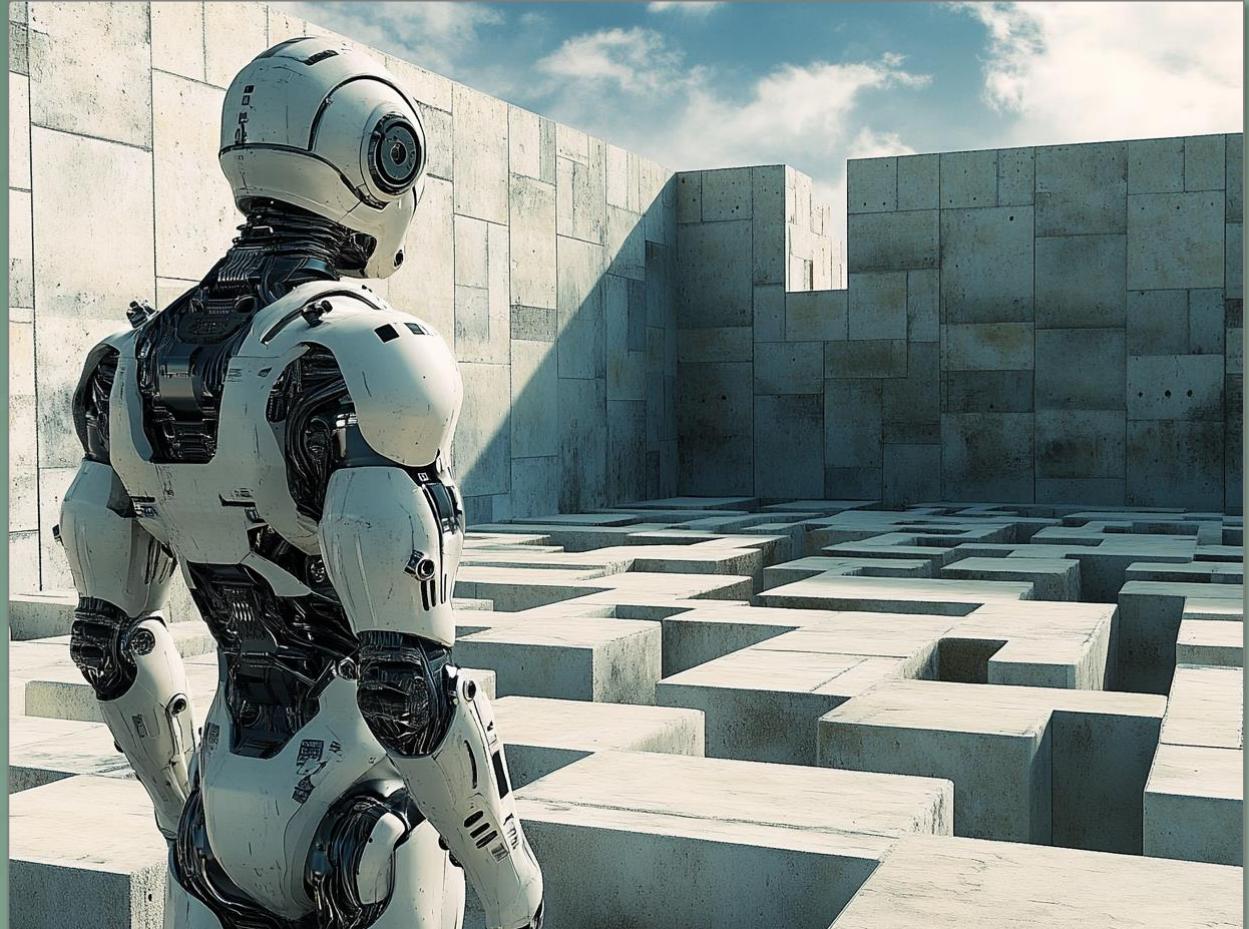




Reinforcement Learning Q-Learning

Dr. Harald Stein
Aug 2024



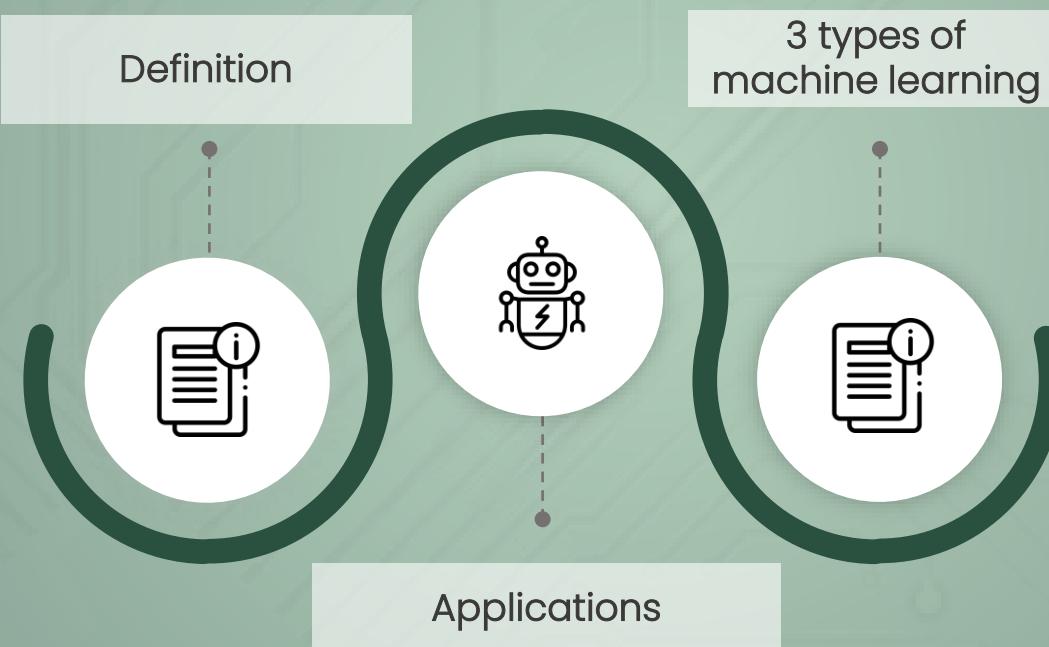
Agenda

- **What is Reinforcement Learning (RL)?**
- **Basic concepts of RL**
- **Temporal difference learning**
- **Q-learning**
- **Code example: Taxi driver with Q-learning**



