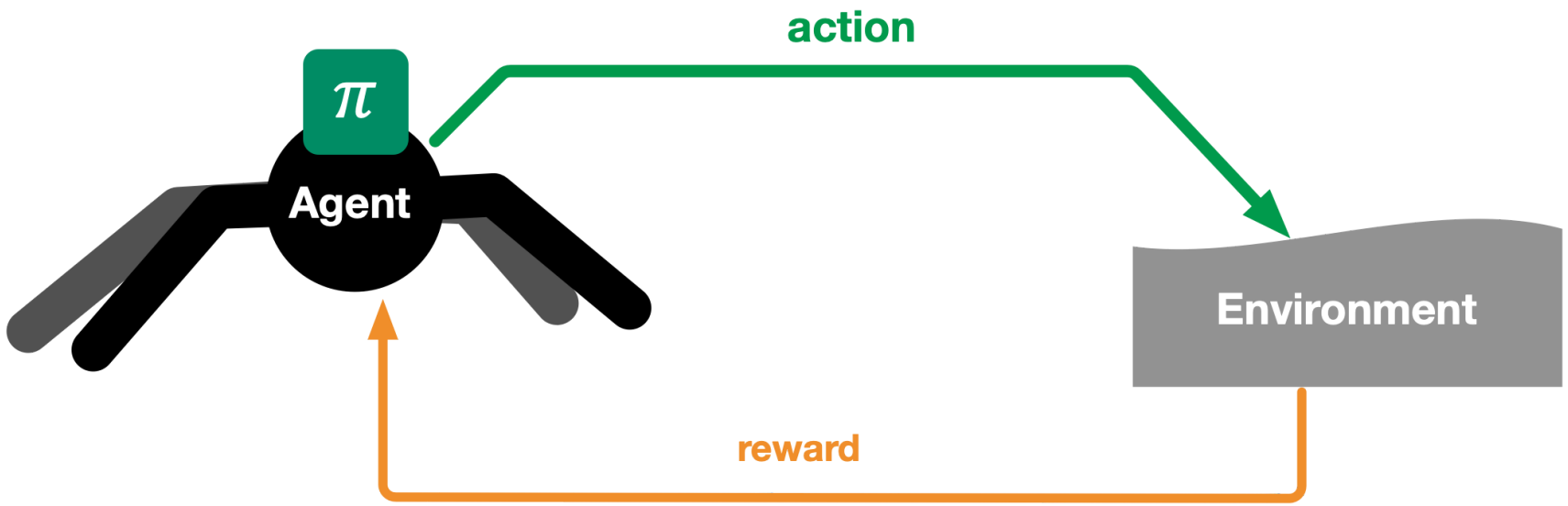
Deep Reinforcement Learning

3 - Markov Decision Process

Prof. Dr. Malte Schilling

Autonomous Intelligent Systems Group

Recap – Decision Making



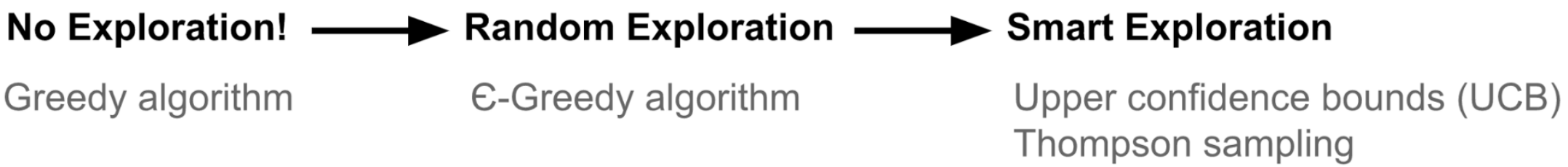
Agent

- A **policy** *is* the agent's behavior – chooses an action.
- **Value function**:
  $Q_t(a) = \mathbb{E}\Big[R_t \mid A_t = a\Big]$

Environment

- Provides reward

Recap – Multi-Armed Bandits



Exploitation-Exploration Tradeoff

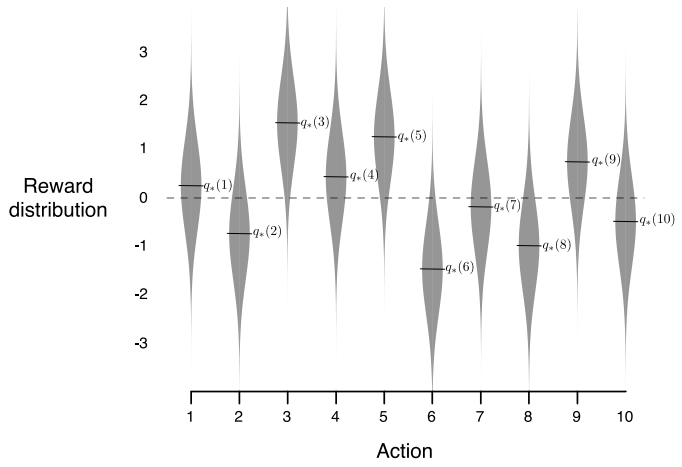
Decision Making: sticking to a good past experience might make you miss out on even better options, but at least you can be confident to get something good.

(Weng 2018b)

Can be realized in different ways and using different forms of distributions.

Normal Distribution

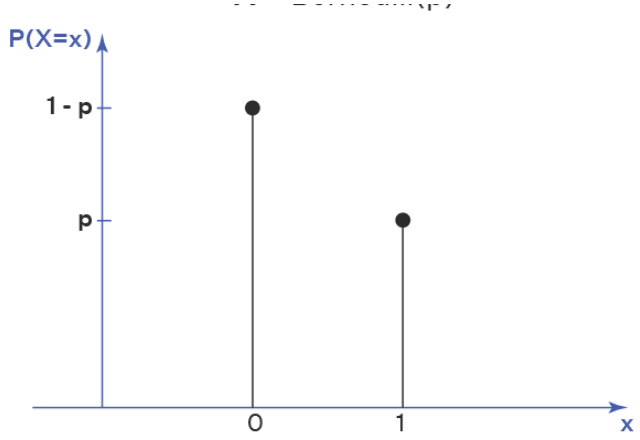
In the simplest case, we assume the reward as following a normal distribution. Such a bandit can be represented through a mean and standard deviation. Rewards are sampled from this distribution.



Bernoulli Distribution

For discrete rewards: a Bernoulli Distribution can be used. The probability describes how often a fixed reward is given (otherwise zero would be returned).

The PD of the expected reward should be described using a beta distribution.



Overview Lecture

- Models for Sequences
- Markov Decision Process
- Value Functions and Bellman Equation
- Overview Approaches in RL
- Optimal Policy and Optimal Value Function

Models of Sequences

Sequential Decision Making

The interaction between agent and environment is a **sequence** of actions and returned observations plus rewards.

Goal of the agent: select actions to maximise total future reward

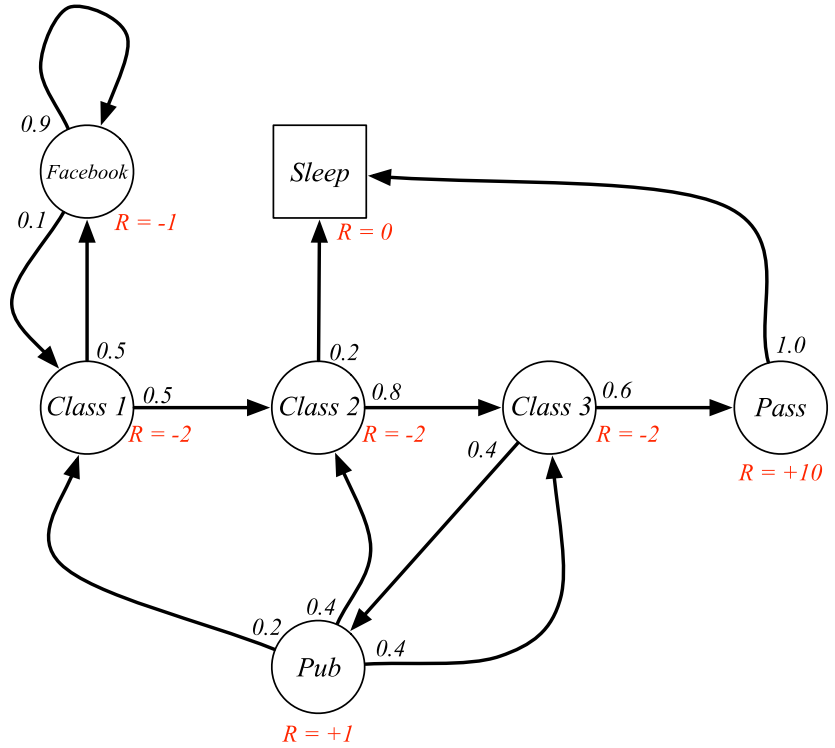
- But actions may have long term consequences and reward may be delayed.
- It may be better to sacrifice immediate reward to gain more long-term reward

An agent's **policy** $\pi(s) = a$ describes which action a an agent selects depending on the current state.

For the stochastic state, a policy is a probability distribution over actions:
$\pi(a|s) = \mathbb{P}_\pi[A = a|S = s]$.

(Silver 2015)

A Markov Reward Process



Before turning to action, we focus on a simpler class of Markov Chains: the Markov Reward Process. We could sample trajectories from it.

(Silver 2015)

Markov Process

A Markov process is a memoryless random process – a sequence of random states $S_1, S_2, \ldots$ with the Markov property.

Markov Property

A state S_t is Markov iff $P(S_{t+1}|S_t) = P(S_{t+1}|S_1, \ldots, S_t)$

States captures all relevant information from the history ("The future is independent of the past given the present").

Markov Process (Markov Chain)

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix: $P_{ss'} = P(S_{t+1} = s'|S_t = s)$

Markov Reward Process

A Markov reward process is a Markov Chain with associated reward values.

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix: $P_{ss'} = P(S_{t+1} = s' | S_t = s)$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1} | S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

Value Function

A **Value Function** $V(s)$ measures the estimated goodness of a state, i.e. how rewarding a state is by a prediction of the cumulative future reward.

Return

cumulative future reward is a total sum of discounted rewards going forward, starting from time t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

As future rewards are usually more uncertain, they are weighted less through the **discount factor** $\gamma \in [0, 1]$.

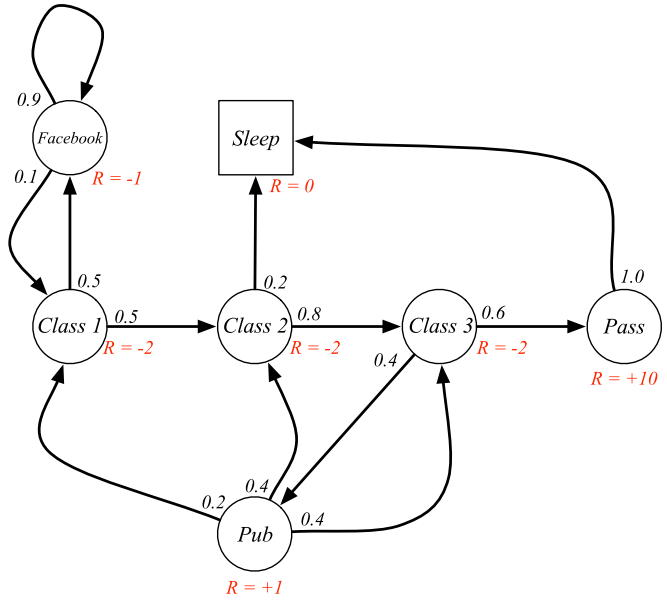
(Weng 2018a)

Discouting of reward

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- Animal/human behaviour shows preference for immediate reward
- It can be possible to use undiscounted Markov reward processes for terminating sequences.

