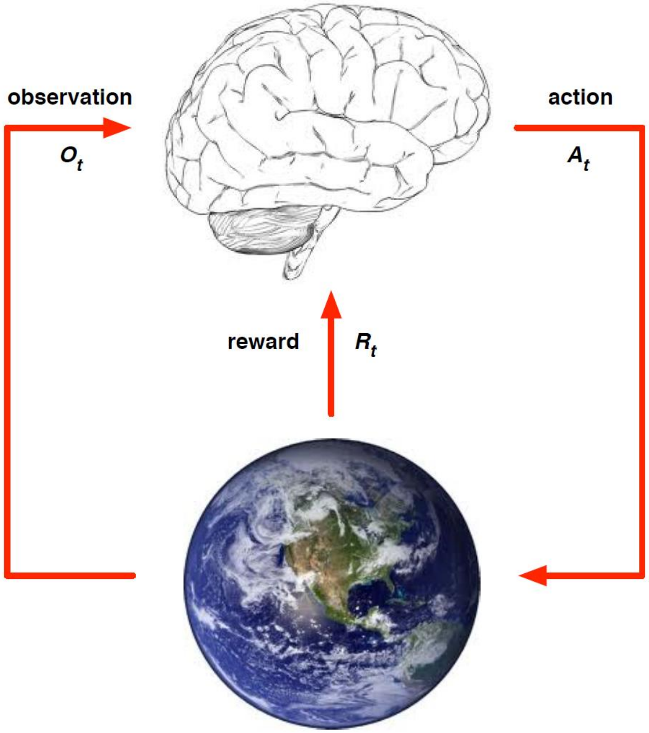
Basics of RL

Basics of RL

General setup

- An **agent** interacts with its **environment** at discrete timesteps t
- At each time step t the agent
  - executes **action** A_t
  - receives **observation** O_{t+1}
  - receives scalar **reward** R_{t+1}
- At each time step the environment
  - receives **action** A_t
  - emits **observation** O_{t+1}
  - emits scalar **reward** R_{t+1}
- The **history** H_t is the sequence of observations/ rewards/actions until time t, e.g. all the information that the agent has received so far

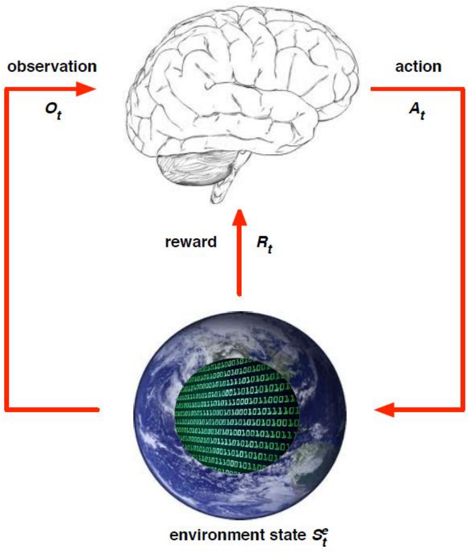
$$H_t = O_1, R_1, A_1, \dots, A_{t-1}, O_t, R_t$$

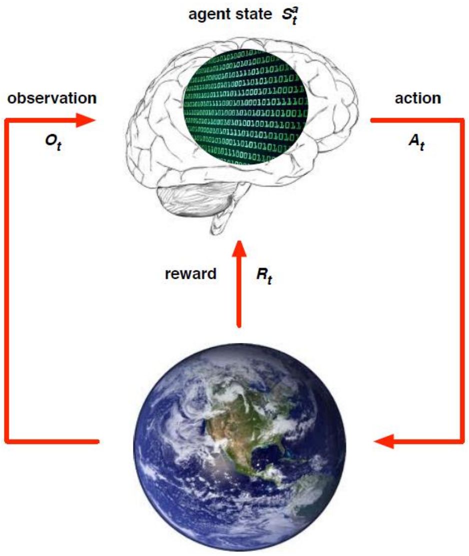


https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

Basics of RL

- The **state** S_t is the information used to determine what happens next
- The **environment state** S_t^e is the environments private representation (e.g. the entire world)
- The **agent state** S_t^a is the agent's internal representation based on the history $S_t^a = f(H_t)$ (the information used to determine which action to pick next)





https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

Basics of RL

- A **Markovian state** (state with Markov property) contains all useful information from the past (once the state is known, the history may be thrown away)
  - Mathematically: $P(S_{t+1}|S_t) = P(S_{t+1}|S_1 \dots S_t)$
  - Colloquially: Next state depends only on the current state and not the past states

- If the environment has only Markovian states, it is called a **Markovian environment**

- If the environment is **fully observable**, the agent observes the environment state (and usually sets its own state to the observed one)

$$O_t = S_t^e (= S_t^a = S_t)$$

- If the environment is **partially observable**, the agent observes only a part of the environment state and has to "guess" its environment state from the observation

$$O_t \subset S_t^e$$

Basics of RL

- The entity of state/action transitions until reaching a **goal/terminal state** (or until the time expires) is called an **episode**
- Common temporal sequence: Agent starts in state S_0, performs actions A_0, experiences next state S_1 and reward R_1, performs actions A_1, experiences next state S_1 and reward R_1, … until reaching a goal state

$$S_0 \to A_0 \to S_1, R_1 \to A_1 \to S_2, R_2 \to \ \dots \ \to S_{goal}$$

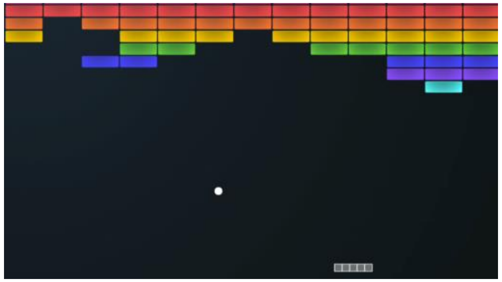
- Note: There exist also nonepisodic tasks without goal state (covered only superficially within this lecture)

Task: Can you think of a nonepisodic task that can be solved with RL?

Basics of RL

Examples for non-Markovian states

- Breakout



https://www.coolmathgames.com/sites/default/files/Atari%20Breakout%20OG%20Image.png

Examples for Markovian states

- Chess (no need to keep track of the move history)

https://www.chess.com/de/article/view/die-besten-schachpartien-aller-zeiten

Examples for fully observable environments

- Chess

Examples for partially observable environments

- Uno/Poker (you do not know the other players' hands)
- Robot with camera (can only sense a part of its environment)
- Driving (cars behind a corner are invisible)

Basics of RL

Task: Figure out if the following environments are fully/partially observable and if a given state for you as the agent is Markovian or not. How long is an episode (if applicable)?