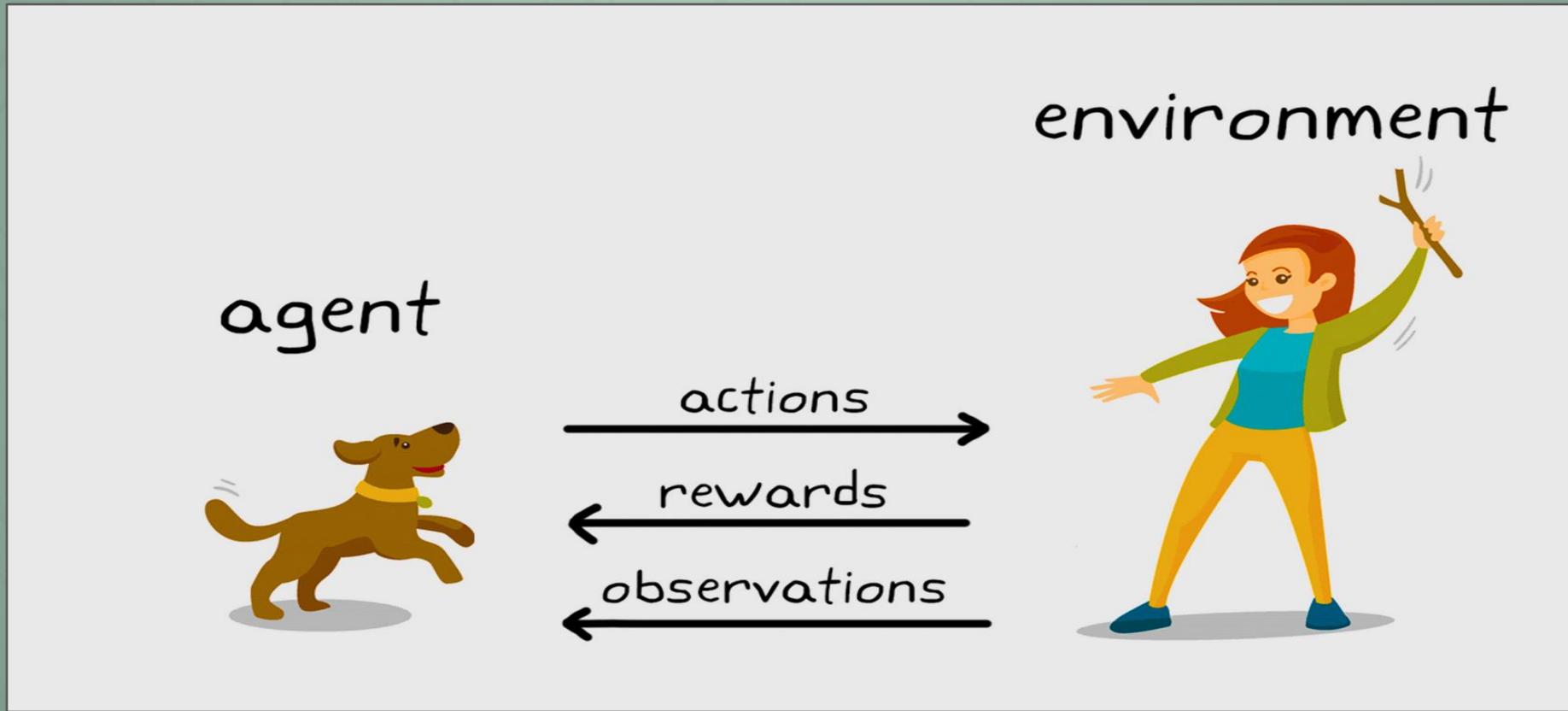
What is Reinforcement Learning?

... and what is it good for?



What is Reinforcement Learning?

It is type of machine learning where agents learn how to behave in an environment



What is Reinforcement Learning?

It is type of machine learning where agents learn how to behave in an environment

- takes actions based on observations
- aiming to achieve best possible outcome

Agent

Goal Setting

- Each action the agent takes impacts its environment
- leading to new situations (states)

Environment Interaction

Decision Strategy

- After each action, agent receives feedback in form of rewards/penalties
- indicating the quality of its decision.

Feedback Mechanism

- Primary objective of agent is to maximize cumulative reward over a period
- refining its actions and strategies over time

- Through experience, agent develops policy, a guideline on which action to choose in a given situation
- to achieve its long-term reward maximization goal.

Challenges Faced

- Ongoing challenge in reinforcement learning is the balance between exploration (trying new actions) exploitation (sticking with known beneficial actions).

Applications of Reinforcement Learning

Examples are:

- **Game Playing:** Classic games like Tic-Tac-Toe, where the state and action spaces are manageable without deep learning.
- **Multi-Armed Bandit Problems:** Used in areas such as online advertising, clinical trials, and A/B testing to make decisions under uncertainty.
- **Grid World Navigation:** Simple agent navigation tasks in a grid-like environment to reach a goal.
- **Customer Service:** Automated decision-making for handling customer queries or complaints in a predefined and rule-based environment.
- **Dynamic Pricing:** Adjusting prices of products or services in real-time based on demand, competitor prices, or other factors.