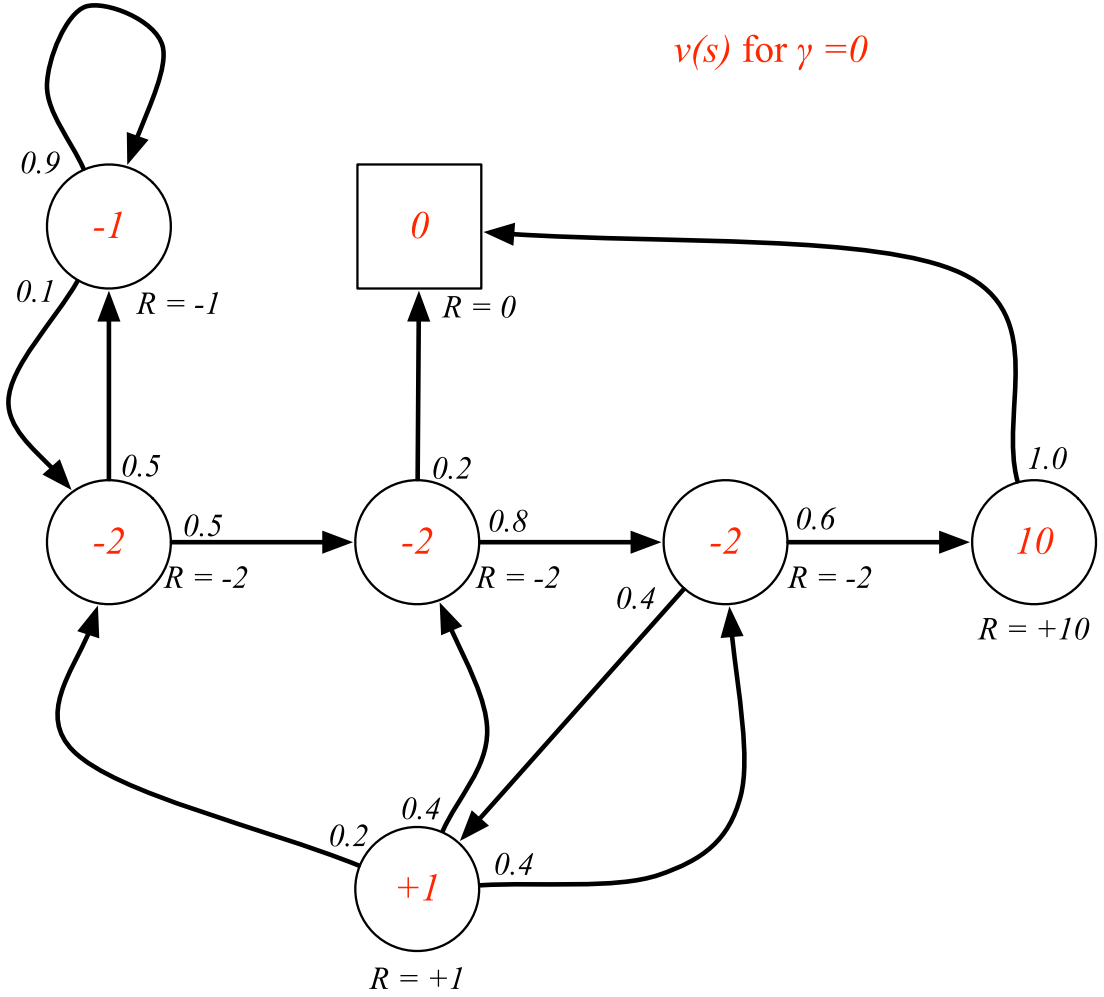
(Silver 2015)

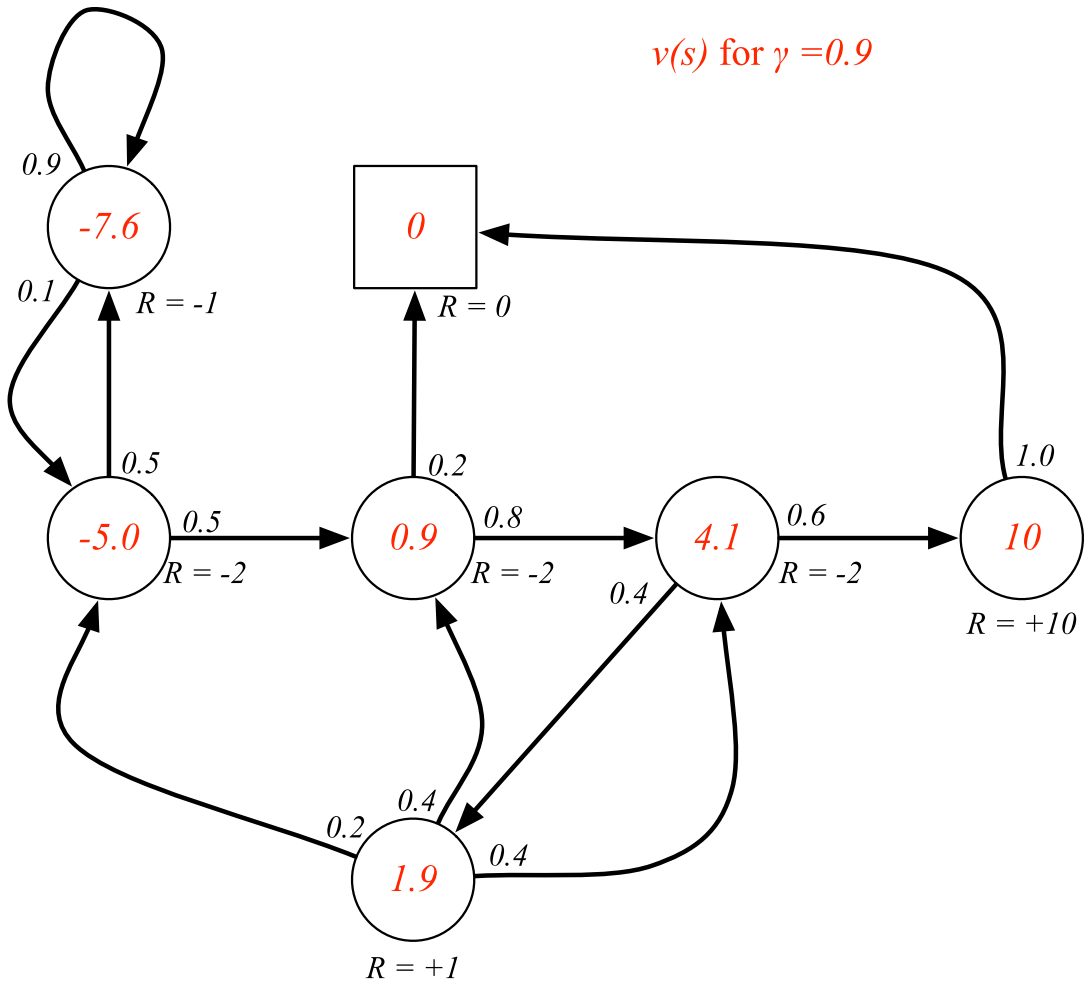


C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8} = -2.25$
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} = -3.125$
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \ldots = -3.41$
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \ldots = -3.20$
FB FB FB C1 C2 C3 Pub C2 Sleep	

(Silver 2015)

Convergence of the value function





(Silver 2015)

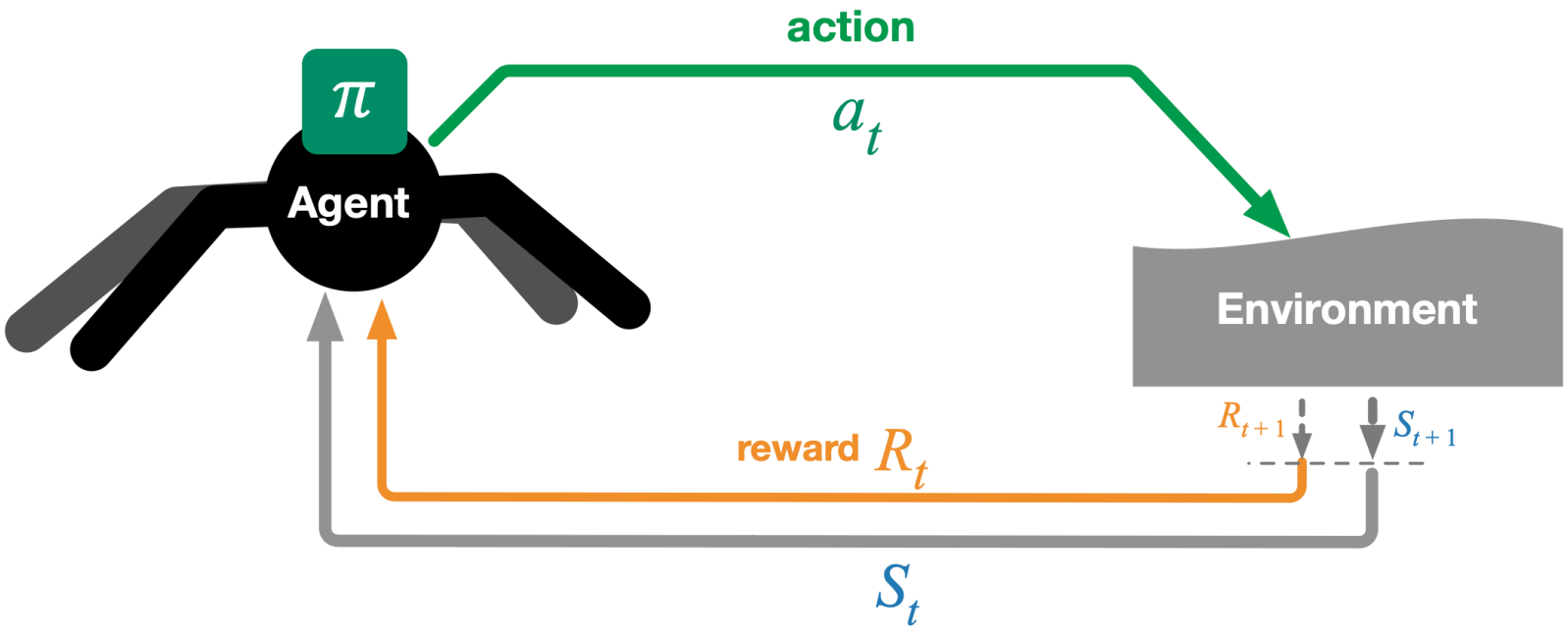
Bellman Equation

The value function can be decomposed into two parts:

- immediate reward R_{t+1}
- discounted reward from this point forward for the next reached state – which is estimated by the state-value function: $v(S_{t+1})$

$$
\begin{aligned}
V(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \ldots) | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s]
\end{aligned}
$$

RL Cycle – Sequential Decision Making



Agent

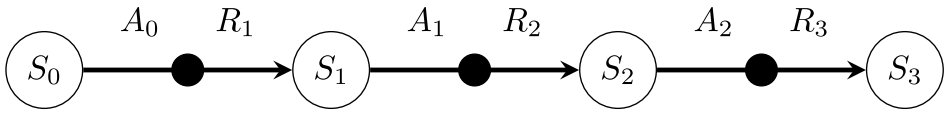
- Policy: choose an action.
- Value-Function: Estimate of achievable return from a state.

Environment

- Reward: Describing goal
- **State**: observation for agent

Markov Decision Process

Markov decision processes formally describe an environment for reinforcement learning.



Requirements:

The environment is fully observable (current state completely characterises the process).

Almost all RL problems can be formalised as MDPs

- Optimal control primarily deals with continuous MDPs
- Partially observable problems can be converted into MDPs
- Bandits are MDPs with one state

(Silver 2015)

Markov Decision Process

MDP Definition (Sutton und Barto 2018)

A Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, p, \gamma)$, consisting of