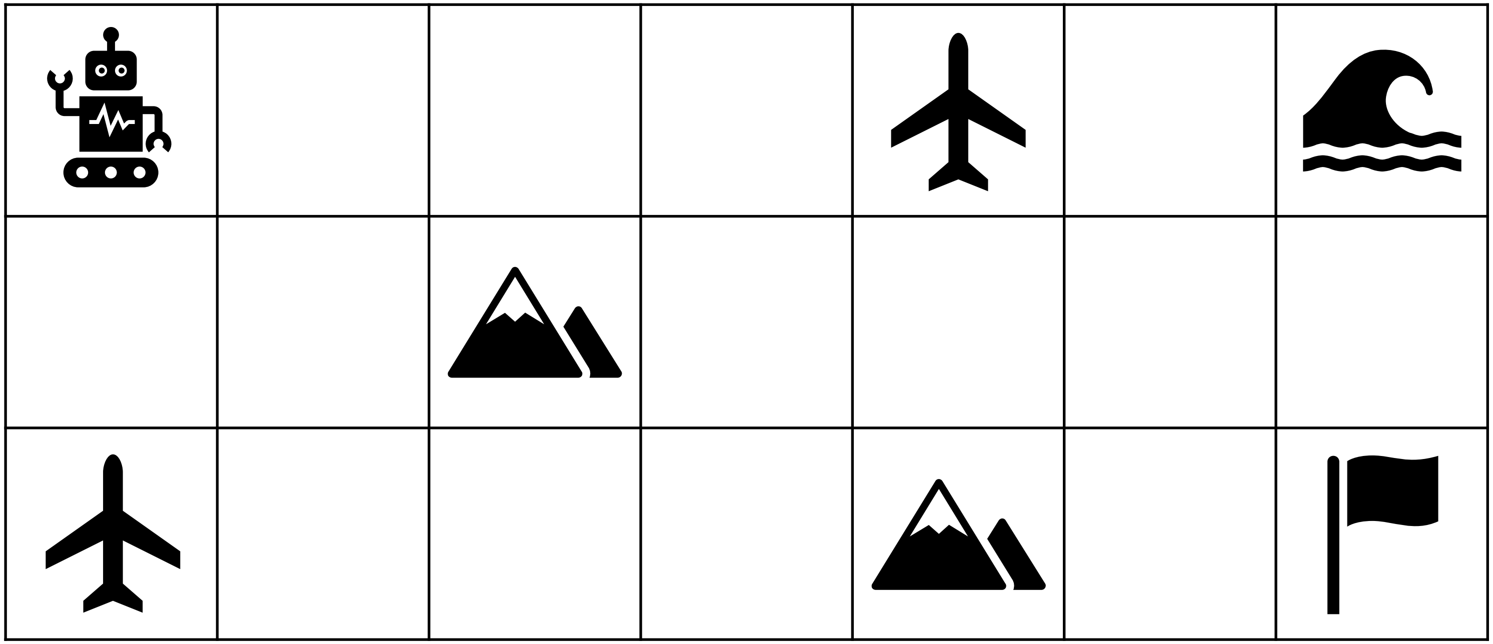
- Monopoly
- Poker
- Black Jack
- Russian roulette
- weather
- stock market data (only today's data)
- stock market data (including historical data)

https://m.media-amazon.com/images/I/51AnnChCJcL._AC_SY580_.jpg

Basics of RL

Example: Robot maze

- Robot 🤖 wants to find shortest path from its current location to the goal ⚑
- The episode is over if he either reaches the goal (win) or drowns in the waves 🌊 (loss)
- There are cells ⛰ it cannot trespass and two airports ✈
(if he enters one of the the two cells, he will be placed immediately on the other one)

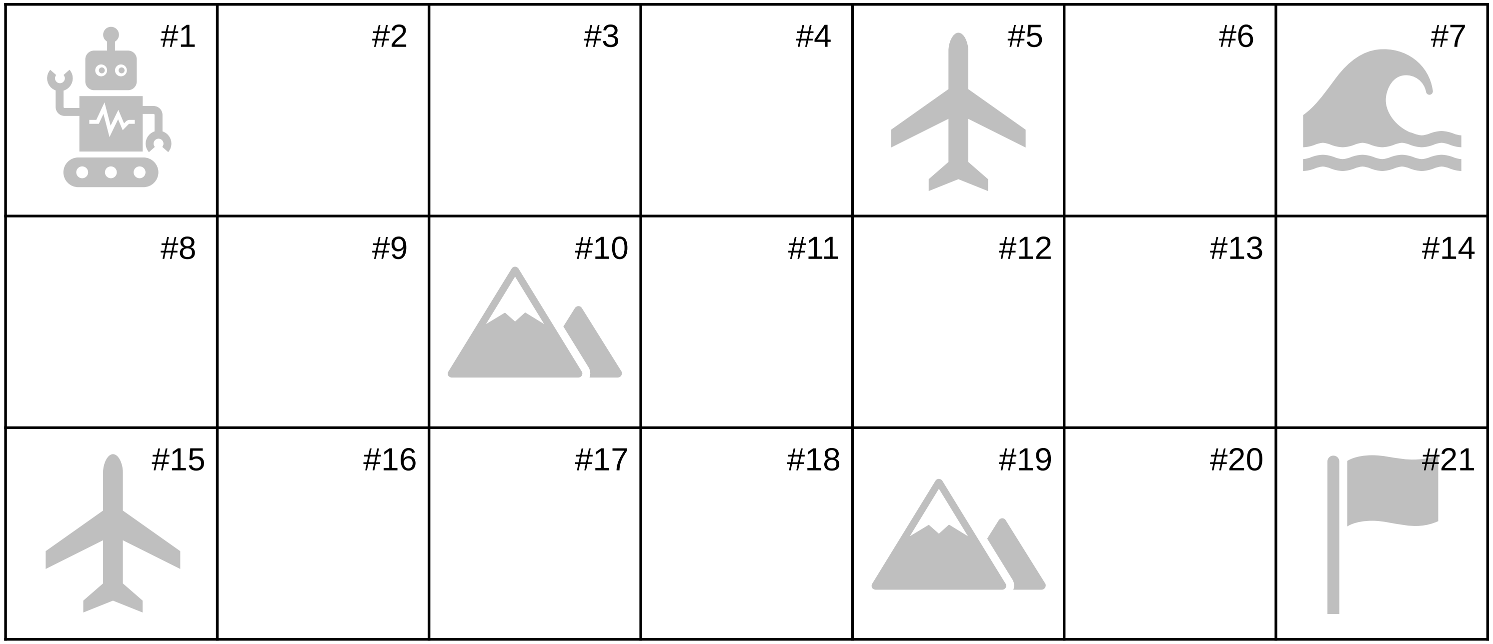


🤖				✈		🌊
		⛰				
✈				⛰		⚑

Basics of RL

Example: Robot maze states

- The state is the cell index (e.g. #16) that the robot (agent) is currently occupying
- The goal state is #21. If the robot reaches the goal state, the episode is finished