3 types of machine learning

Unsupervised Learning

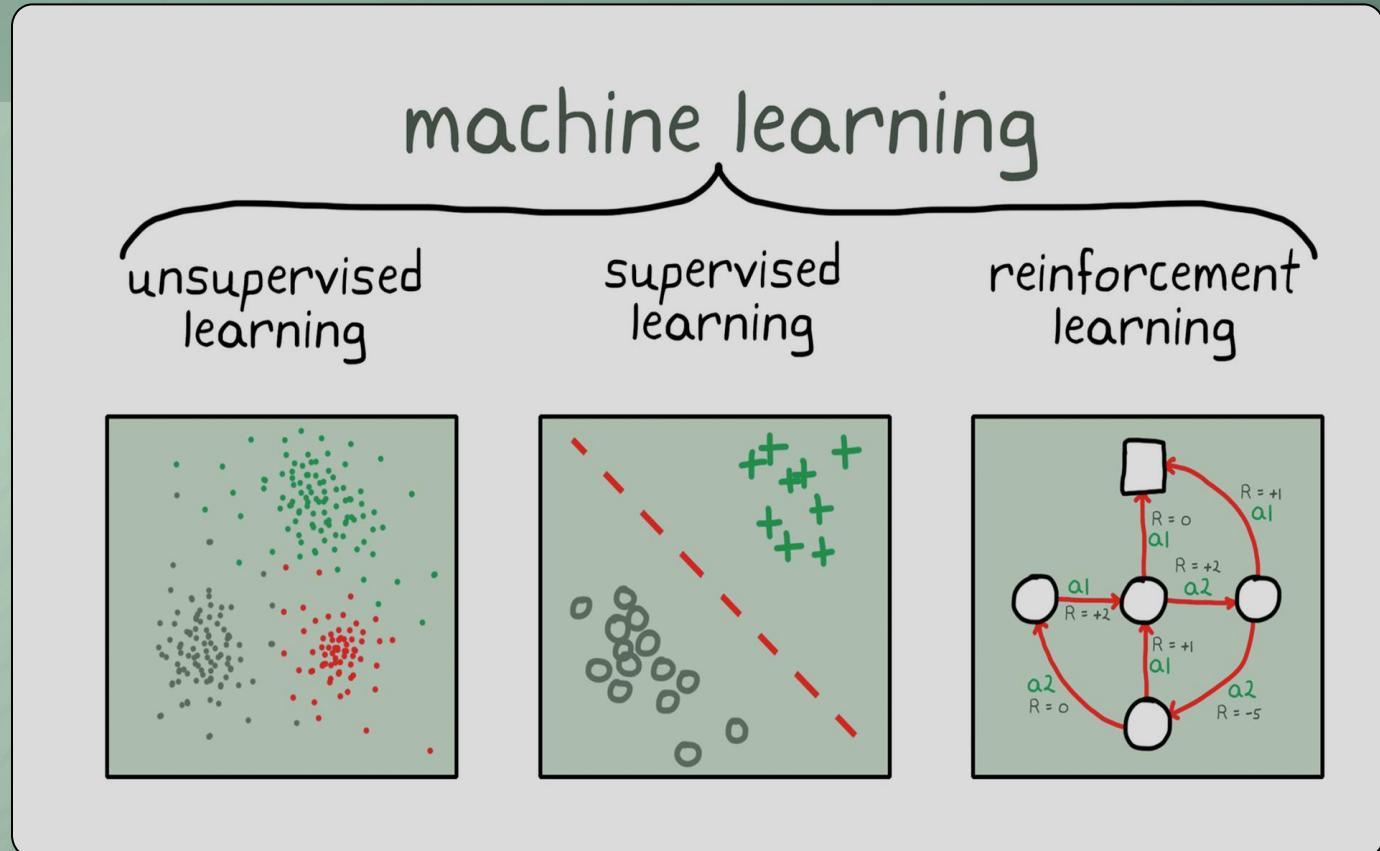
- Type of machine learning that deals with unlabeled data and aims to identify underlying patterns or structures.
- The model learns to represent the data without explicit guidance.

Supervised Learning

- Machine learning paradigm where a model is trained on labeled data to make predictions or classifications.
- The model learns a function that maps input features to output labels.

Reinforcement Learning

- Represents the policy or value function, guiding the agent's decisions.
- These methods play a crucial role in capturing semantic meaning and relationships between words and phrases.



3 types of machine learning

Focus

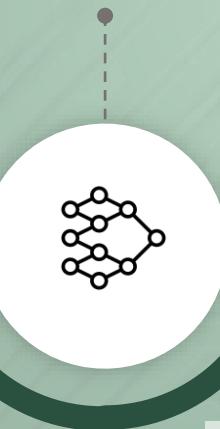
	1 Unsupervised Learning	2 Supervised Learning	3 Reinforcement Learning
INPUT DATA	Unlabeled: no „right answer“ specified	Labeled: the „right answer“ is included	Data are not part of the input, they are collected through trial and error
GENERAL TASKS	Discovery of clusters, patterns, relationships	Classification, regression	Explorative controlling: Solution of reward-based problems by exploration and exploitation
SOLUTION	Finds similarities and differences in input data	Maps input to output	Finds which states and actions would maximize the total cumulative reward of the agent
EXAMPLES	Customer segmentation, product recommendation	Image detection, stock market prediction	Game playing, robotic vacuum cleaners
FEEDBACK	No	Yes. The correct set of actions is provided.	Yes, through rewards and punishments (positive and negative rewards)