- a set of states $\mathcal{S}$
- a set of actions $\mathcal{A}$
- a joint probability $p(r, s'|s, a)$ describing the dynamics of the environment as
  - transition probabilities for switching states

$$p(s'|s, a) = \sum_r p(r, s'|s, a)$$

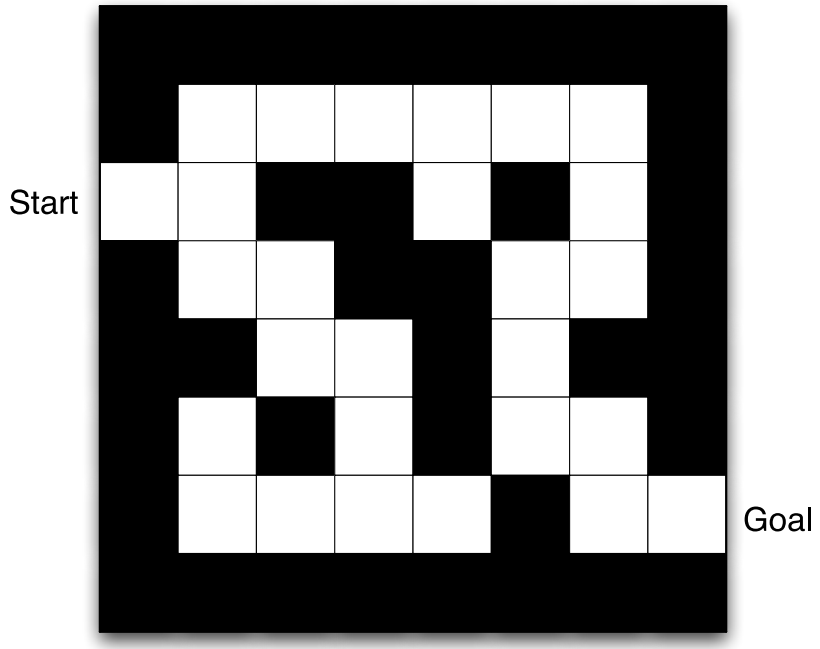
  - and expected reward

$$\mathbb{E}(R|s, a) = \sum_r r \sum_{s'} p(r, s'|s, a)$$

- the discount factor $\gamma \in [0, 1]$

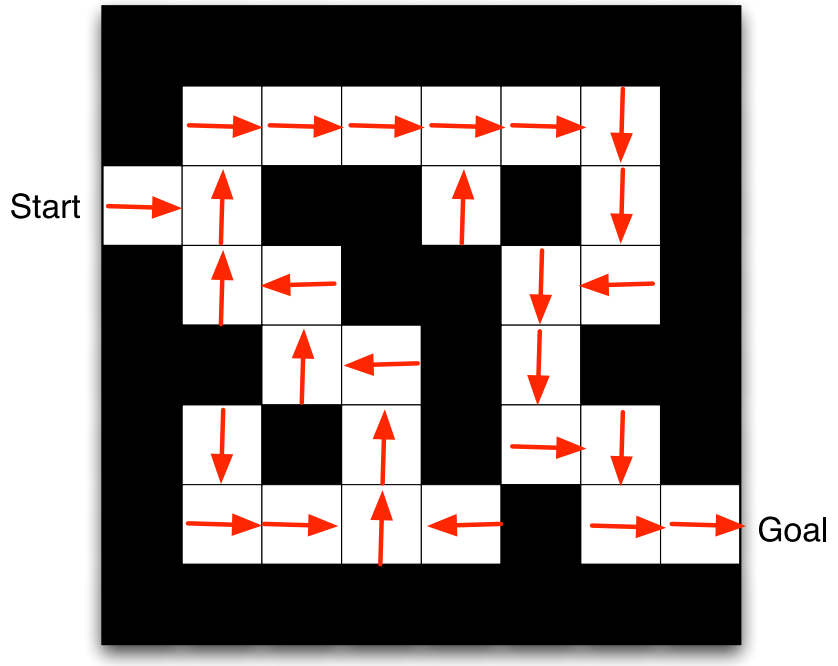
Maze Example: Policy

Task



- Reward of -1 per time step in maze
- Actions are move N, S, W, E
- State is location

Policy Representation

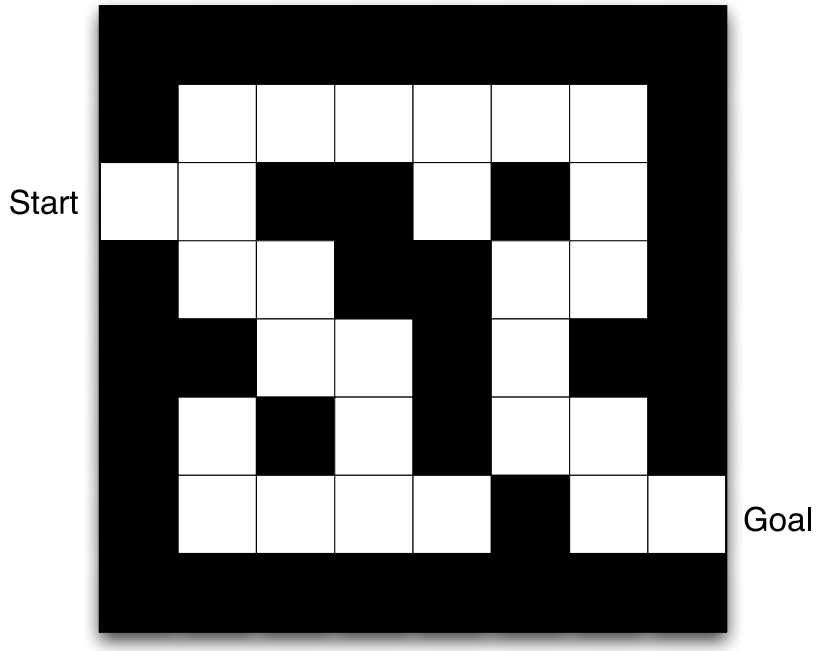


Arrows represent policy $\pi(s)$ for all the states.

(Sutton und Barto 2018)

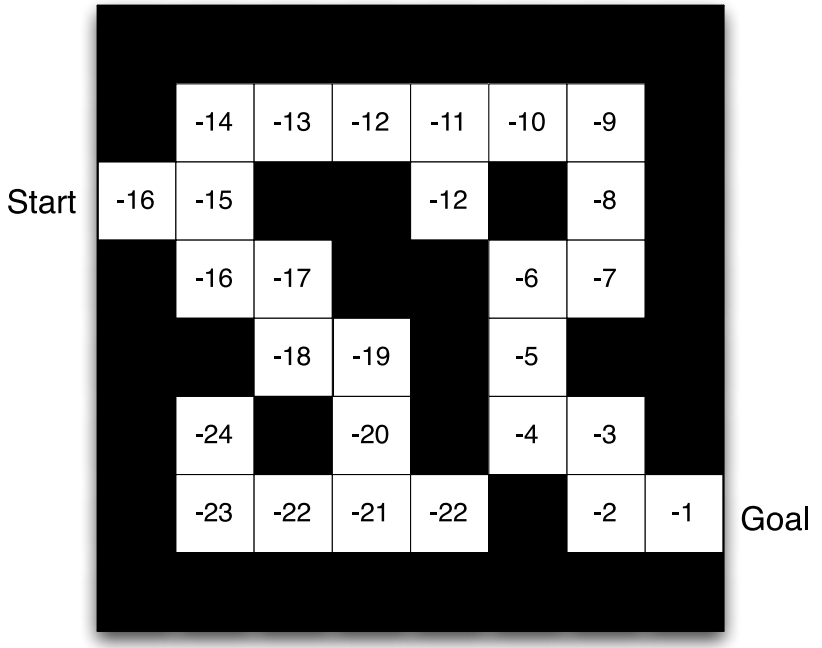
Maze Example: Value Function

Task



- Reward of -1 per time step in maze
- Actions are move N, S, W, E
- State is location

State Value



Shown are values $v_\pi(s)$ for the different states.

(Sutton und Barto 2018)

Partially Observable Environments

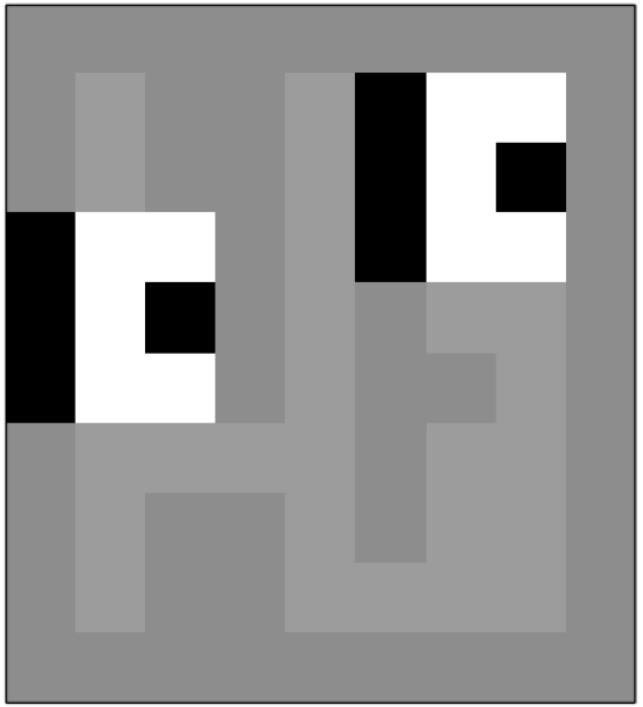
We will focus on fully observable environments: The agent state contains all necessary information required for making an informed decision.

Partial Observability

The agent only indirectly experiences the environment, e.g. no exact position information, but only relying on a camera.

Importantly, he might not be able to distinguish states.

The agent, therefore, must construct its own internal state representation (which could, e.g., include information on history).



MDP Alternative Definition

A Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, consisting of

- a set of states $\mathcal{S}$
- a set of actions $\mathcal{A}$
- $p(s'|s, a)$ is the probability of transitioning to s', given a state s and action a
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the expected reward
- the discount factor $\gamma \in [0, 1]$

(Hasselt und Borsa 2021)

Reminder – Markov Property

A state S_t is Markov iff

$$p(S_{t+1}|S_t) = p(S_{t+1}|S_1, \ldots, S_t)$$

States captures all relevant information from the history ("The future is independent of the past given the present").

In a Markov Decision Process all states are assumed to have the Markov Property.

Task – Understanding the Markov Property?

In a Markov Decision Process all states are assumed to have the Markov Property.
Which of the following statements are therefore true for an MDP?

$$p(S_{t+1} = s' | S_t = s, A_t = a) = p(S_{t+1} = s' | S_1, \ldots, S_{t-1}, A_1, \ldots, A_{t-1}, S_t = s)$$

$$p(S_{t+1} = s' | S_t = s, A_t = a) = p(S_{t+1} = s' | S_1, \ldots, S_{t-1}, S_t = s, A_t = a)$$

$$p(S_{t+1} = s' | S_t = s, A_t = a) = p(S_{t+1} = s' | S_1, \ldots, S_{t-1}, S_t = s)$$

$$p(R_{t+1} = r, S_{t+1} = s' | S_t = s) = p(R_{t+1} = r, S_{t+1} = s' | S_1, \ldots, S_{t-1}, S_t = s)$$

(Hasselt und Borsa 2021)

Selection of Action

Policies (def.)

A policy π is a distribution over actions given states,

$$\pi(a|s) = p(A_t = a|S_t = s)$$

- A policy fully defines the behaviour of an agent.
- MDP policies depend only on the current state (but not the history), which means: Policies are stationary (time-independent)

(Silver 2015)

Tabular Approaches in RL

A policy can be a deterministic mapping from states to actions.

Important: policies depend only on the state, which defines all the things relevant for action selection, and not on other things like time.

Action Mapping

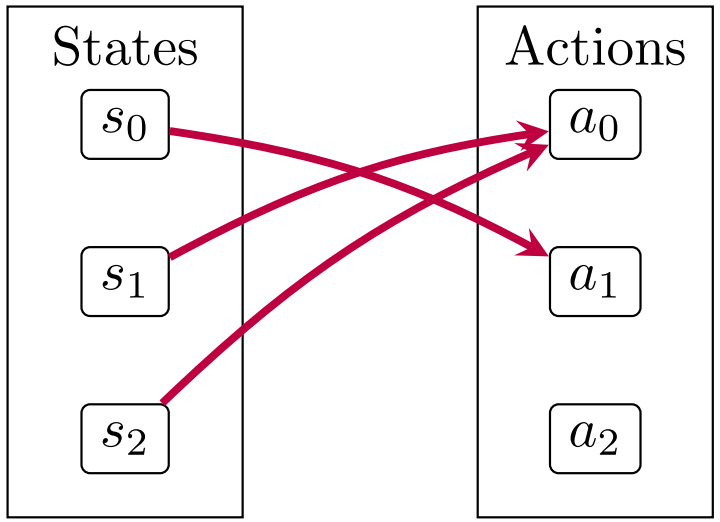


Table Representation

A deterministic policy can also be represented as a table: Each row describes the selected action.

State	Action
s_0	a_1
s_1	a_0
s_2	a_0

State-Value Function in an MDP

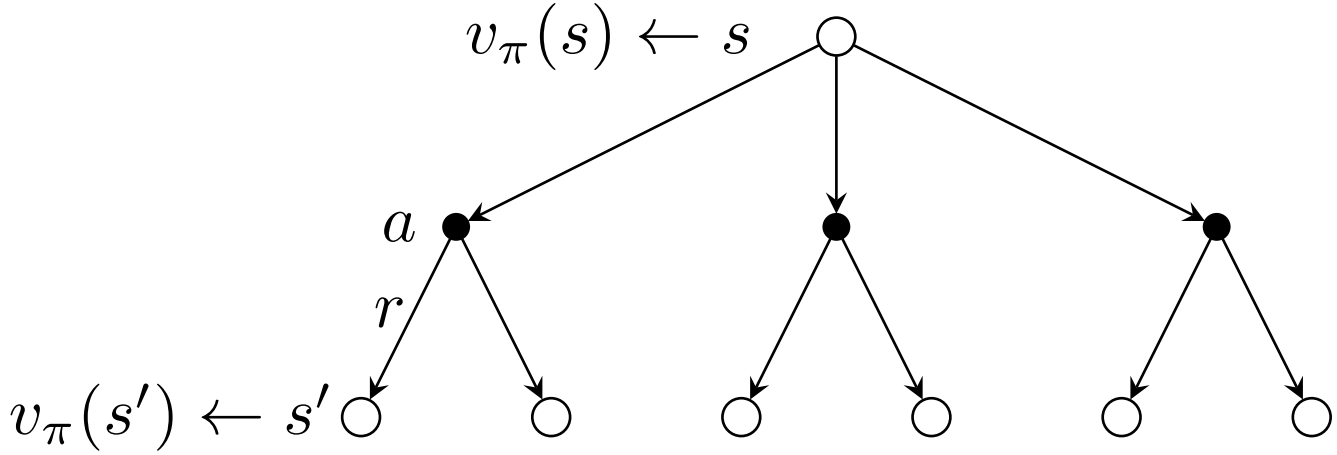
The **state-value function** of a state s is the expected return if we are in this state at time $t, S_t = s$:

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

- Used to evaluate the goodness/badness of states,
- and therefore can be used to select between actions.