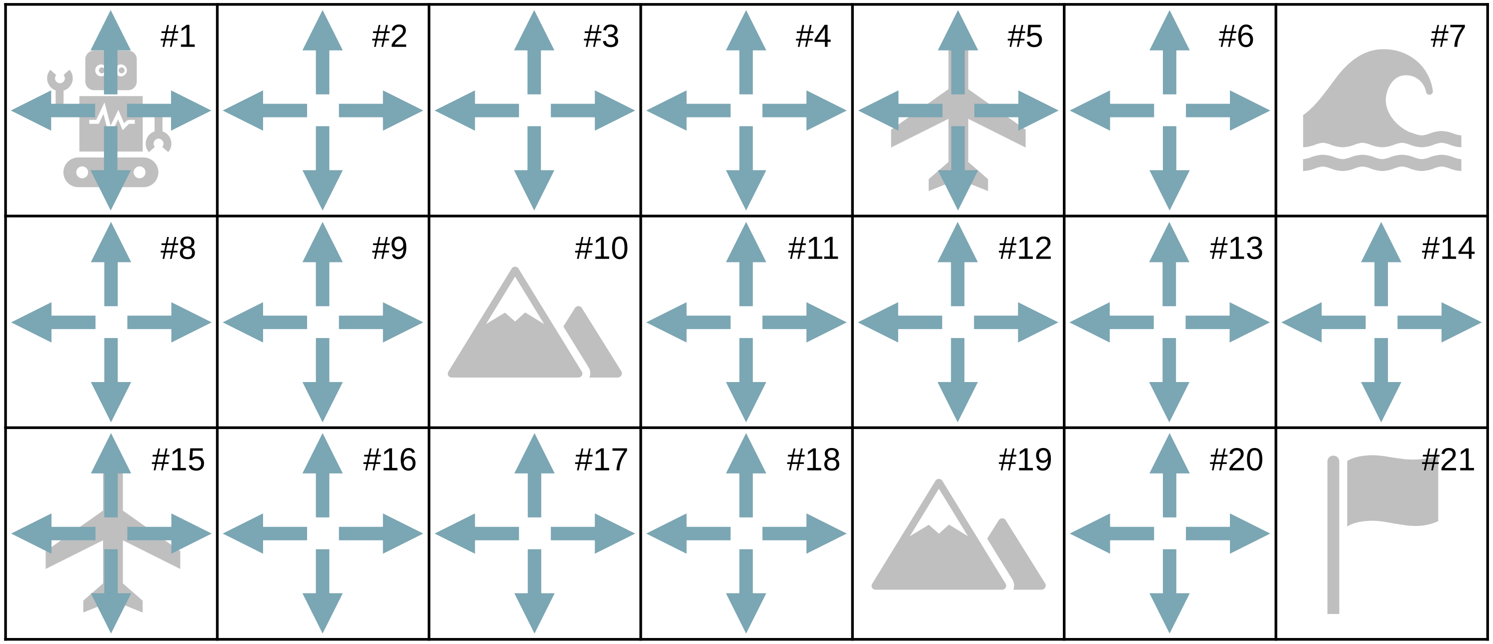


#1 🤖	#2	#3	#4	#5 ✈	#6	#7 🌊
#8	#9	#10 ⛰	#11	#12	#13	#14
#15 ✈	#16	#17	#18	#19 ⛰	#20	#21 🚩

Basics of RL

Example: Robot maze actions

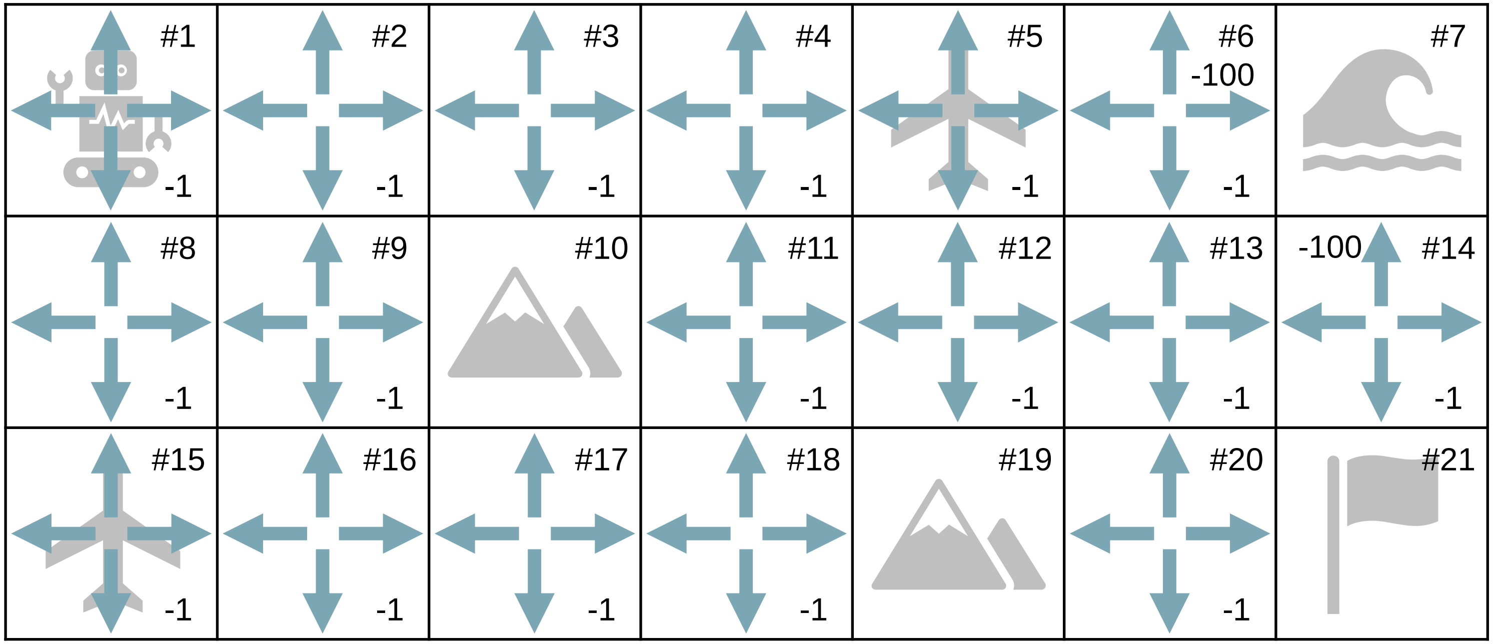
- The robot's actions are move up / down / left / right
- If the corresponding adjacent cell is free, it will move to it. Otherwise nothing will happen



Basics of RL

Example: Robot maze rewards

- "Handcrafted" reward: The robot gets a reward of -1 for every action he takes



#1 -1	#2 -1	#3 -1	#4 -1	#5 -1	#6 -100 -1	#7
#8 -1	#9 -1	#10	#11 -1	#12 -1	#13 -1	-100 #14 -1
#15 -1	#16 -1	#17 -1	#18 -1	#19	#20 -1	#21

Basics of RL

Example: Robot maze state-action transitions

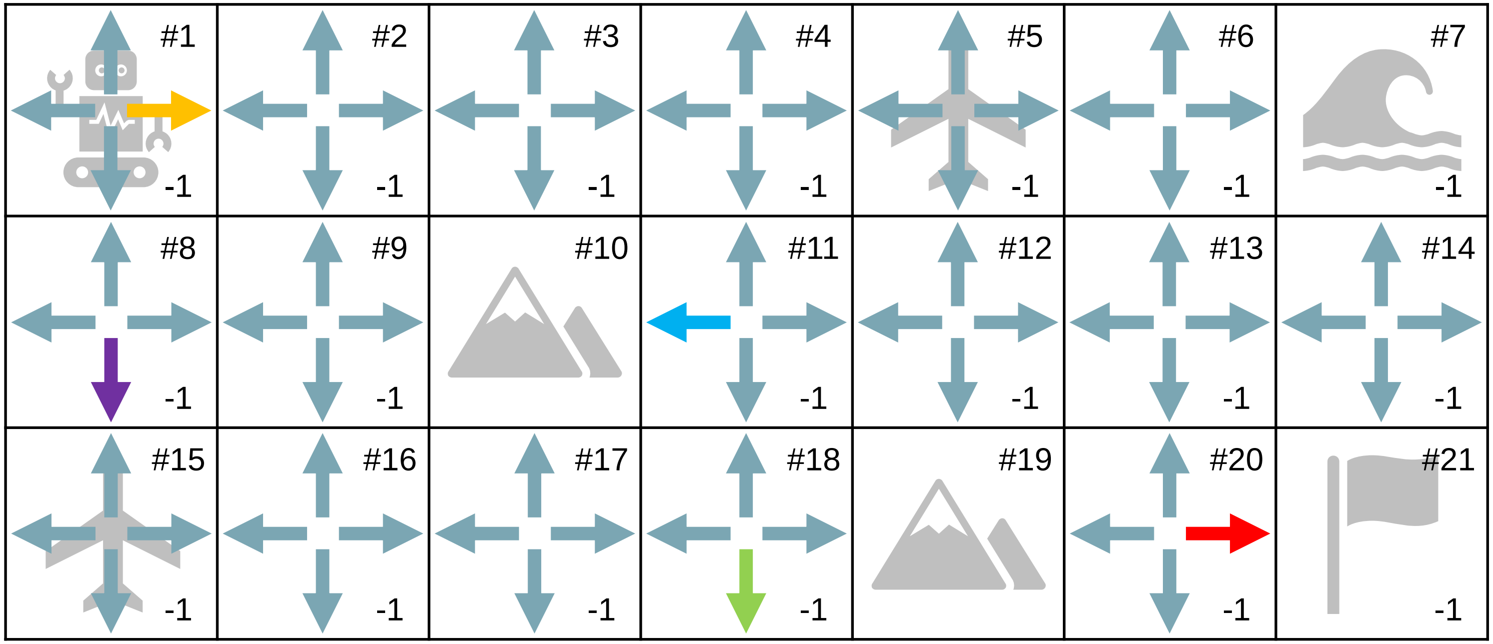
→ $S_t = \#1, A_t = right \;\rightarrow\; R_{t+1} = -1, S_{t+1} = \#2$

