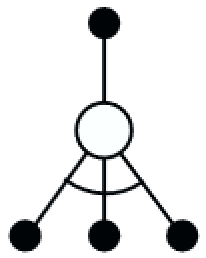
Convergence under the condition that every state- action pair is visited an infinite number of times (exploration)

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

Behavior policy – used to sample from the environment (ε-soft policy)

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

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



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

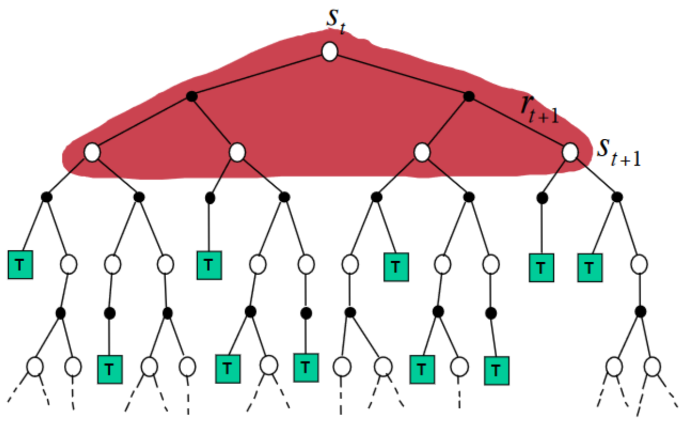
$$\boxed{\pi'(s) = \arg\max_a Q^{\pi}(s, a)}$$

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

Backup diagrams

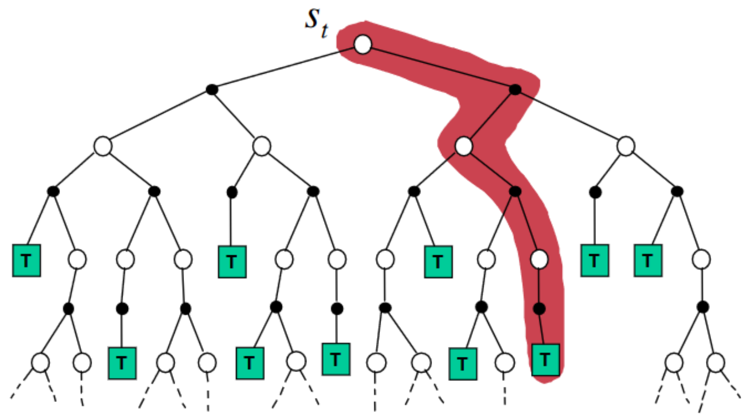
Dynamic programming

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



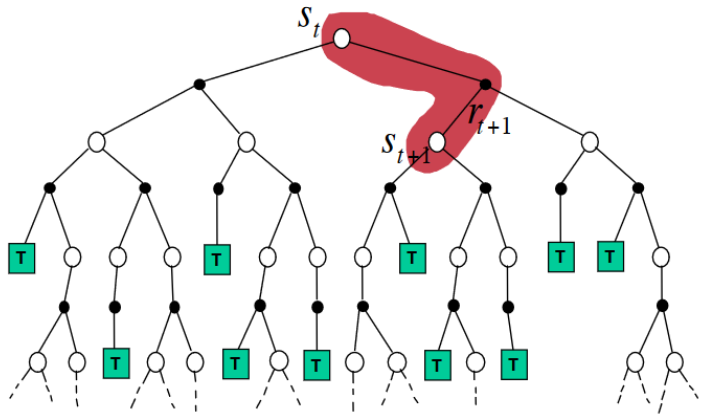
Monte Carlo Methods

$$V(s_t) \leftarrow V(s_t) + \alpha(R_t - V(S_t))$$



Temporal difference (0)

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



Dynamic programming

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

- Bootstraps
- Does not sample

Monte Carlo Methods

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

- Does not Bootstrap
- Samples

Temporal difference (0)

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

- Bootstraps
- Samples

1. Where is the boundary between agent and environment?
2. How does the course of dimensionality effect reinforcement learning?