(Weng 2018a)

Bellman Expectation for V_π



Recursive Formulation for State-Value Function

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_r \sum_{s' \in \mathcal{S}} p(r, s'|s, a) \Big(r + \gamma v_\pi(s') \Big)$$

With reward independent of s':

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \Big(R^a + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_\pi(s') \Big)$$

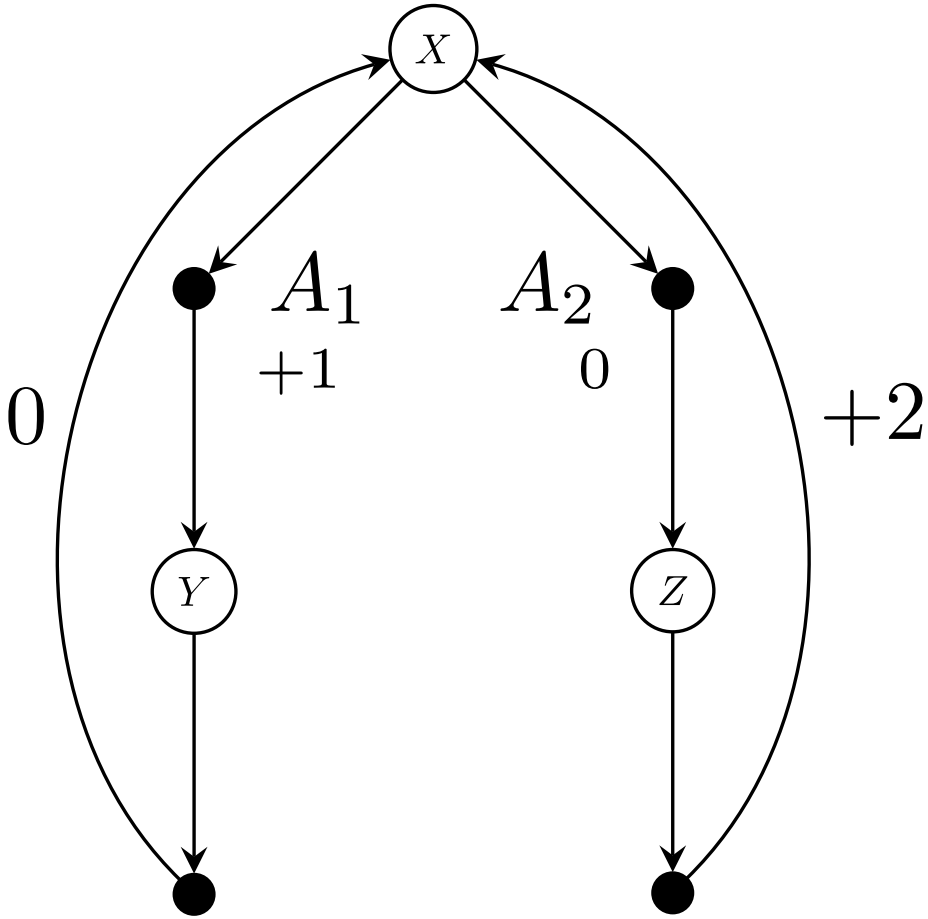
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(r_s^a + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_\pi(s') \right)$$

Task – Iteratively Calculating the Value Function

Given is the MDP on the right with two possible actions A_1 or A_2 for state X.

These lead deterministically to following staes Y and Z for which only a single action is available.

Calculate the value function for X for two different policies: either prefering A_1 or A_2. How does the value function depend on selection of γ?



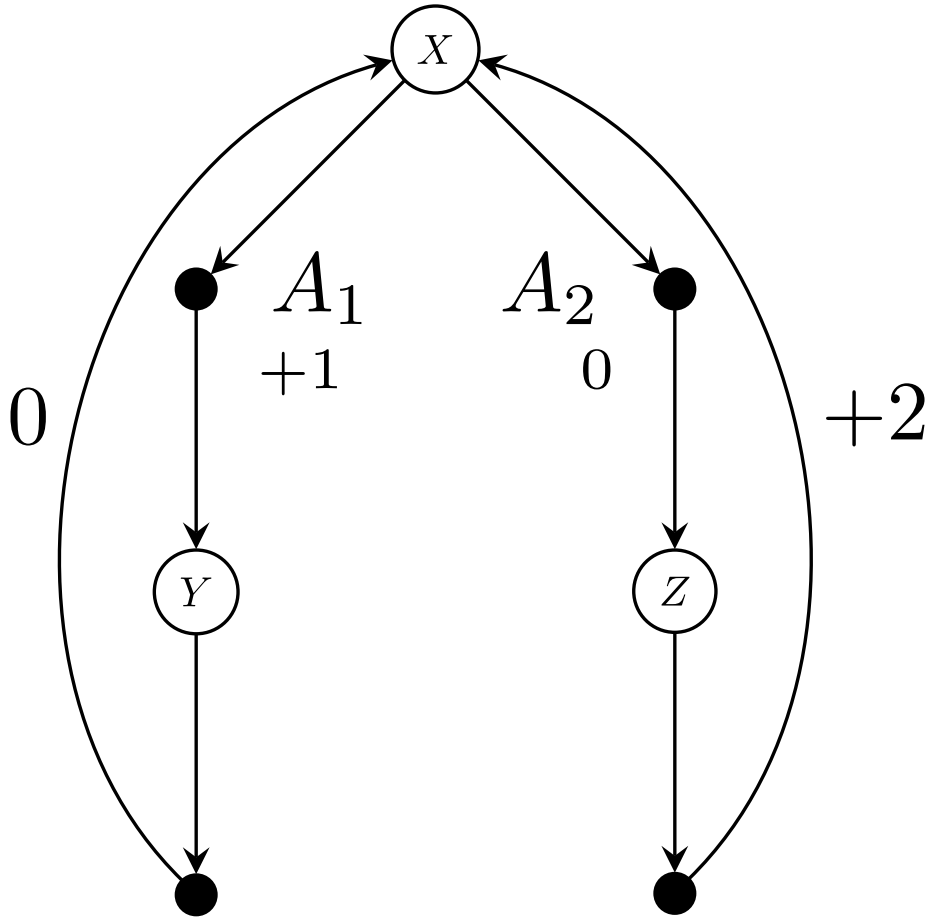
Task – Calculation of Value Function

Focus on Immediate Reward

For $\gamma = 0$:

The value $V(X)$ under $\pi_1(X) = A_1$ is $+1$.

While the value under $\pi_2(X) = A_2$ is 0.



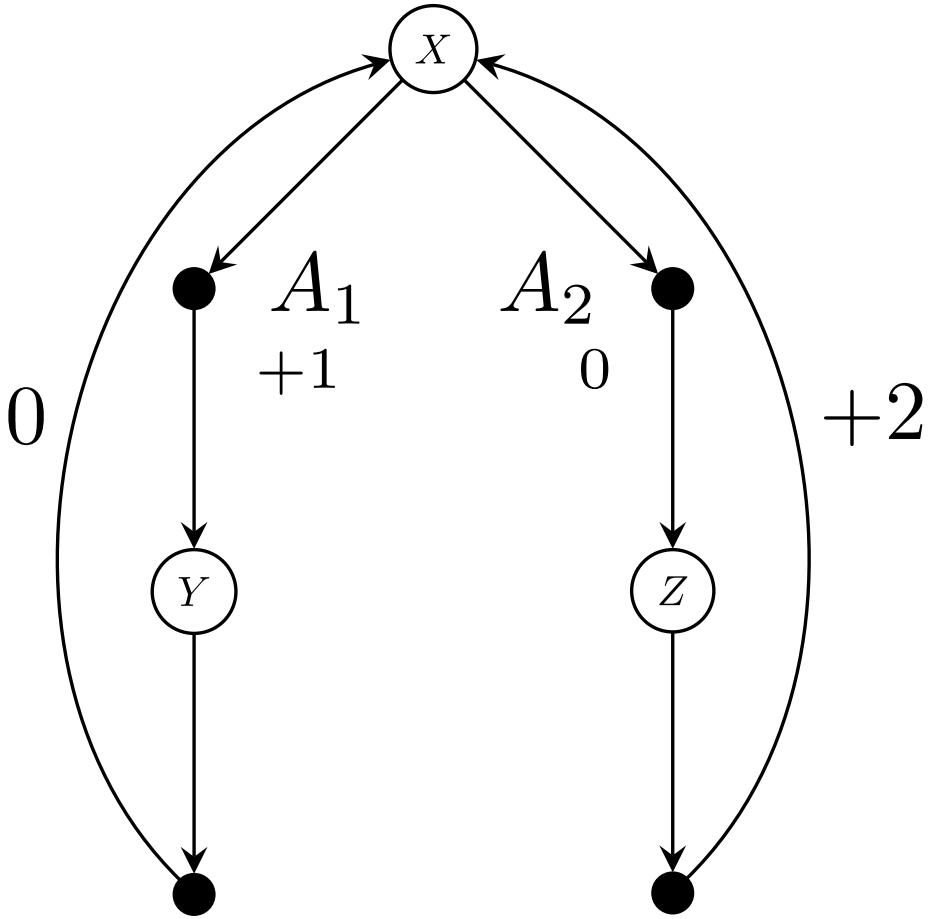
Task – Calculation of Value Function

Discounting Future Returns

For $\gamma = 0.9$:

$$v_{\pi_1}(X) = 1 + 0.9 * 0 + (0.9)^2 * 1 + \ldots$$
$$= \sum_{k=0}^{\infty} 0.9^{2k} = \frac{1}{1 - 0.9^2} \approx 5.3$$

$$v_{\pi_2}(X) = 0 + 0.9 * 2 + (0.9)^2 * 0 + \ldots$$
$$= \sum_{k=0}^{\infty} (0.9)^{2k+1} * 2 = \frac{0.9}{1 - 0.9^2} * 2 \approx 9.5$$



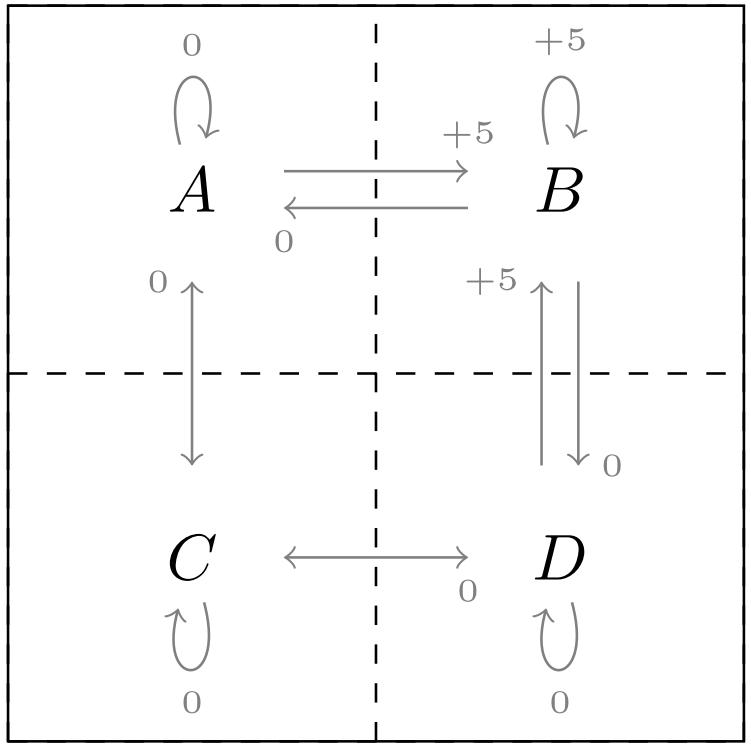
States: four states, labeled A, B, C, and D on a grid.