↓ $S_t = \#18, A_t = down \;\rightarrow\; R_{t+1} = -1, S_{t+1} = \#18$

↓ $S_t = \#8, A_t = down \;\rightarrow\; R_{t+1} = -1, S_{t+1} = \#5$

← $S_t = \#11, A_t = left \;\rightarrow\; R_{t+1} = -1, S_{t+1} = \#11$

→ $S_t = \#20, A_t = right \;\rightarrow\; R_{t+1} = -1, S_{t+1} = \#21$



#1 -1	#2 -1	#3 -1	#4 -1	#5 -1	#6 -1	#7 -1
#8 -1	#9 -1	#10	#11 -1	#12 -1	#13 -1	#14 -1
#15 -1	#16 -1	#17 -1	#18 -1	#19	#20 -1	#21 -1

Basics of RL

Markov decision process (MDP)

- Formalizes state-action transitions (previous example)
- Also considers probabilistic transitions

A Markov decision process (MDP) consists of

- A set of states $\mathcal{S}$ (**state space**)
- A set of actions $\mathcal{A}$ (**action space**)
- **Transition probabilities** of the form

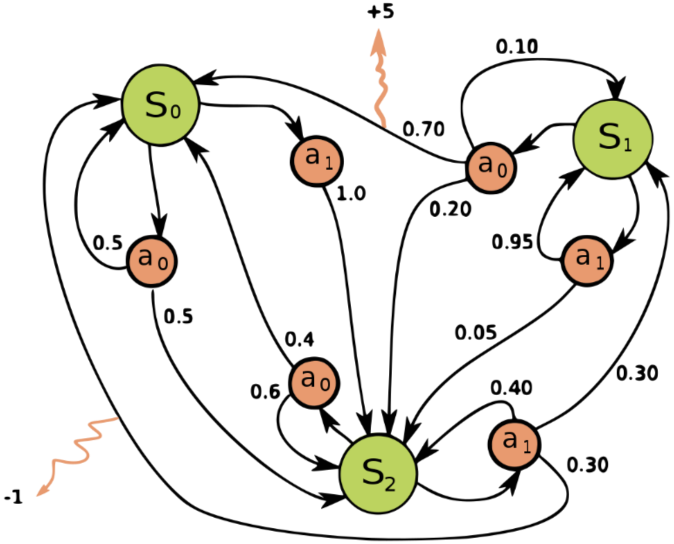
$$P(S_{t+1} = s' \,|S_t = s, A_t = a)$$

specifying the probability that taking action a in state s will lead to state s' at the next timestep

- An immediate **reward function**

$$R_a(s, s')$$

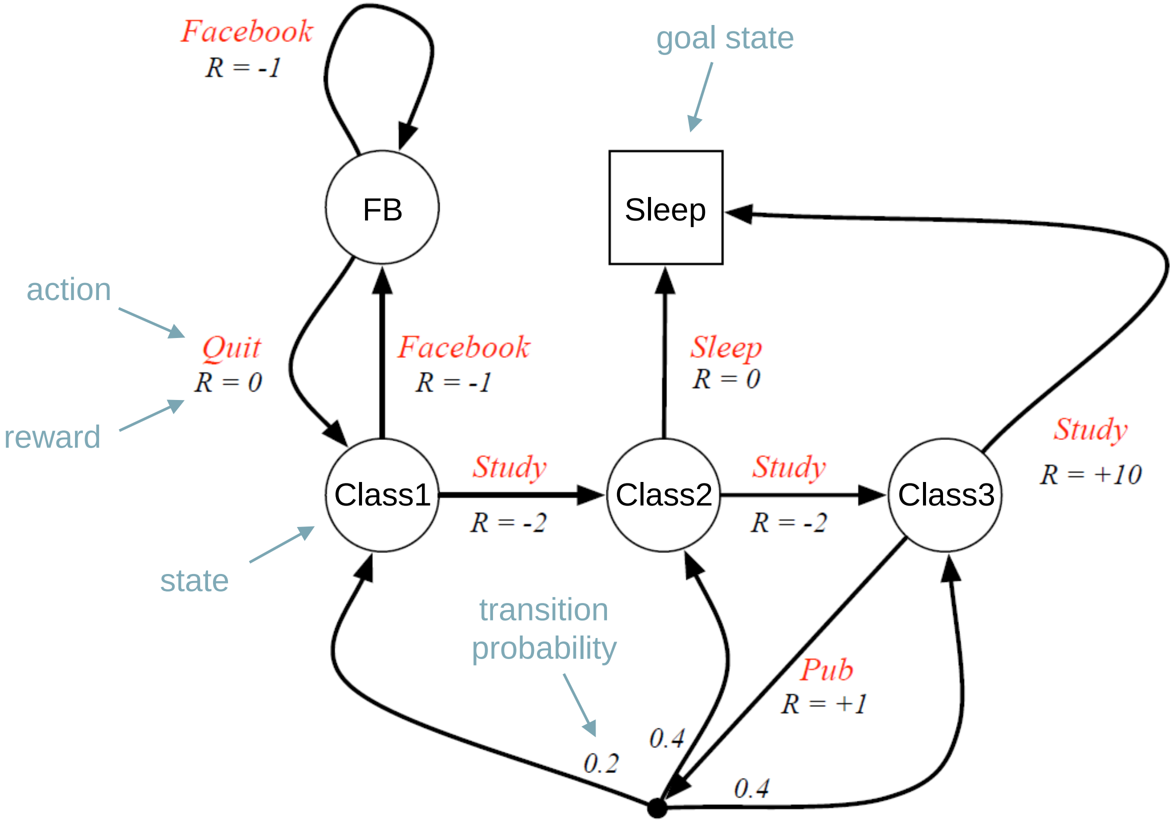
when transitioning to state s' after taking action a in state s