Basic concepts of RL

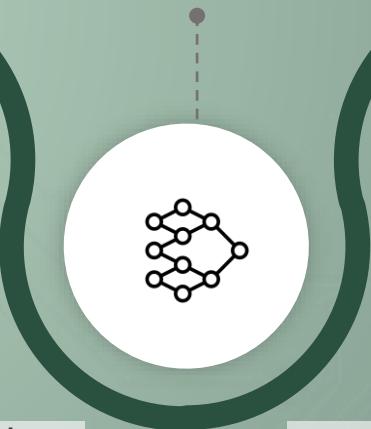
Agent-Environment interaction



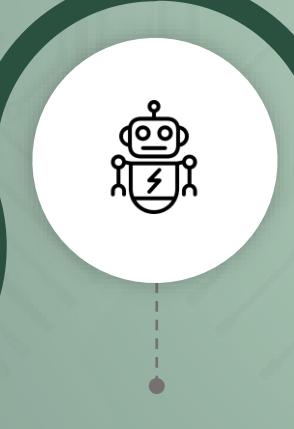
Reward and discounting



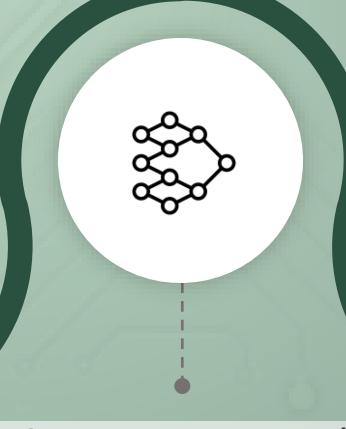
Tasks: Episodic vs. continuous



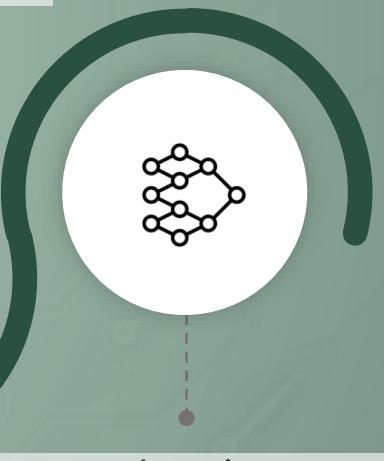
Reinforcement Learning process & loop



States space and action space

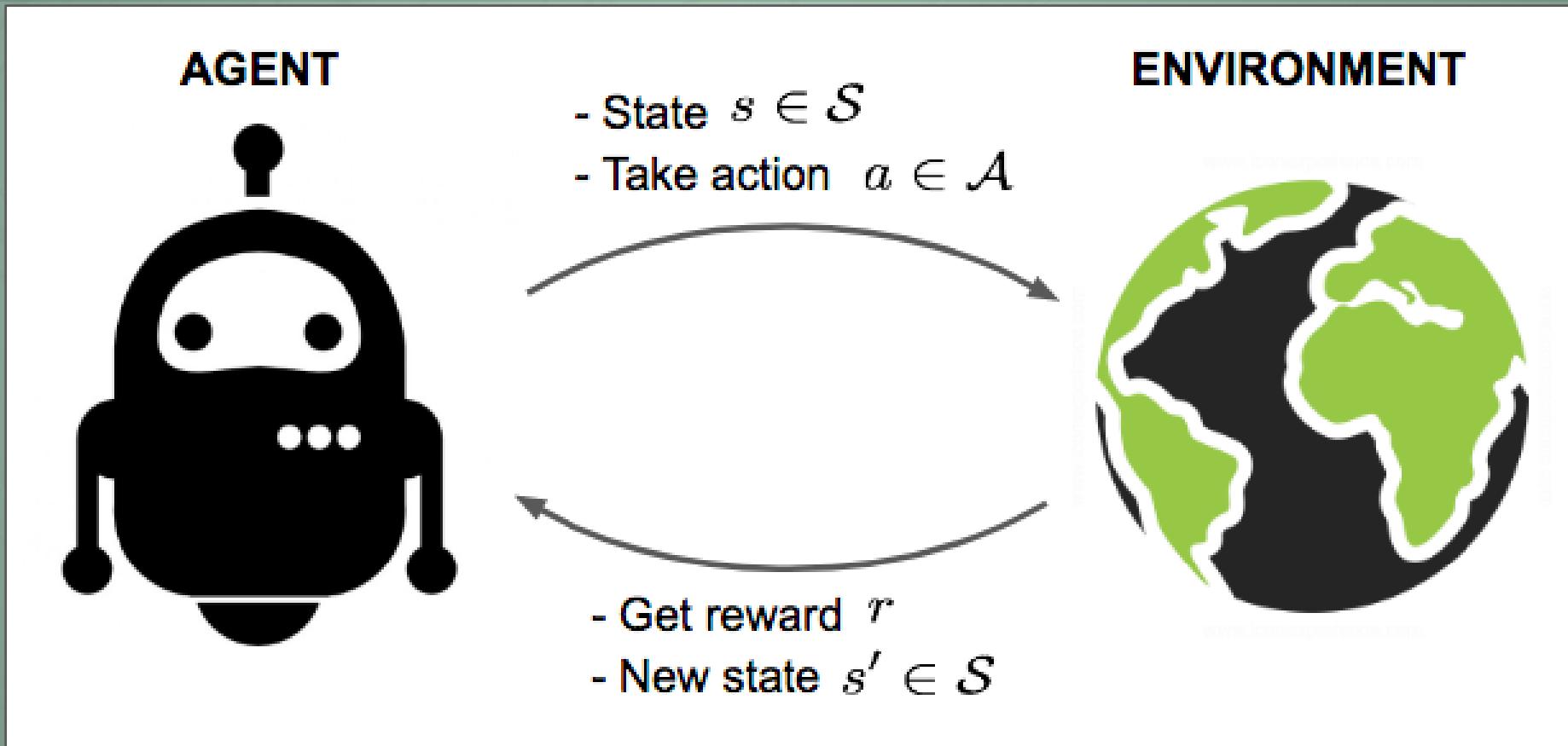


Exploration-Exploitation tradeoff



Agent-environment interaction

... provides a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision-maker



Reinforcement Learning process

... is a Markov decision process that provides mathematical framework for modeling decision-making when outcomes are partly random, partly under decision-maker's control

- 5-tuple of (S, A, T, R, γ) with
 - S the state space
 - A the action space
 - T the transition function
 - R the reward function
 - γ the discount factor