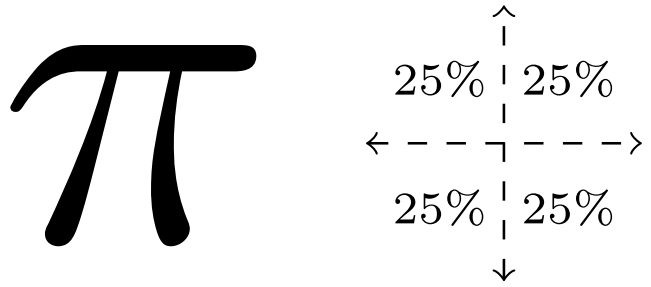
Action space: moving up, down, left, and right. Actions which would move off the grid, keep the agent in place.

Reward: everywhere 0 except for landing in state B, where the agent gets a reward of $+5$ (also when staying in B).

Policy: uniform random (probab. 0.25).

$\gamma = 0.7$ for continuing task.



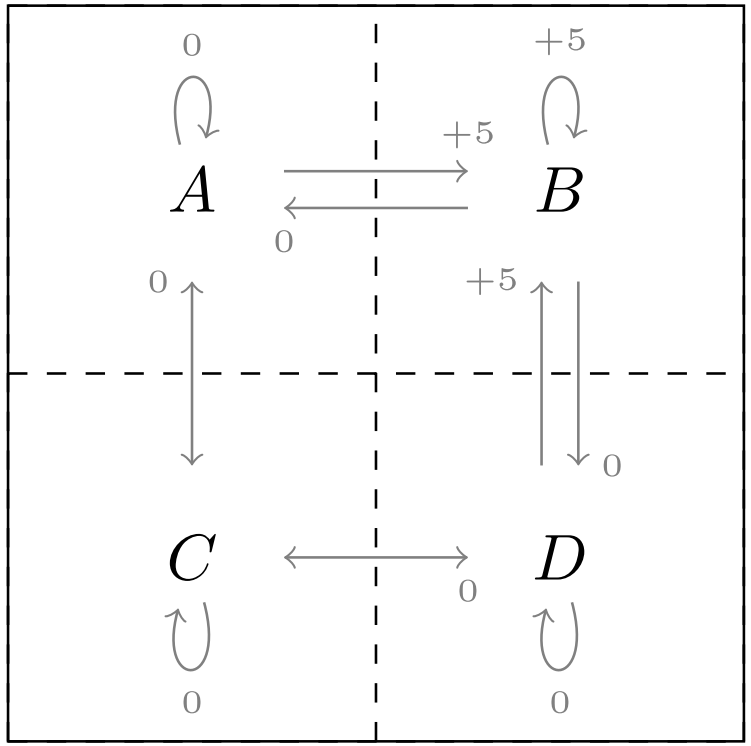


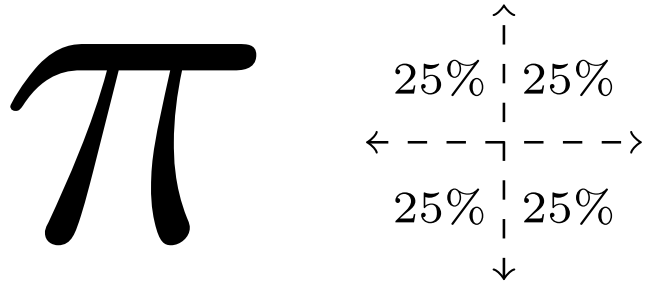
(Santucci 2019)

$$v_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} P\left(s', r|s, a\right) \left[r + \gamma v_\pi(s')\right]$$

Simple MDP: Deterministic state update and reward, therefore, for A:

$$\begin{aligned} v_\pi(A) &= \sum_a \pi(a|s) \left(r + 0.7 v_\pi(s')\right) \\ &= \frac{1}{4}(5 + 0.7 v_\pi(B)) + \frac{1}{4}(0.7 v_\pi(C)) + \frac{1}{2} 0.7 v_\pi(A) \end{aligned}$$

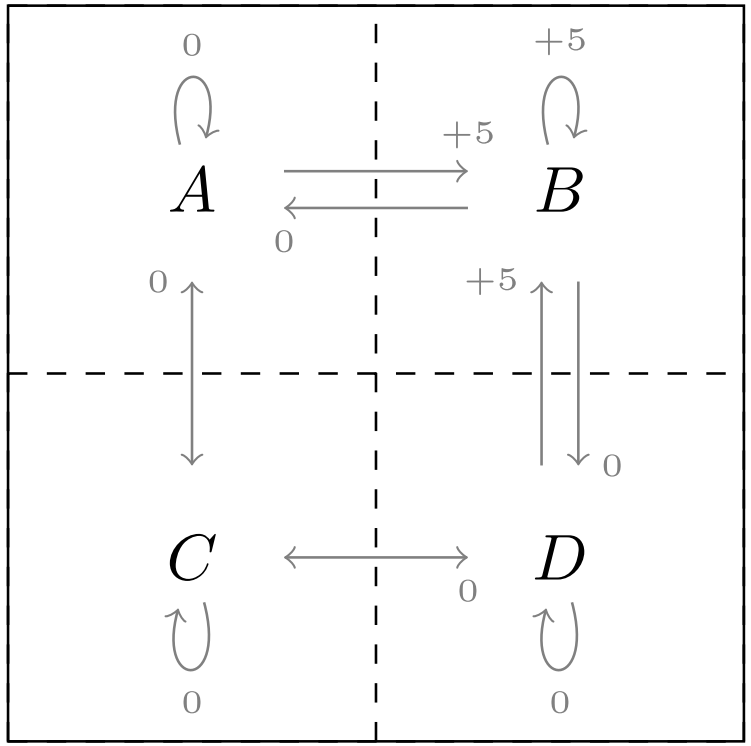


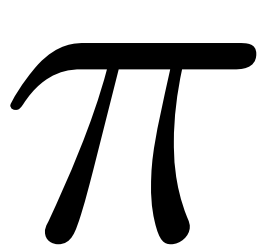


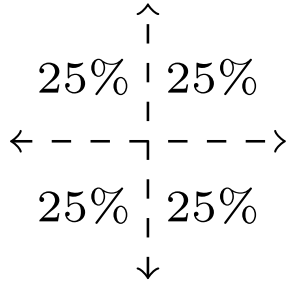
$$v_\pi(A) = \frac{1}{4}(5 + 0.7v_\pi(B)) + \frac{1}{4}(0.7v_\pi(C)) + \frac{1}{2}0.7v_\pi(A)$$
$$v_\pi(B) = \frac{1}{2}(5 + 0.7v_\pi(B)) + \frac{1}{4}0.7v_\pi(A) + \frac{1}{4}0.7v_\pi(D)$$
$$v_\pi(C) = \frac{1}{4}0.7v_\pi(A) + \frac{1}{4}0.7v_\pi(D) + \frac{1}{2}0.7v_\pi(C)$$
$$v_\pi(D) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7v_\pi(C) + \frac{1}{2}0.7v_\pi(D).$$

Iterative solution (or solving lin. equ.)

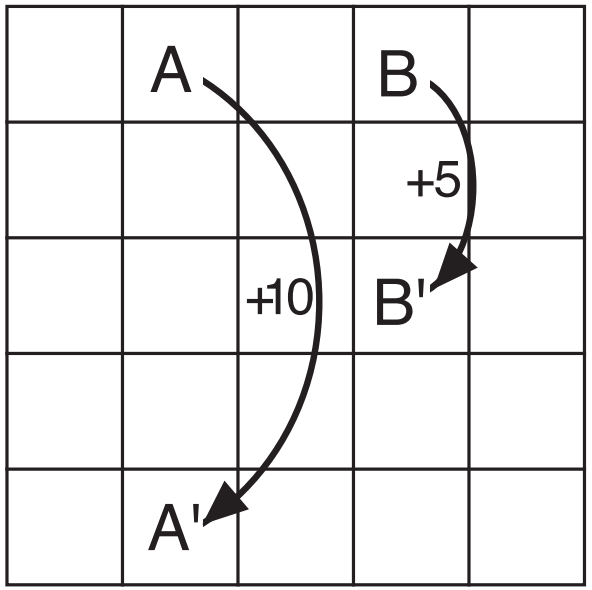
$$v_\pi(A) = 4.2, \quad v_\pi(B) = 6.1, \quad v_\pi(C) = 2.2, \quad v_\pi(D) = 4.2.$$

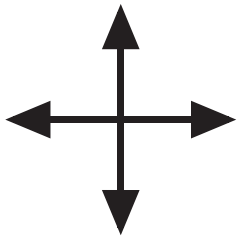






A Gridworld: Value-function for a Random Policy





Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Each location is a state.

Actions: North, West, South, East

Reward: -1 when trying to move out of the grid, 0 otherwise

For state A and B: all actions lead to A' and a reward of $+10$ (respectively B', $+5$).

(Sutton und Barto 2018)

Q-Function

The **action-value function** ("Q-value") of a state-action pair is defined as:

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

When following a target policy π, we can integrate over the probility distribution of possible actions which again leads to the state-value function:

$$V_\pi(s) = \sum_{a \in \mathcal{A}} Q_\pi(s, a) \pi(a|s)$$

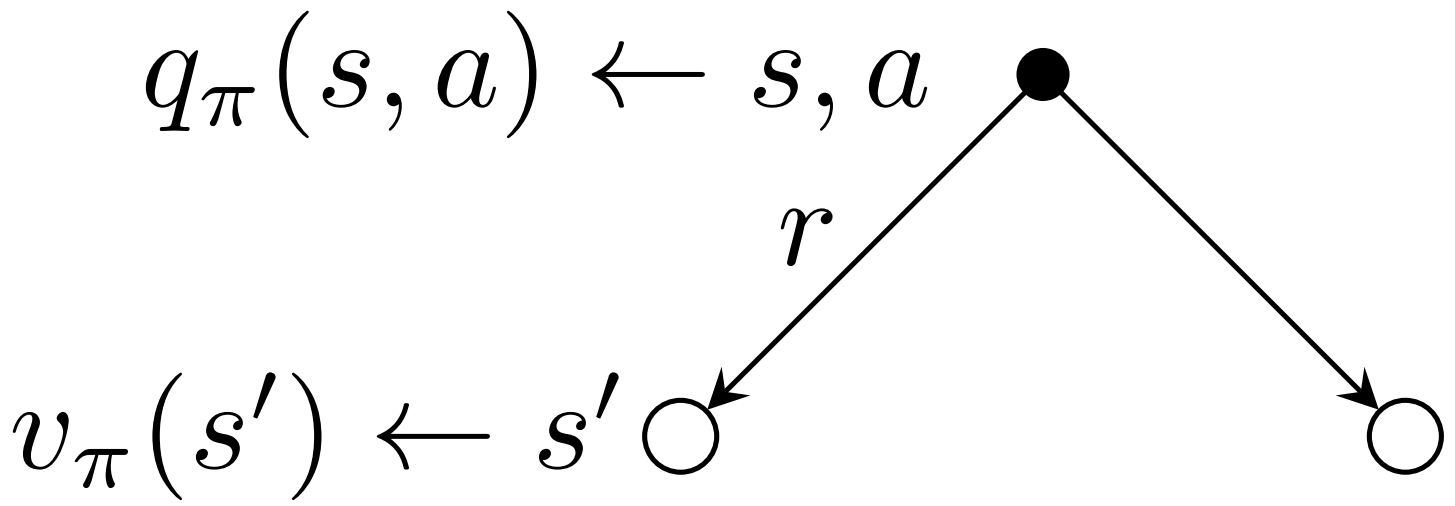
Advantage Function

The difference between action-value and state-value is the **action advantage function**

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$$

(Weng 2018a)