https://en.wikipedia.org/wiki/Markov_decision_process

Basics of RL

Example: One day of a student

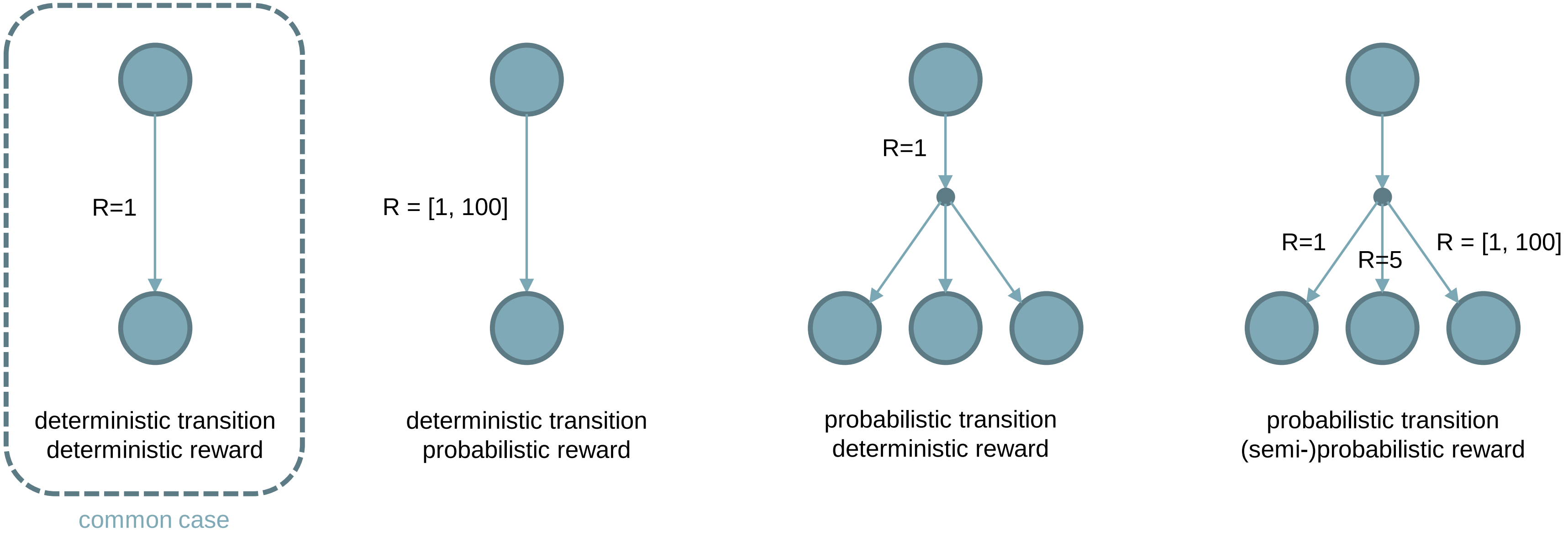


https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

Task: How can the robot maze be written as a MDP?

Basics of RL

- There exist different combinations of reward functions and transition probabilities



- In the non-deterministic case, one has to use probabilities / expectation values → don't be afraid!

Basics of RL

Policy

- Describes the agent's behaviour for a given MDP by a set of rules of the form "when in state s, do action a"
- A **policy** π is a distribution over actions given states

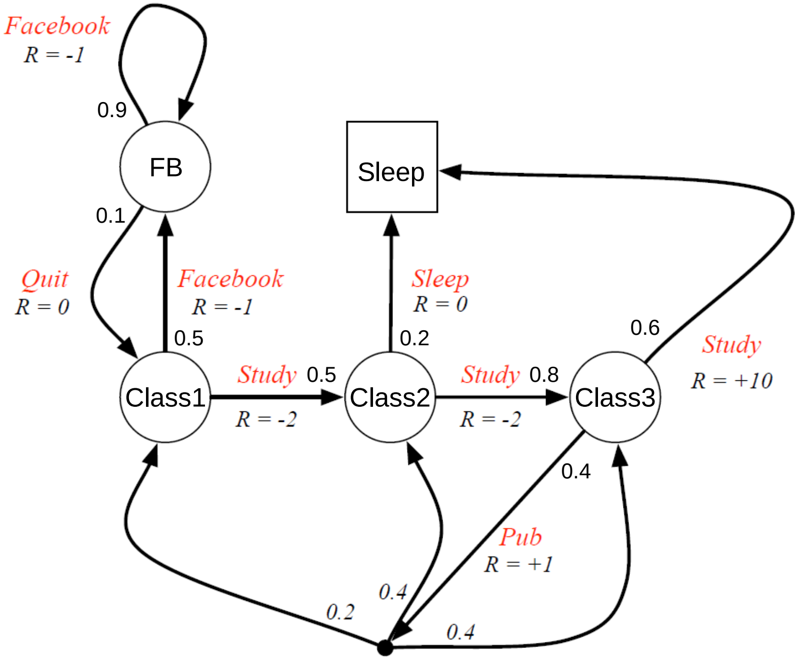
$$\pi(a|s) = P(A_t = a \mid S_t = s)$$

- Above formulation allow both for **deterministic policies** (when in state s, do only action a) as well as **stochastic policies** (when in state s, do action a with a certain percentage, otherwise do another action)
- Policies are (in general) **stationary**, i.e. they do not depend on the time t

Basics of RL

Example: One day of a student policy

- $P(Facebook \mid FB) = 0.9$
- $P(Quit \mid FB) = 0.1$
- $P(Facebook \mid Class1) = 0.5$
- $P(Study \mid Class1) = 0.5$
- $P(Sleep \mid Class2) = 0.2$
- $P(Study \mid Class2) = 0.8$
- $P(Pub \mid Class3) = 0.4$
- $P(Study \mid Class3) = 0.6$

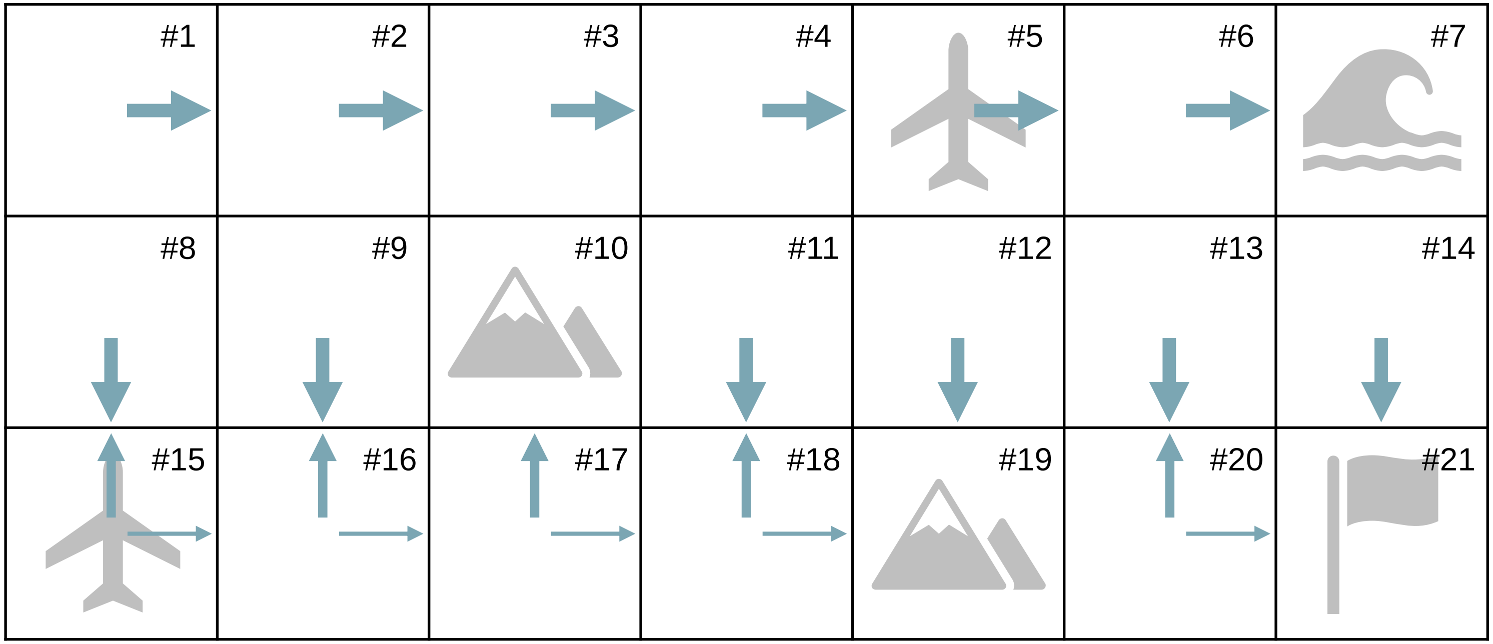


https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

Basics of RL

Example: Robot maze policy (not the best one…)