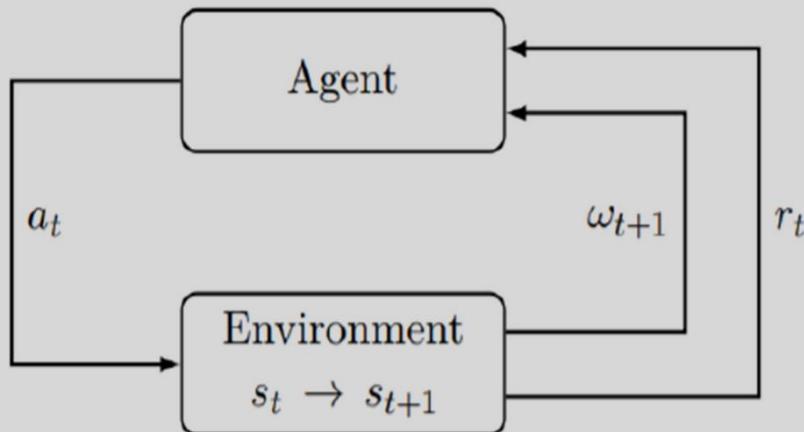


Figure 1: The agent-environment interaction [1]

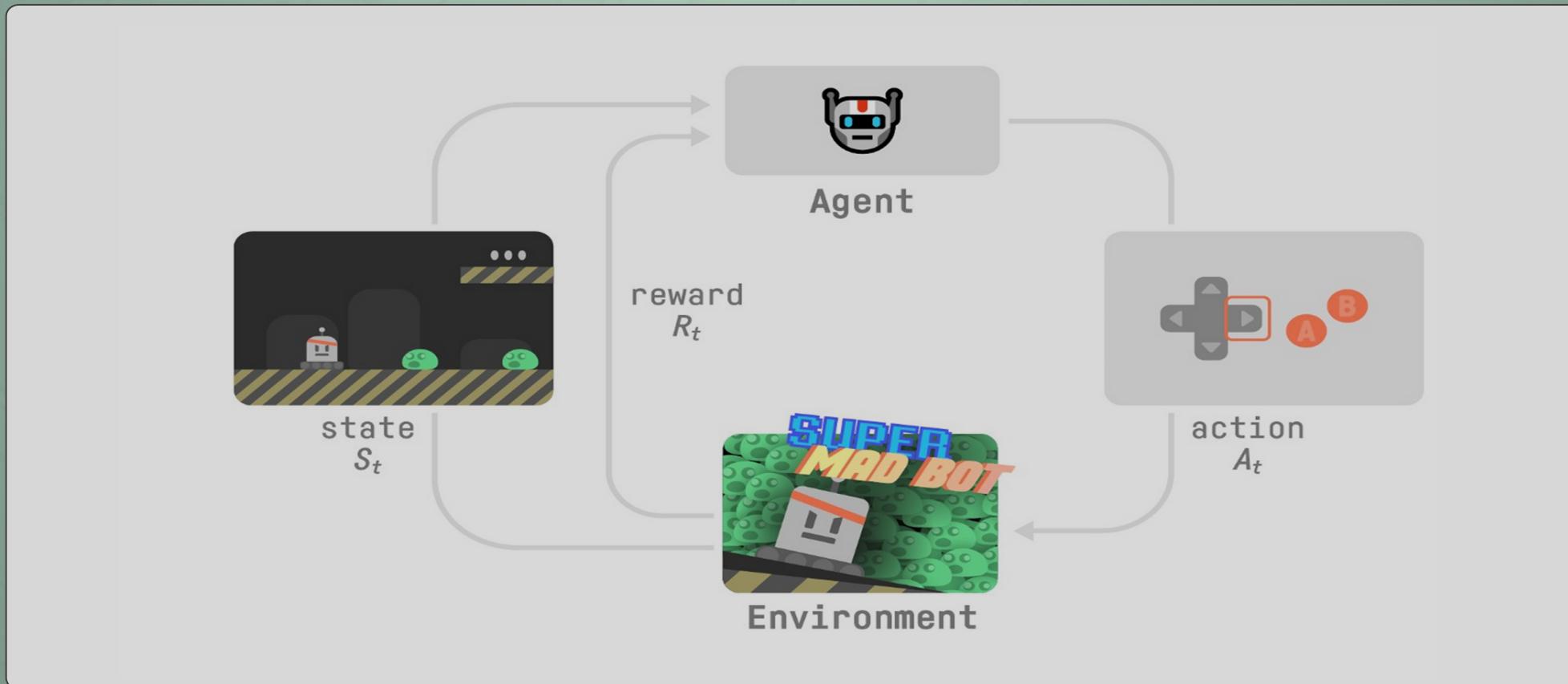
- Theoretical Framework of Reinforcement Learning
- The agent wants to maximize his rewards

Markov Property:

implies that our agent needs only current state to decide what action to take without the history of all states and actions they took before.

Example: Video game

... for Markov Decision Process in RL context



Reinforcement Learning loop

... outputs a sequence of state, action, reward and next state.



The agent's goal:

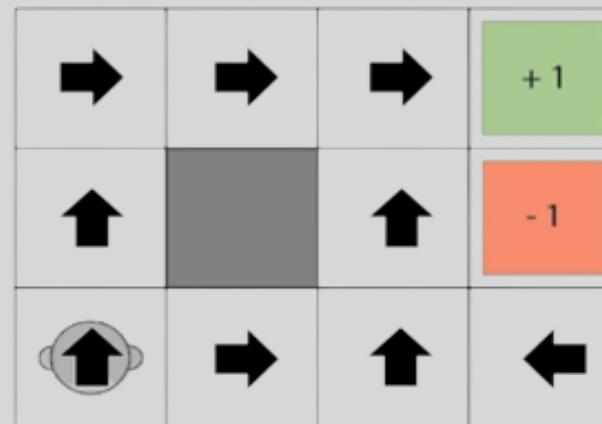
... to maximize its cumulative reward called the expected return.