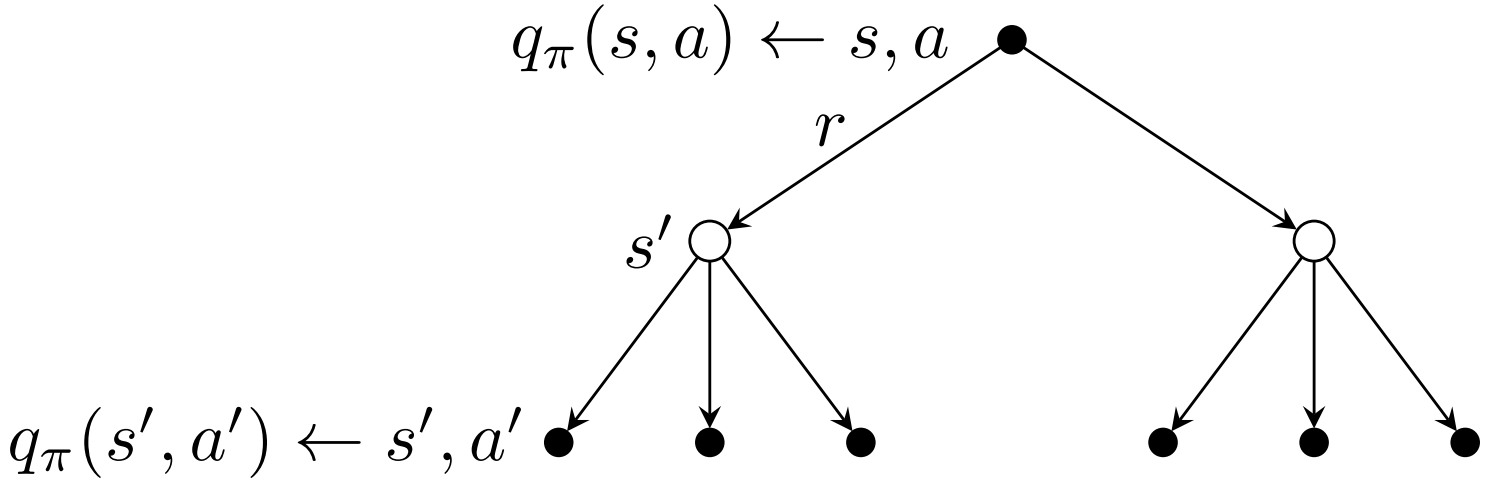
Bellman Expectation for Q_π



Action-Value Function using Bellman Expectation

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_\pi(s')$$

Bellman Expectation for Q_π (2)



Action-Value Function using Bellman Expectation

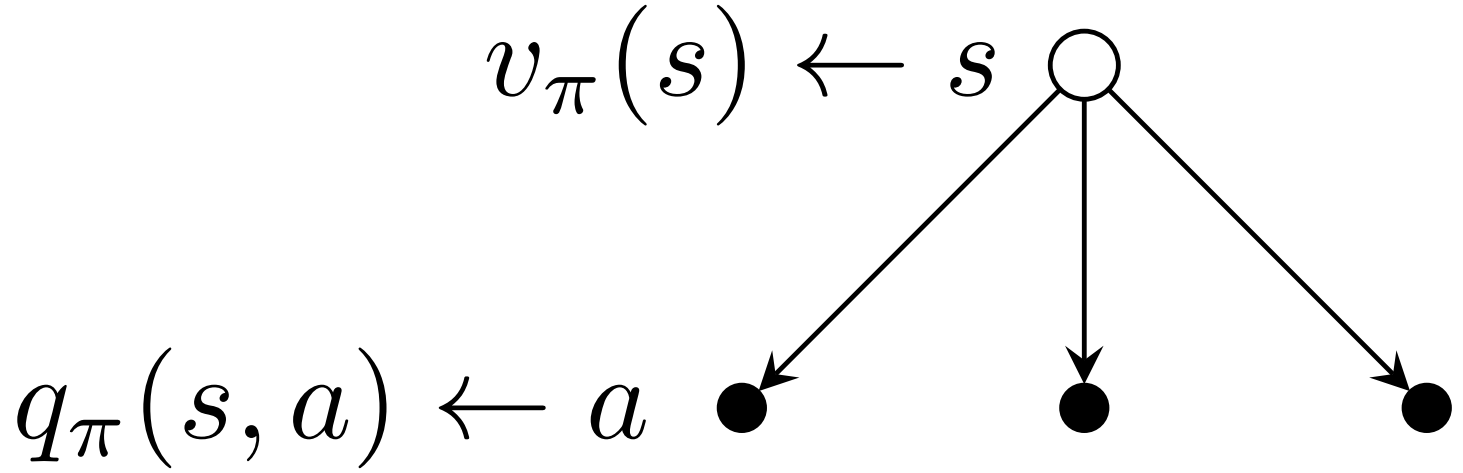
$$q_\pi(s,a) = \sum_r \sum_{s' \in \mathcal{S}} p(r,s'|s,a)\left(r + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')\right)$$

With reward independent of s':

$$q_\pi(s,a) = R_s^a + \gamma \sum p(s'|s,a) \sum \pi(a'|s') q_\pi(s',a')$$

$$\sum_{s' \in \mathcal{S}} \sum_{a' \in \mathcal{A}}$$

Bellman Expectation for V_π



Recursive Formulation for State-Value Function

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

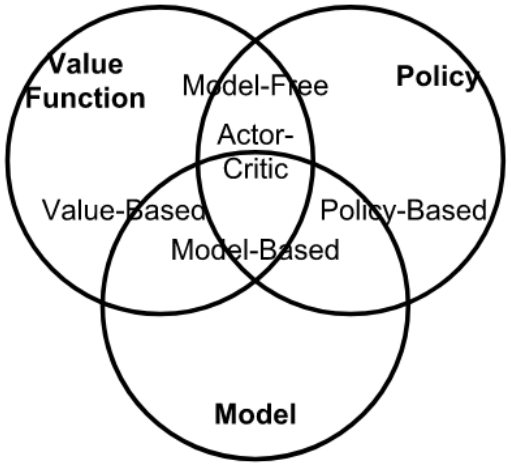
Important: A value function depends on a given policy which the agent follows.

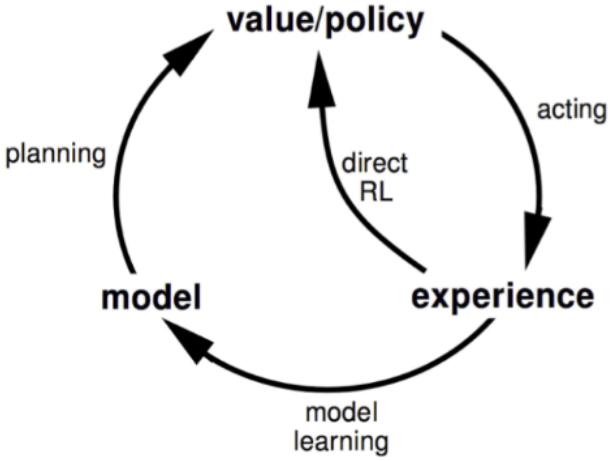
Approaches to Reinforcement Learning

Model of the Environment

The reaction of the environment to certain actions can be represented by a **model** which the agent may or may not know.

The model defines the reward function and transition probabilities.



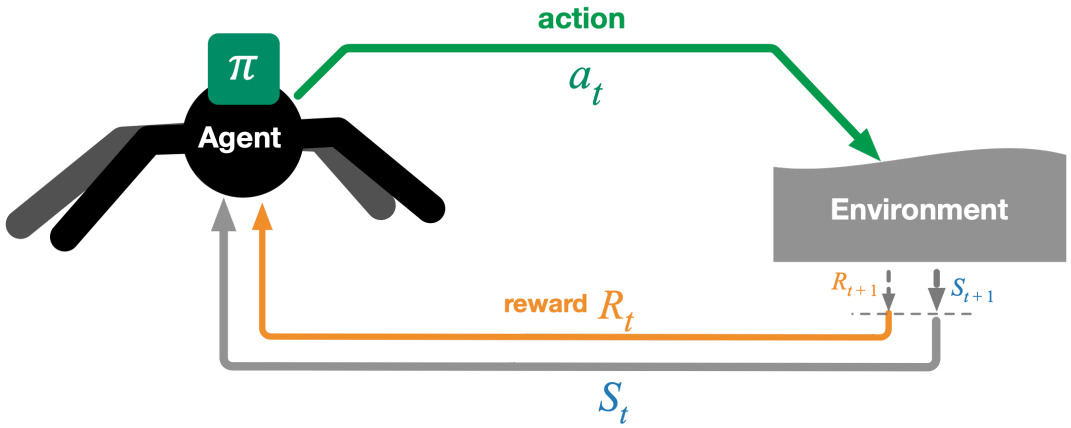


- Model-based: Rely on the model of the environment; either the model is known or the algorithm learns it explicitly. Use planning on learned or given model.

- Model-free: No dependency on the model during learning. Learning with imperfect

(Weng 2018a)

RL Cycle – Sequential Decision Making



Agent

- Policy: choose an action *depending on current state*.
- Value-Function: Estimate of achievable return from a state (following π).
- **Model**: A *predictor* of the environment.

Environment

- Reward: Describing goal
- State: observation for agent

Model

A model allows to predict how the environment will react (as a probability distribution).

- $\mathcal{P}$ predicts the subsequent state:

$$\mathcal{P}^a_{ss'} \approx p(S_{t+1} = s' | S_t = s, A_t = a)$$

- $\mathcal{R}$ predicts the next reward:

$$\mathcal{R}^a_s = \mathbb{E}(R_{t+1} | S_t = s, A_t = a)$$

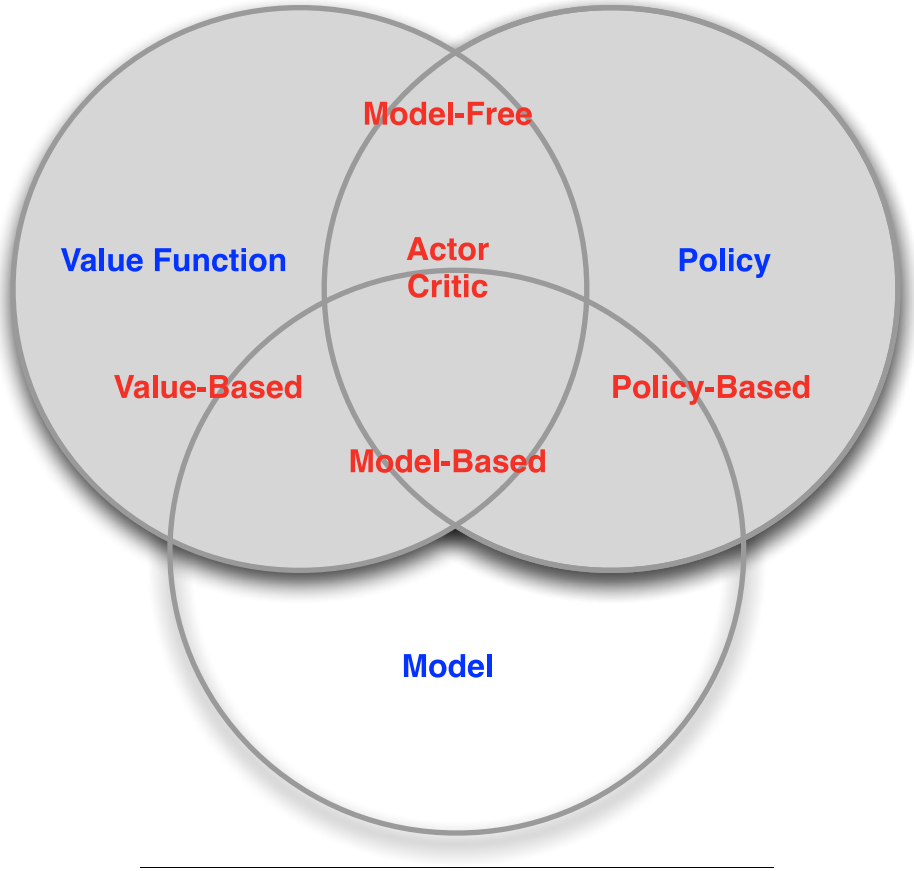
A model does not give us immediately a good policy.

But it allows us to plan – test possible alternative actions.

(Hasselt und Borsa 2021)

Categorization of Reinforcement Learning Agents

- Value Based
  - No Policy (Implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function



(Silver 2015)

Optimal Policies

Optimal Policy

There is a partial ordering over policies which means:

$$\pi \geq \pi' \text{ if } V_\pi(s) \geq V_{\pi'}(s), \forall s$$

For any Markov Decision process ...

- There is an optimal policy π_* which is better (or equal) than all other policies.
- Every optimal policy achieves the optimal value function and the optimal action-value function.