- $P(right \mid \#1 \ldots \#6) = 1$
- $P(down \mid \#8 \ldots \#14) = 1$
- $P(right \mid \#15 \ldots \#20) = 0.3$
- $P(up \mid \#15 \ldots \#20) = 0.7$



Basics of RL

Return

- The **return** G_t for an episode is the discounted cumulative reward from time step t on

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \dots \ = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The **discount factor** $y \in [0,1]$ is the present value of future rewards
- $y = 0$ only considers the instant reward
- y close to 0 leads to myopic evaluation
- y close to 1 leads to far-sighted evaluation

Basics of RL

Example: One day of a student

- Given is the episode with

  $Class1 \rightarrow Study \rightarrow Class2 \rightarrow Study \rightarrow Class3 \rightarrow Pub \rightarrow Class3 \rightarrow Study \rightarrow Sleep$

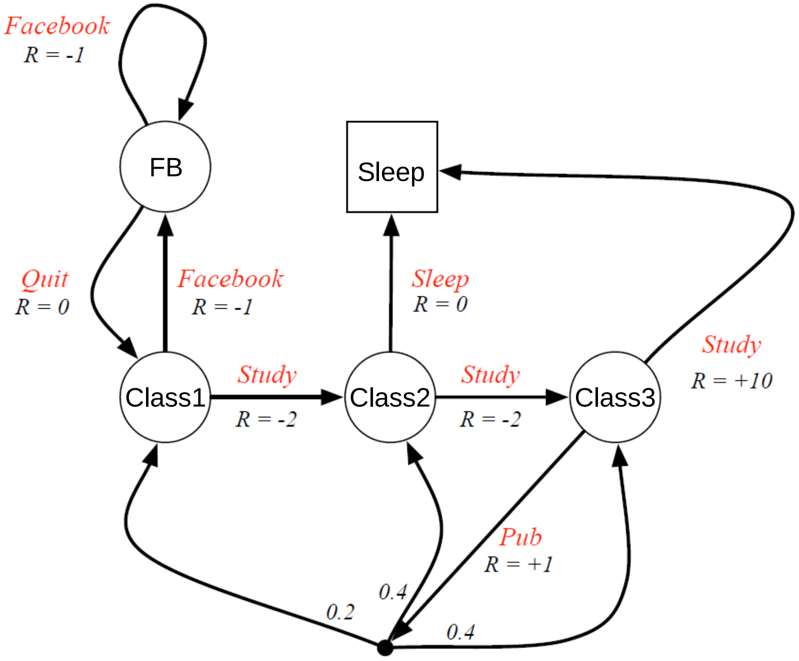
- With $\gamma = 1$ the return of each state of the episode becomes

  $G_0(Class1) = -2 - 2 + 1 + 10 = 7$

  $G_1(Class2) = -2 + 1 + 10 = 9$

  $G_2(Class3) = +1 + 10 = 11$

  $G_3(Class3) = +10 = 10$



https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

Basics of RL

- With $\gamma = 0.9$ the return of each state of the episode becomes

$$G_0(Class1) = -2 - 2 \cdot 0.9 + 1 \cdot 0.9^2 + 10 \cdot 0.9^3 = 4.3$$
$$G_1(Class2) = -2 + 1 \cdot 0.9 + 10 \cdot 0.9^2 = 7$$
$$G_2(Class3) = +1 + 10 \cdot 0.9 = 10$$
$$G_3(Class3) = +10 = 10$$

Task: What is the return of each state for the given episode for $\gamma = 0.8$?

Task: You have an episode with the following rewards:

$$R_1 = -3, R_2 = 1, R_3 = 2, R_4 = -2, R_5 = 6$$

What are $G_0 \dots G_5$ for $\gamma = 1$ resp. $\gamma = 0.8$?

Basics of RL

Example: Pocket money

- You get 100€ this week ($\gamma = 0.9$)

$$G_0 = 100 + 0 \cdot 0.9 + 0 \cdot 0.9^2 + \ \dots \ = 100$$

- You get 100€ in two weeks ($\gamma = 0.9$)

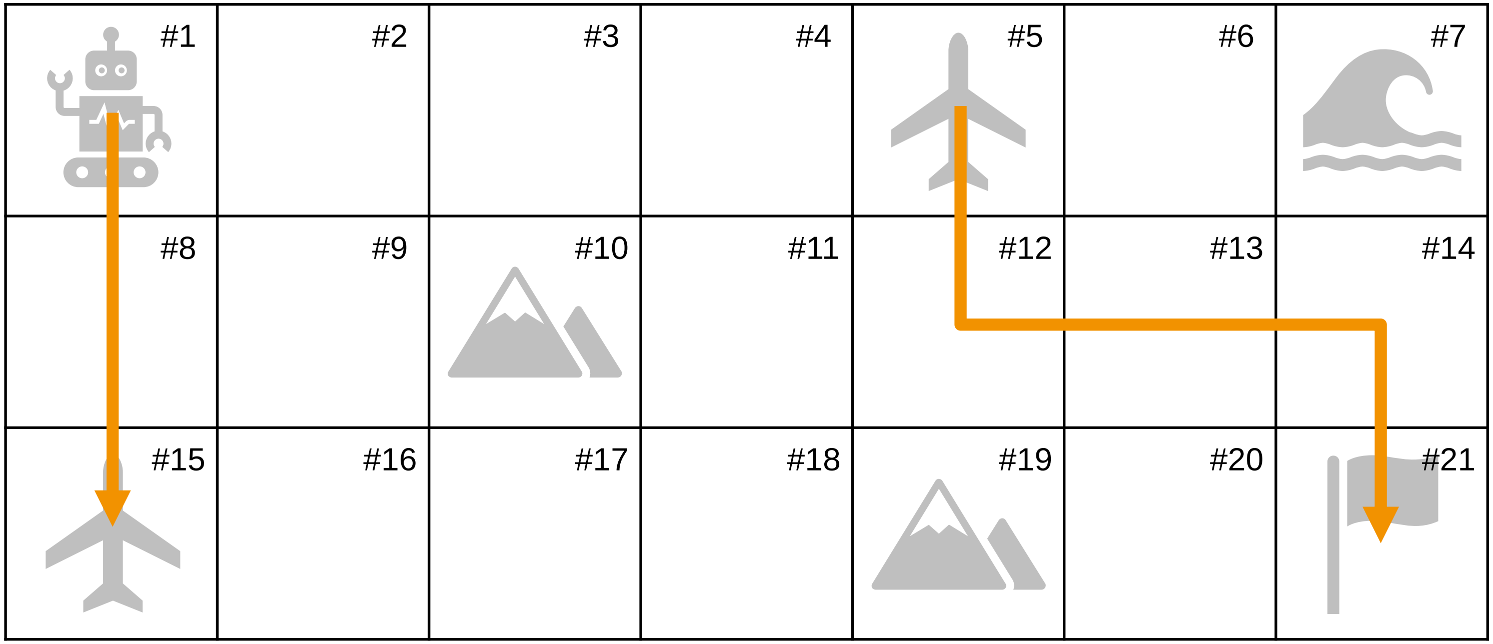
$$G_0 = 0 + 0 \cdot 0.9 + 100 \cdot 0.9^2 + \ \dots \ = 81$$

- You get 100€ every week till infinity ($\gamma = 0.9$)

$$G_0 = 100 + 100 \cdot 0.9 + 100 \cdot 0.9^2 + \ \dots \ = 1000$$

Basics of RL

Task: You are given the following robot maze episode. What is the return of every state if the reward after each action is -1 and the discount factor $\gamma = 0.8$?