The reward hypothesis

... is that all goals can be described as the maximization of the expected return (expected cumulative reward)

Sum of Future Rewards

- Discount factor $\gamma = 0.9$
- Return at state = (0, 2) is 1
- Return at state = (0, 1) is 0.9
- Return at state = (0, 0) is 0.81



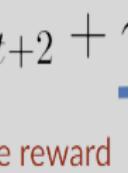
Rewards and discounting

Reward is fundamental in RL because it's the only feedback for the agent. Thanks to it, our agent knows if the action taken was good or not

To discount the rewards, we proceed like this:

- Discount rate gamma between 0 and 1 is defined
- Most of the time between 0.95 and 0.99.
(The larger the gamma, the smaller the discount. This means our agent cares more about the long-term reward)
- Each reward will be discounted by gamma to the exponent of the time step.
- As time step increases, agent approaches final round, if task is episodic. Then sum of discounts vanishes.

$$R(\tau) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots$$

 Return: cumulative reward
 Gamma: discount rate

Trajectory (read Tau)
Sequence of states and actions

$$R(\tau) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Observations/States Space

... are the information our agent gets from the environment

- **Board game (complete information):**
position of game pieces
- **Card game (incomplete information):**
visible cards
- **Video game (incomplete information):**
frame / screenshot
- **Trading agent (incomplete information):**
value of certain stock, news, etc.

State
(complete information):
complete description of
the state of the world



Observation
(incomplete information):
Partial description of the
state of the world



Action Space

... is the set of all possible actions in an environment

- **Board game, card game:**
possibilities to move game pieces, play cards
- **video game:**
left, right, accelerate, break, don't push any button
- **trading agent:**
buy, sell, don't change portfolio

Discrete:

Finite number of
possible actions



Continuous:

Infinite number of
possible actions



Task

... is an instance of a Reinforcement Learning problem. Two types: episodic and continuing

Episodic task

- There is starting point and ending point (a terminal state).
- This creates an episode: list of States, Actions, Rewards, and new States.

Continuous task

- Agent must learn how to choose the best actions
- and simultaneously interact with the environment.

Episodic:

Starting point and an ending point (a terminal state)



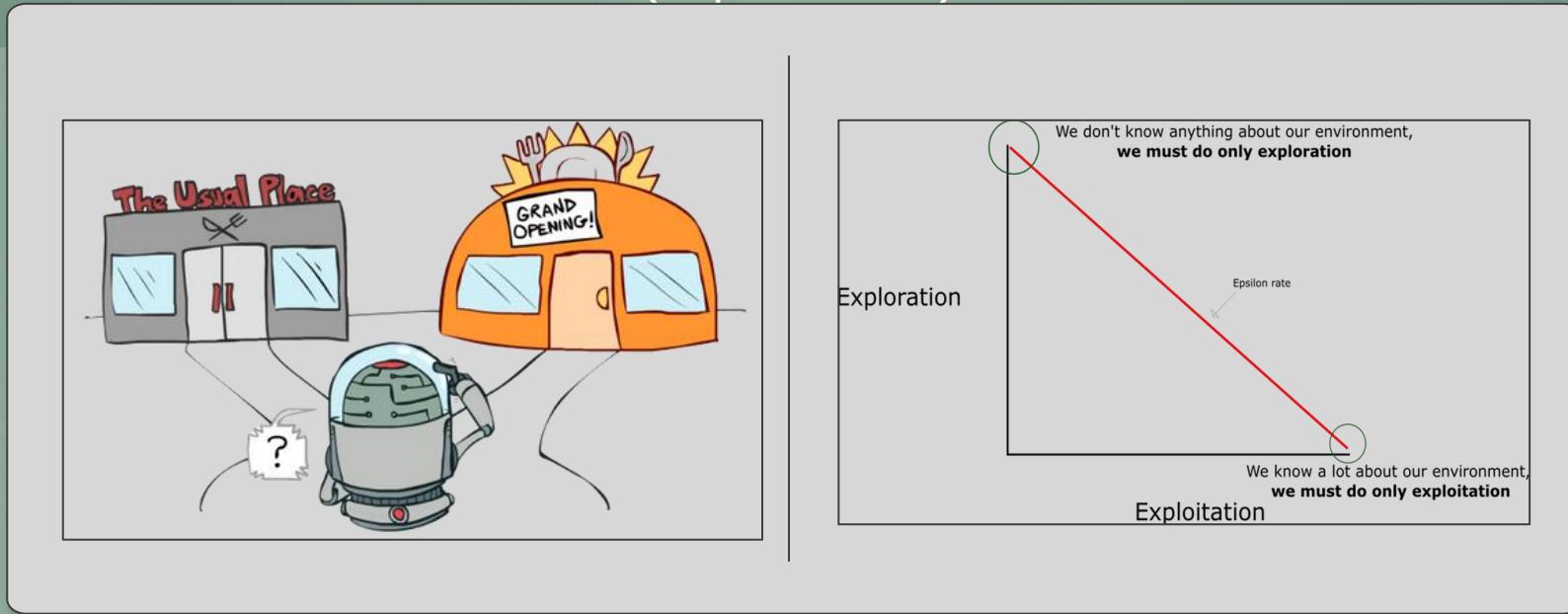
Continuing:

Task that continue forever (no terminal state)



Exploration vs Exploitation

... is the dilemma where an agent must decide between trying new actions (exploration) or sticking with known beneficial actions (exploitation).



Exploration:

Trying new actions to discover their outcomes.

Exploitation:

Choosing actions known to yield good rewards.

Dilemma:

Gaining knowledge (exploration) \Leftrightarrow maximizing rewards with current knowledge (exploitation).