Optimal Value Function

The optimal state-value function $V_*(s)$ is the maximum value function over all policies:

$$V_*(s) = \max_\pi V_\pi(s)$$

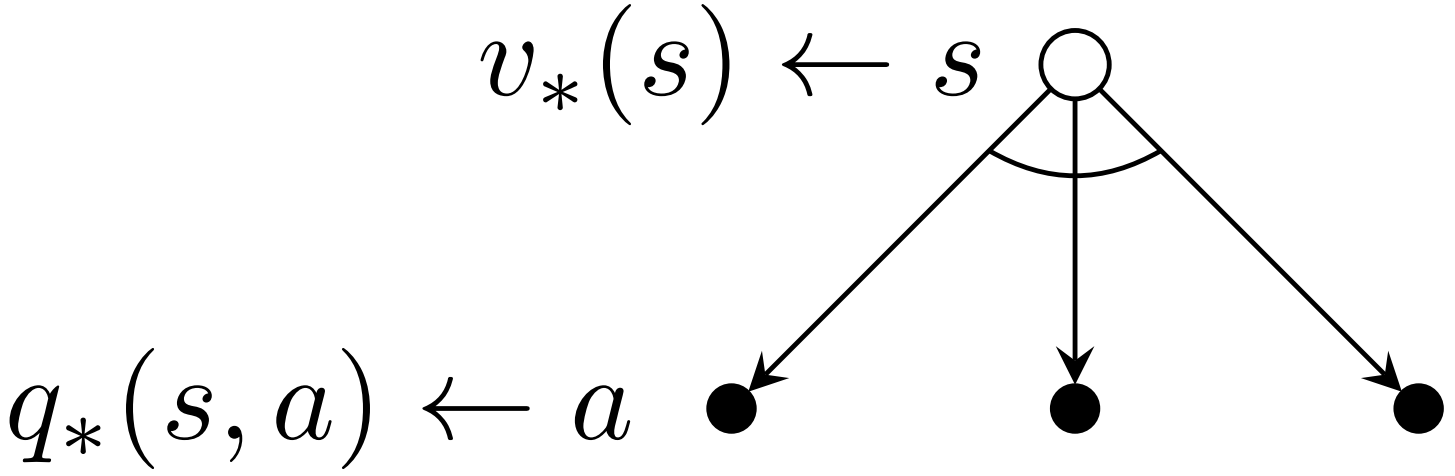
The optimal action-value function $q_*(s, a)$ is the maximum action-value function over all policies

$$Q_*(s, a) = \max_\pi Q_\pi(s, a)$$

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is “solved” when we know the optimal value function.

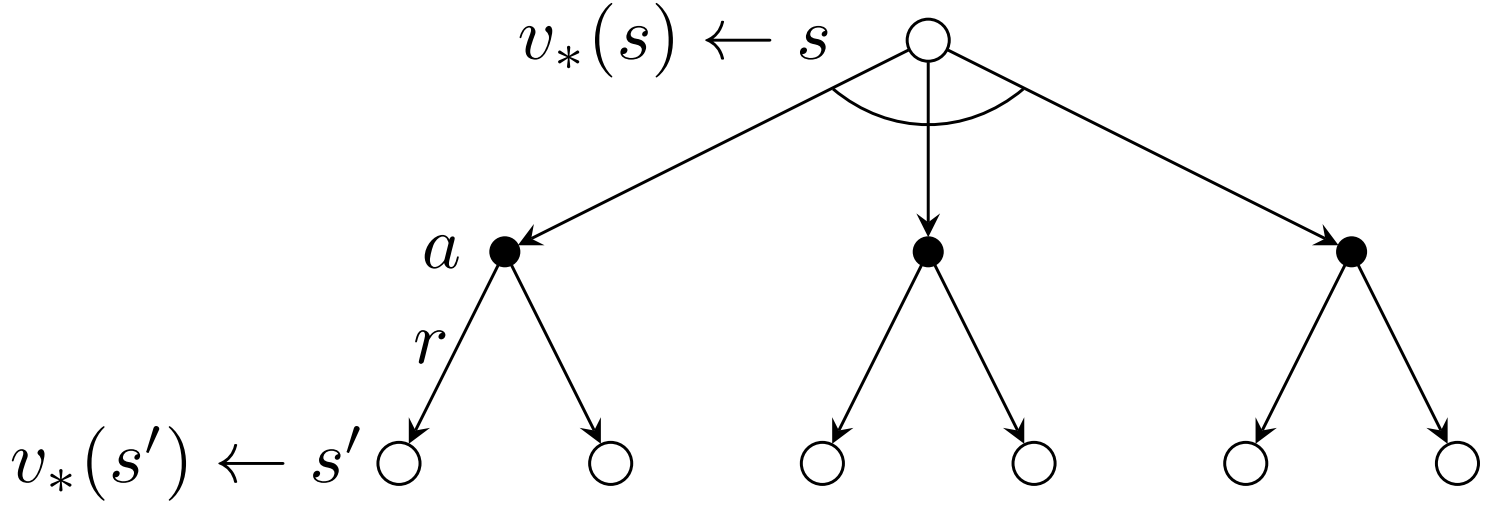
(Silver 2015)

Bellman Optimality Equation for V_*



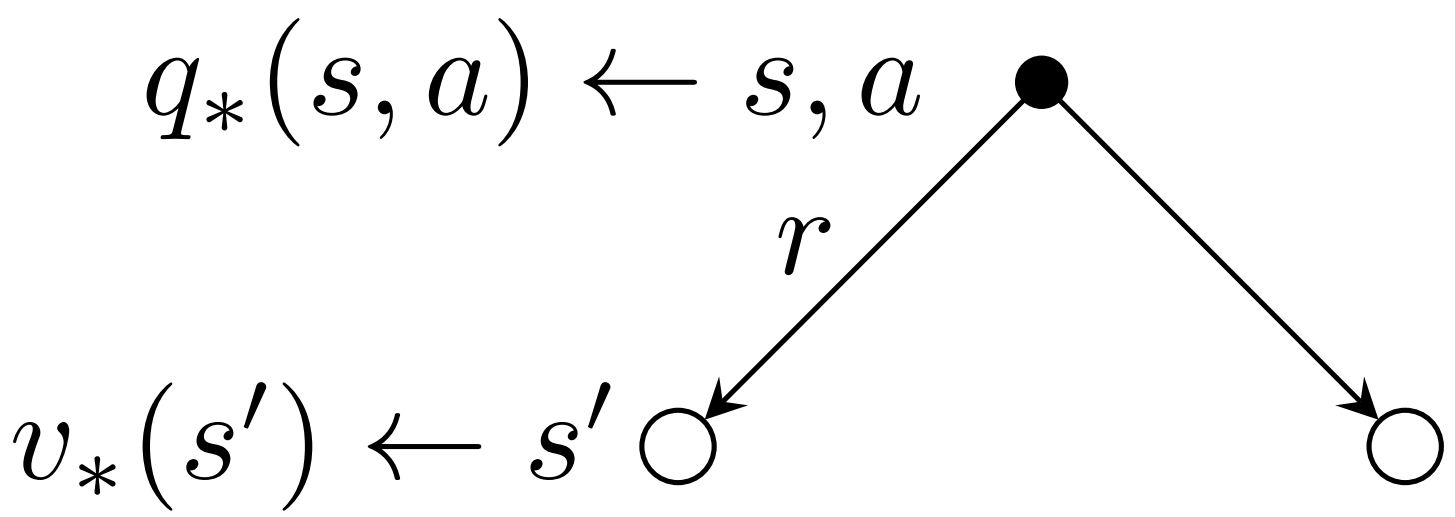
$$v_*(s) = \max_{a \in \mathcal{A}} q_*(s, a)$$

Bellman Optimality Equation for V_* (2)



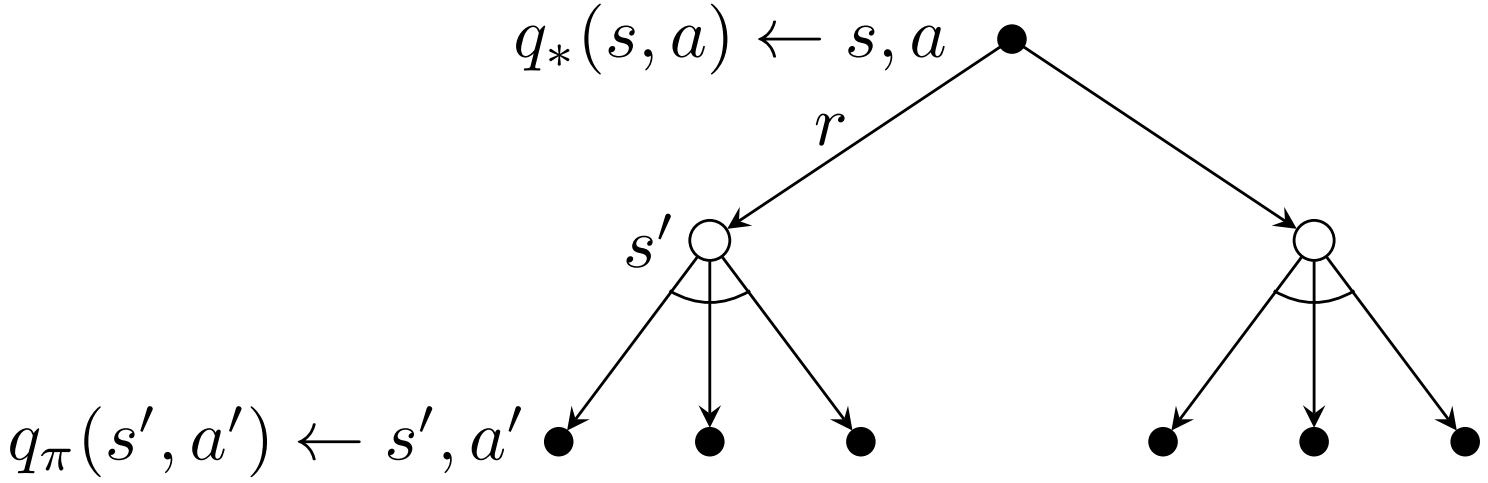
$$v_*(s) = \max_{a \in \mathcal{A}} \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_*(s') \right)$$

Bellman Expectation for Q_*



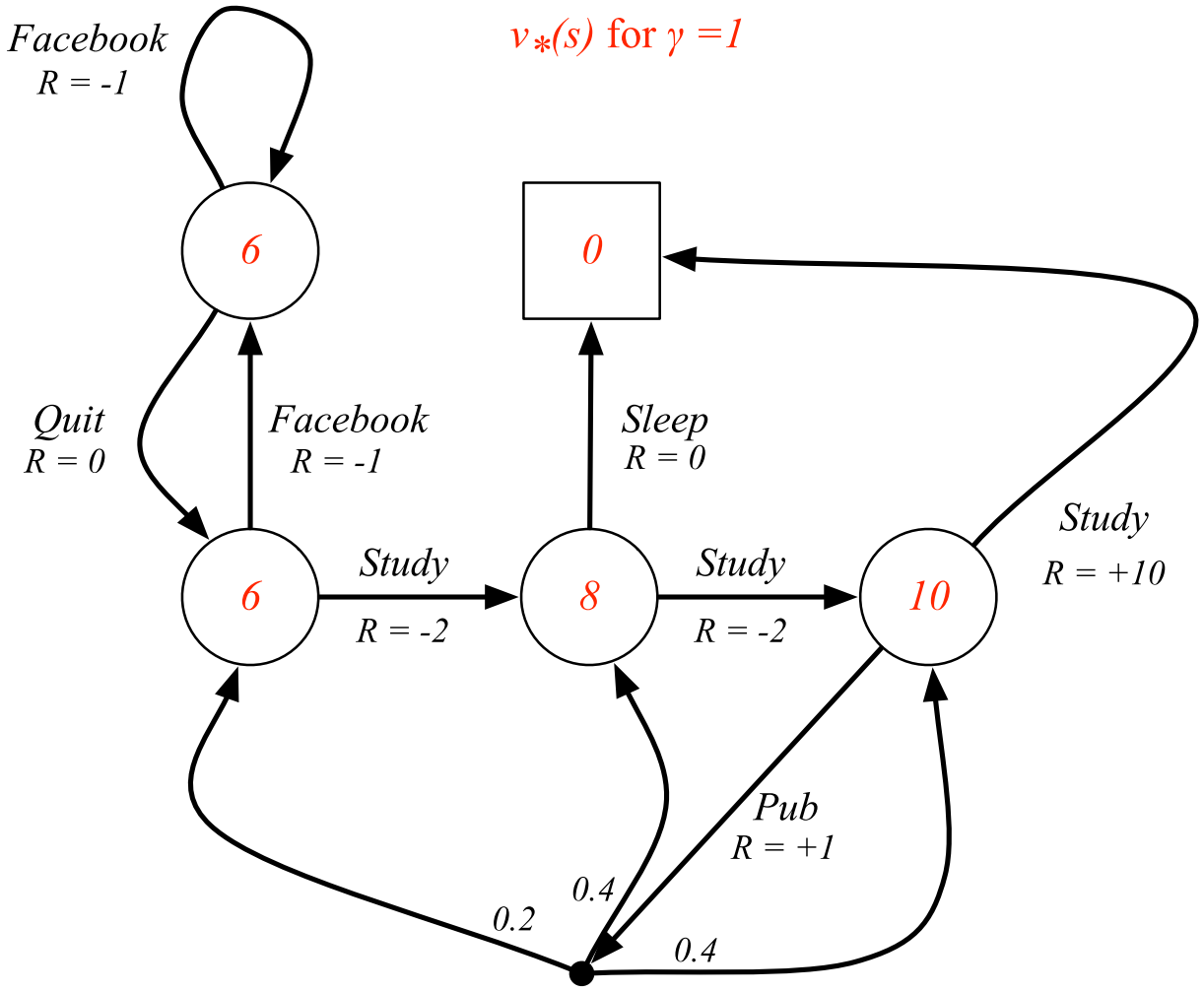
$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_*(s')$$