Basics of RL

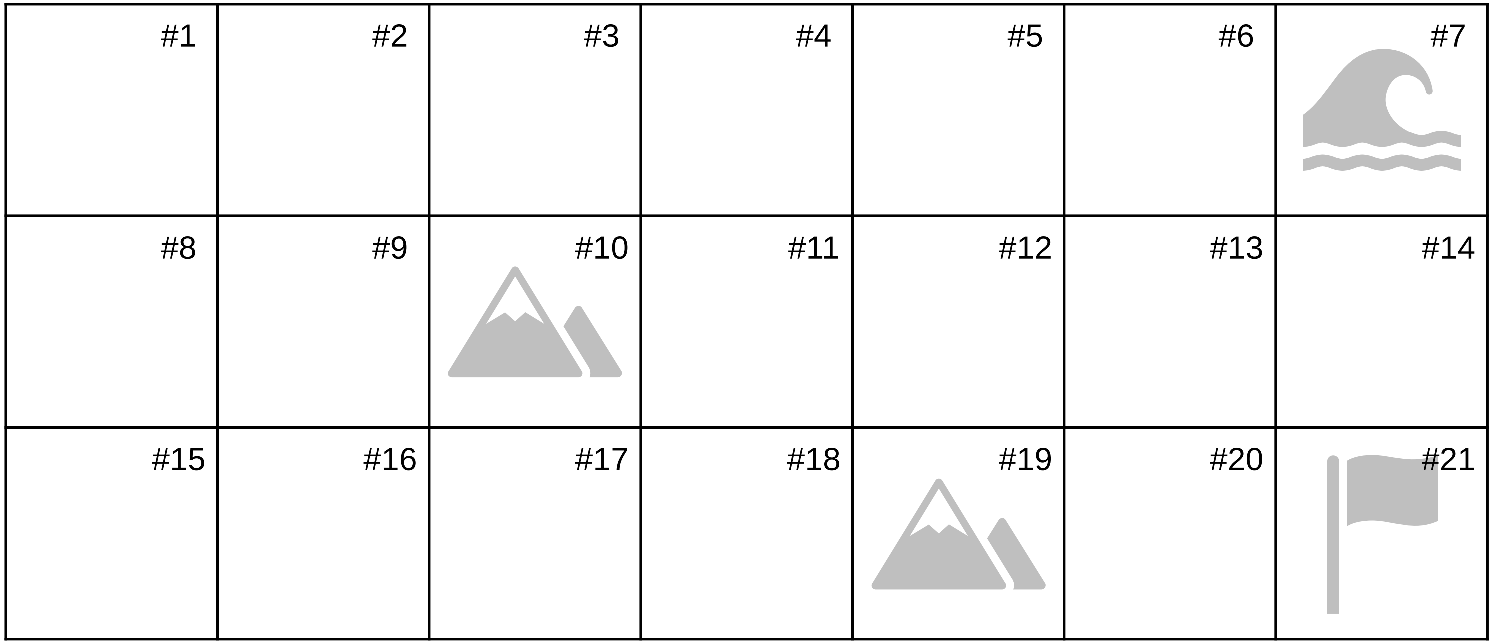
Goal of reinforcement learning (informal)

Find an optimal policy that maximizes the return for all states

Basics of RL

Task: Find an optimal policy that maximizes the return for all states for the robot maze (without airports)

- -1 reward for every action, -100 when entering the waves
- Discount factor $\gamma = 0.9$

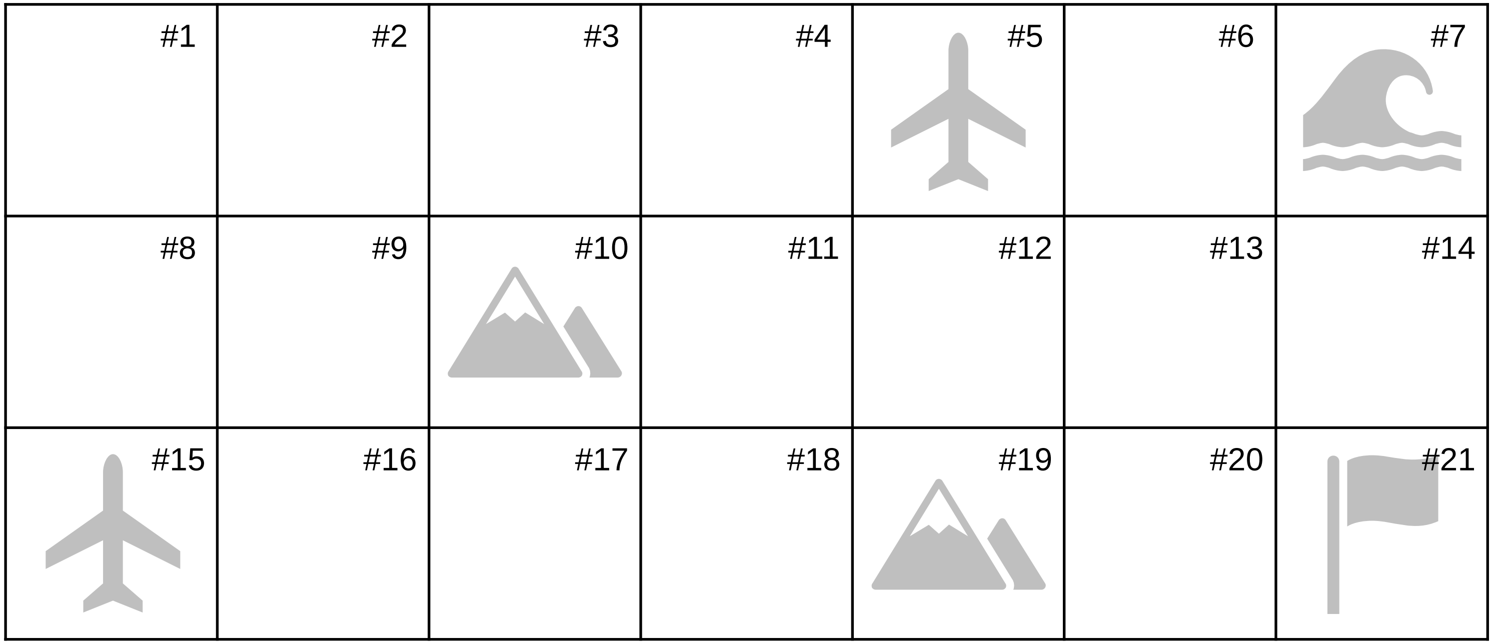


#1	#2	#3	#4	#5	#6	#7 🌊
#8	#9	#10 ⛰	#11	#12	#13	#14
#15	#16	#17	#18	#19 ⛰	#20	#21 🚩

Basics of RL

Task: Find an optimal policy that maximizes the return for all states for the robot maze (with airports)

- -1 reward for every action, -100 when entering the waves
- Discount factor $\gamma = 0.9$