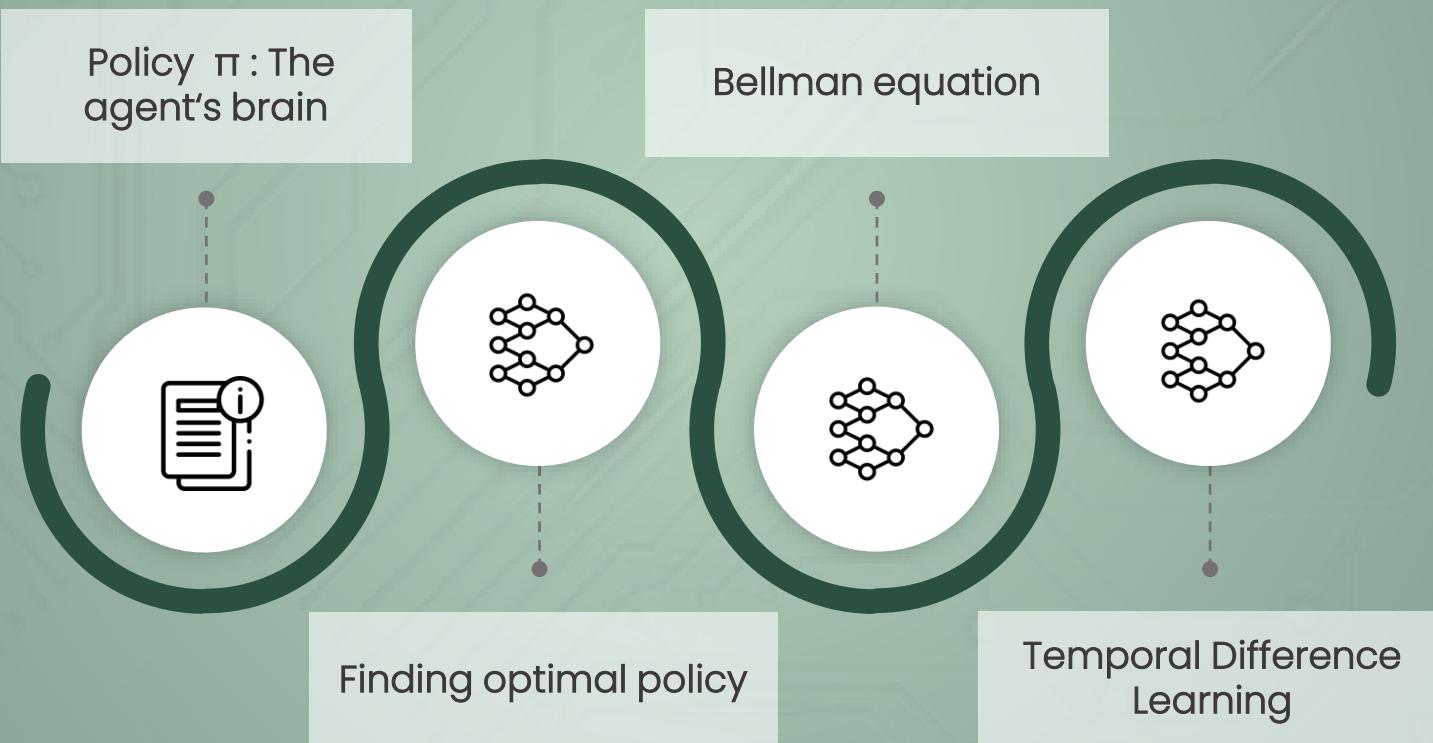
Importance:

Essential for an agent to adapt to changing environments and avoid local optima.

Strategies:

Techniques like ϵ -greedy help manage this trade-off.

Theoretical foundations of Temporal Difference Learning



Policy π : the agent's brain

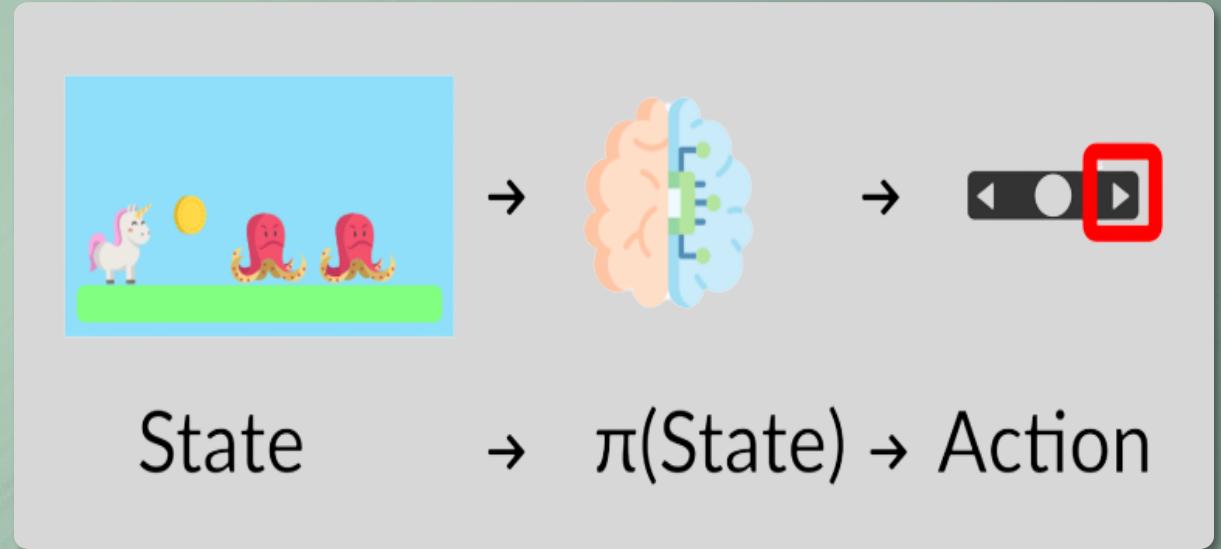
Policy is a strategy used by agent to decide which action to take in a given state.
Represented as $\pi(a|s)$, which denotes the probability of taking action a given state s .

Role of the Policy:

- Acts as the “brain” of the agent, guiding its actions as it interacts with the environment.
- Aims to maximize the cumulative reward over time.

Types of Policies:

- Deterministic Policy: A policy that always outputs the same action for a given state.
- Stochastic Policy: A policy that outputs a probability distribution over actions, allowing for exploration.