Bellman Expectation for Q_* (2)



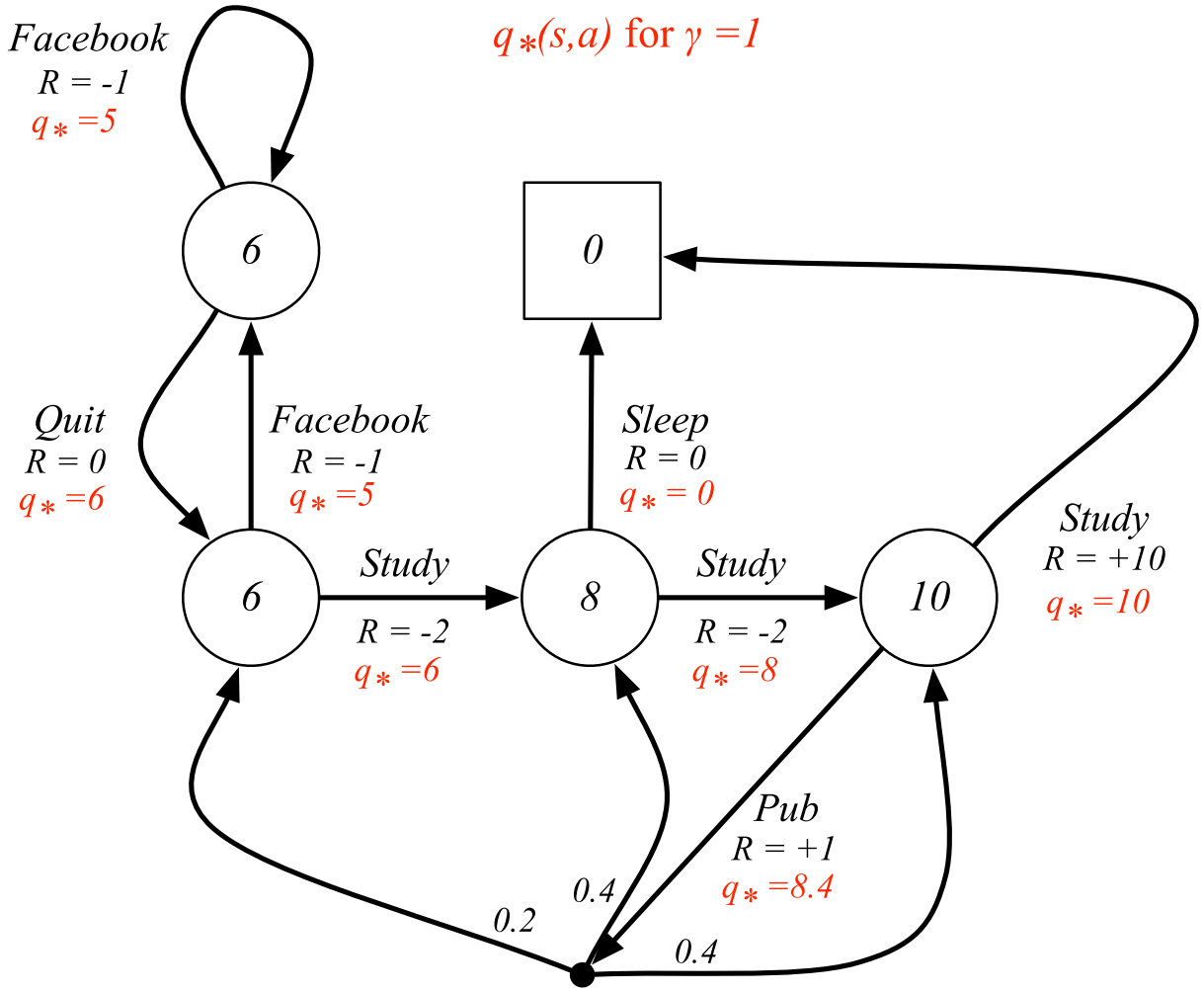
$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} q_*(s',a')$$

Example: Optimal Value Function



(Silver 2015)

Example: Optimal Action-Value Function



(Silver 2015)

Bellman Optimality Equations

We can use these equations directly to compute optimal values for following an optimal policy.

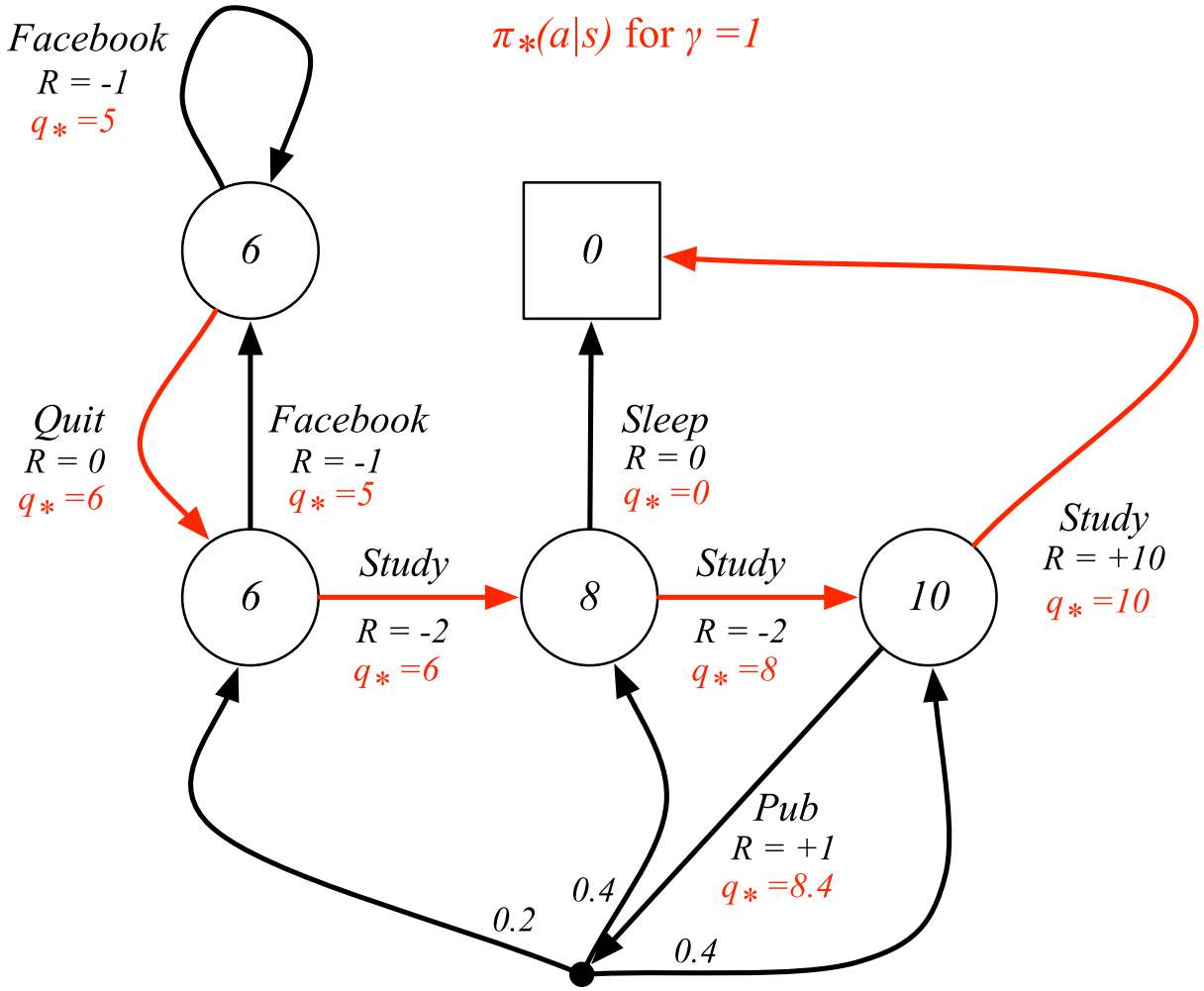
$$V_*(s) = \max_{a \in \mathcal{A}} Q_*(s, a)$$

$$Q_*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_*(s')$$

$$V_*(s) = \max_{a \in \mathcal{A}} \left(R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_*(s') \right)$$

$$Q_*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} \max_{a' \in \mathcal{A}} Q_*(s', a')$$

Example: Optimal Policy



(Silver 2015)

Optimal Value Function and Policy

The goal in RL is to act optimally – this is possible through learning an optimal value function or directly an optimal policy.

The optimal value function produces the maximum return:

$$V_*(s) = \max_\pi V_\pi(s), Q_*(s,a) = \max_\pi Q_\pi(s,a)$$

The optimal policy achieves optimal value functions:

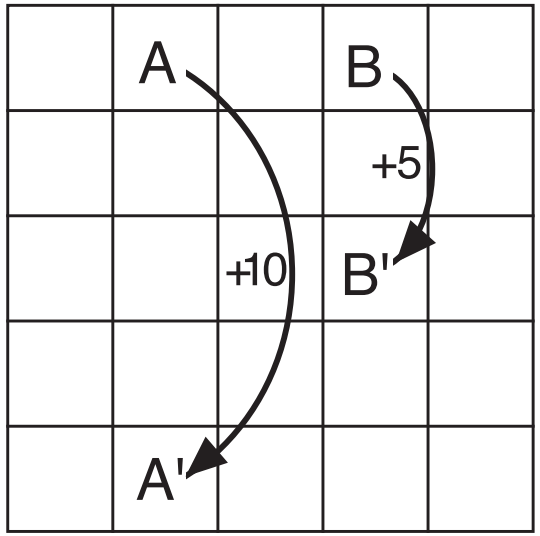
$$\pi_* = \arg\max_\pi V_\pi(s), \pi_* = \arg\max_\pi Q_\pi(s,a)$$

These are directly related as

$$V_{\pi_*}(s) = V_*(s), Q_{\pi_*}(s,a) = Q_*(s,a)$$

(Weng 2018a)

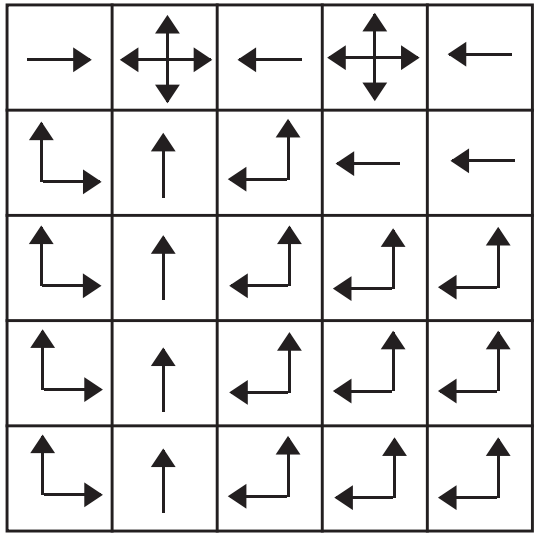
Optimal Solution for the Gridworld Example



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

Each location is a state. Discount factor is 0.9.

Actions: North, West, South, East

Reward: -1 when trying to move out of the grid, 0 otherwise

For state A and B: all actions lead to A' and a reward of $+10$ (respectively B', $+5$). (Sutton und Barto 2018)

Example: Robot Playing Soccer – Hierarchical RL

See Video of robots playing soccer

Details see (Huang u. a. 2022)

References

Hasselt, Hado van, und Diana Borsa. 2021. „Reinforcement Learning Lecture Series 2021“. https://www.deepmind.com/learning-resources/reinforcement-learning-lecture-series-2021.

Huang, Xiaoyu, Zhongyu Li, Yanzhen Xiang, Yiming Ni, Yufeng Chi, Yunhao Li, Lizhi Yang, Xue Bin Peng, und Koushil Sreenath. 2022. „Creating a Dynamic Quadrupedal Robotic Goalkeeper with Reinforcement Learning“. arXiv. doi:10.48550/ARXIV.2210.04435.

Santucci, Andreas. 2019. „Course Notes from RL Specialization, University of Alberta“. https://github.com/asantucci/rl_notes.

Silver, David. 2015. „UCL Course on RL UCL Course on RL UCL Course on Reinforcement Learning“. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html.

Sutton, Richard S., und Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. Second. The MIT Press.

Weng, Lilian. 2018a. „A (Long) Peek into Reinforcement Learning“. https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html.

———. 2018b. „The Multi-Armed Bandit Problem and Its Solutions“. The Multi-Armed Bandit Problem and Its Solutions.