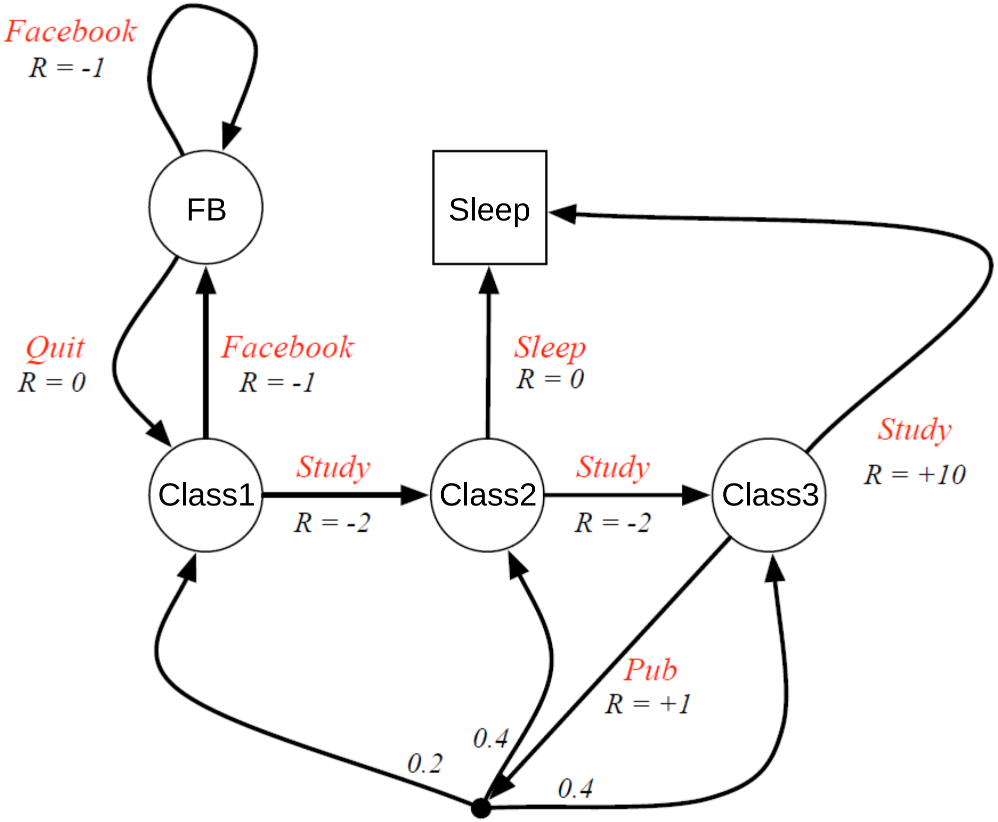


#1	#2	#3	#4	#5 ✈	#6	#7 🌊
#8	#9	#10 ⛰	#11	#12	#13	#14
#15 ✈	#16	#17	#18	#19 ⛰	#20	#21 ⚑

Basics of RL

Task: Find an optimal policy that maximizes the return for all states for "One day of a student"

- Discount factor $\gamma = 0.9$



https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

Basics of RL

Brief summary

- An agent interacts with its environment by executing actions and receiving rewards
- The return is the discounted cumulative reward
- A policy describes the behaviour of the agent (which action to pick in each state)
- The goal of RL is to find an policy that maximizes the agent's return for all states

Basics of RL

Next chapters

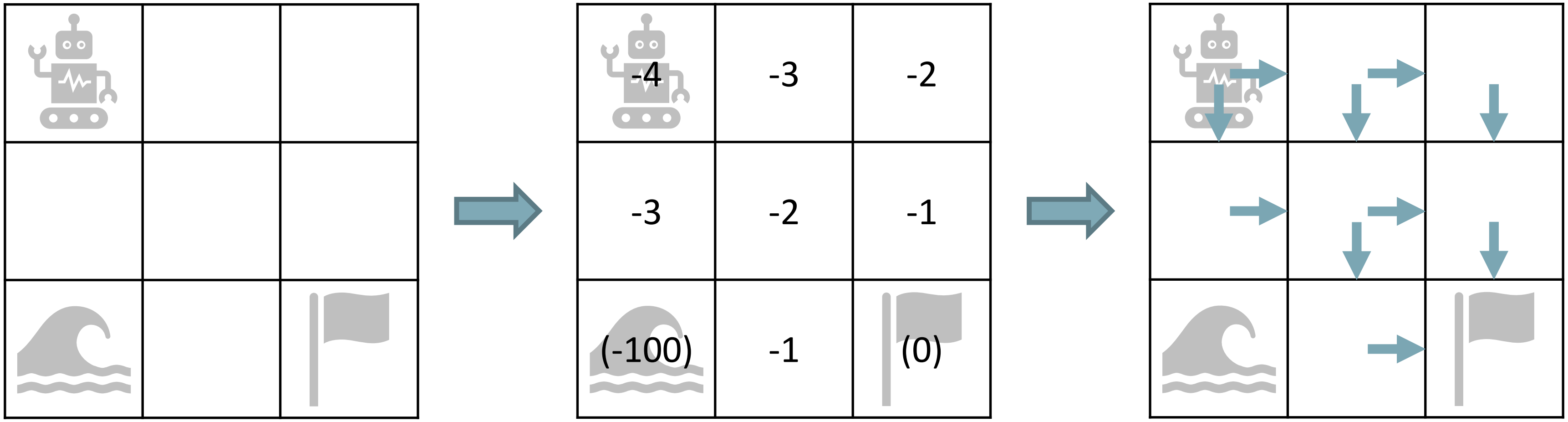
What follows are two conceptually different methods that have the same goal (to find a policy that maximizes the return for all states)

- Value-based methods
- Policy-based methods (aka policy gradients)

Basics of RL

Value-based methods first create a value function for every state/action that tells us how good/bad it is to be in this state/take this action. From this value function the optimal policy can be derived

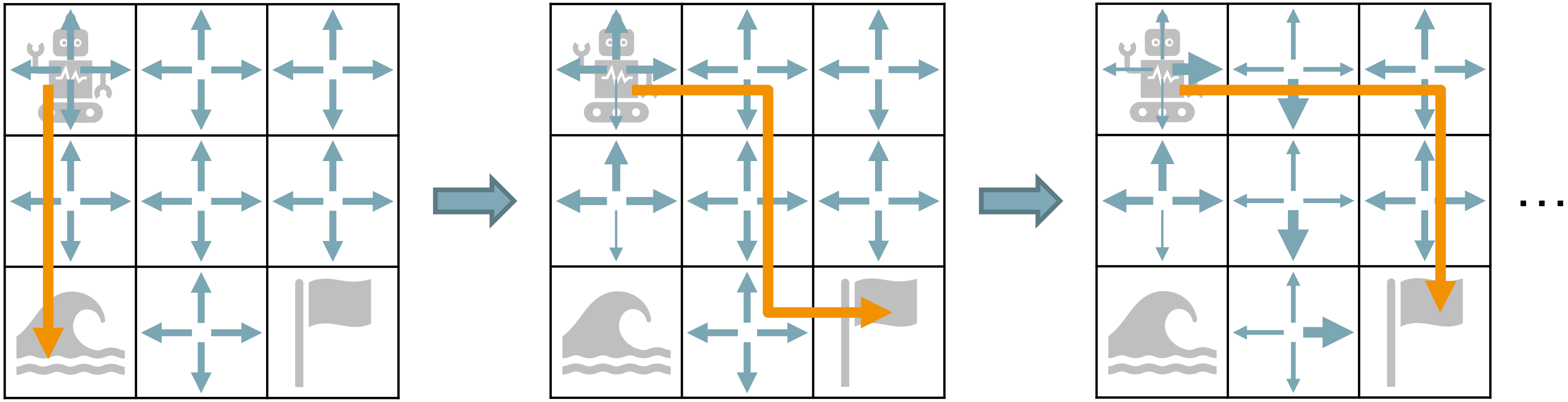
Simplified example (with wave = robot drowns, -100 reward and episode finished)



-4	-3	-2
-3	-2	-1
(-100)	-1	(0)

Basics of RL

Policy gradients iteratively improve an initially bad policy, thus eventually obtaining an optimal policy



Kahoot!