Finding optimal policy

... by iterating between policy and value

Policy Iteration:

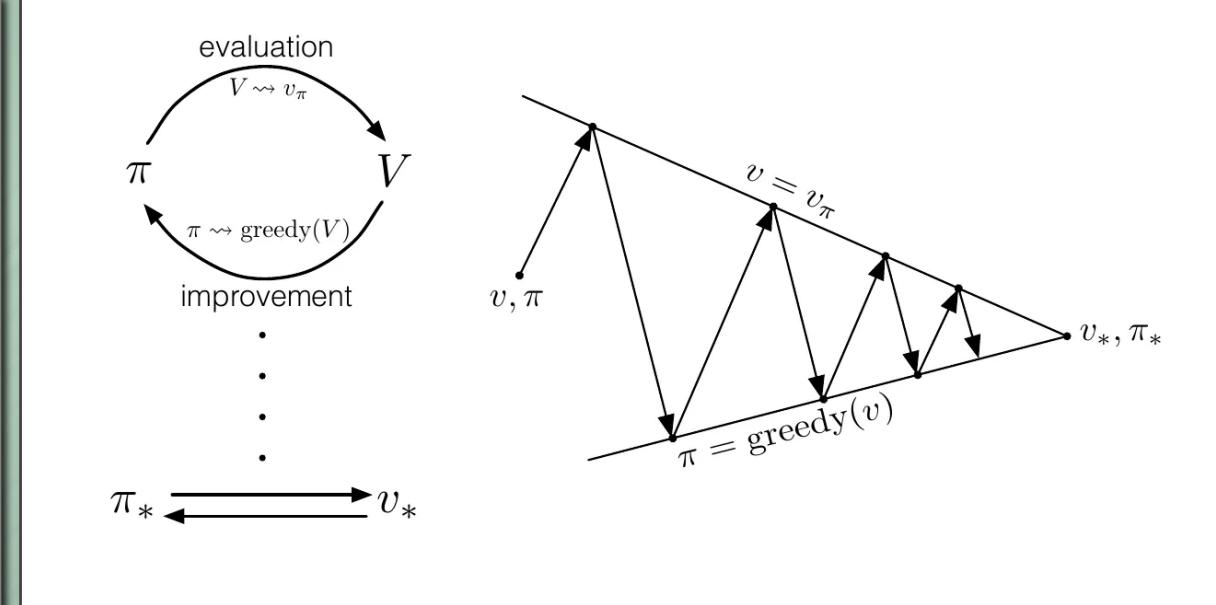
- Evaluation: Circle showing $V \approx \pi$ (value function approximates policy).
- Improvement: Policy is refined using $\pi \approx \text{greedy}(V)$

Convergence:

- Zig-zag arrows: Iterative steps from initial (v, π) to optimal (v^*, π^*)

Optimal Policy and Value:

- Process concludes when reaching optimal policy π^* and corresponding value function v^* , represented at far right of image.



Bellman equation

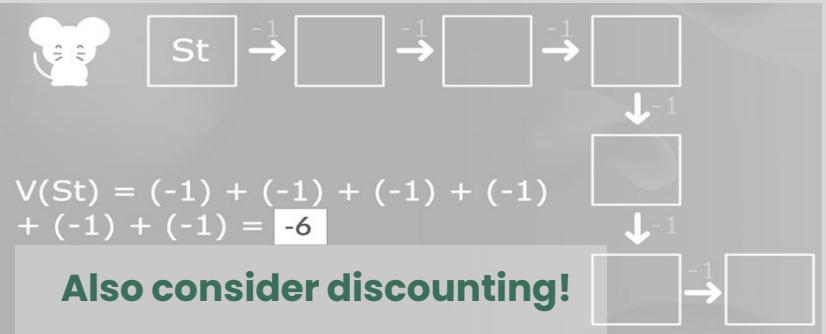
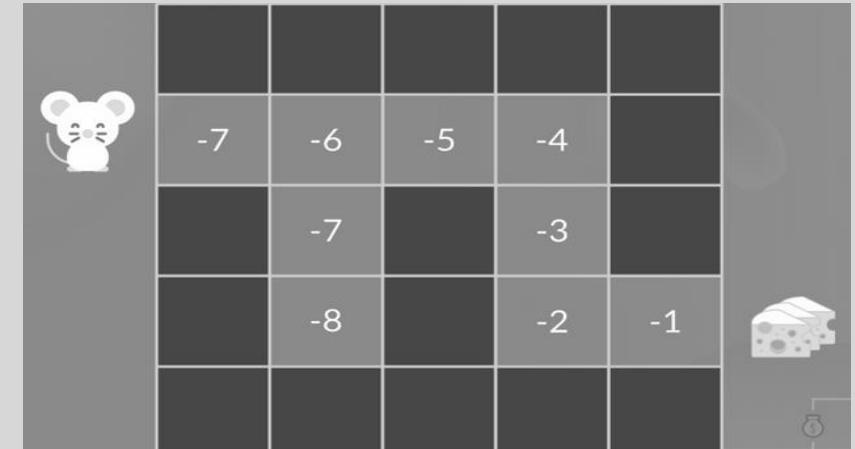
... is under the hood of any reinforcement learning approach.

The idea:

- instead of calculating each value as the sum of the expected return, which is a long process
- we calculate the value as the sum of immediate reward + the discounted value of the state that follows.

Role in Value-Based Methods:

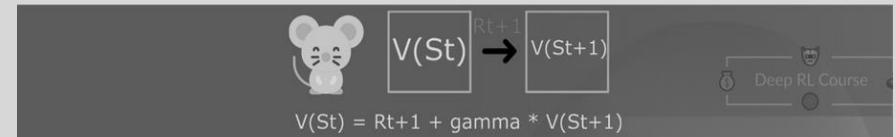
- Bellman Equation is used to update the value functions.
- Leads to optimal policy that maximizes cumulative reward.



$$V_{\pi}(s) = \mathbf{E}_{\pi}[R_{t+1} + \gamma * V_{\pi}(S_{t+1}) | S_t = s]$$

Value of state s Expected value of immediate reward + the discounted value of next_state If the agent starts at state s

And uses the policy to choose its actions for all time steps



Temporal Difference approach

... Updates value estimates based on a sample of the subsequent state, i.e. at each step instead of at the end of the episode

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

New value
of state t

Former
estimation of
value of state
t

Learning
Rate

Reward
Discounted value of next
state

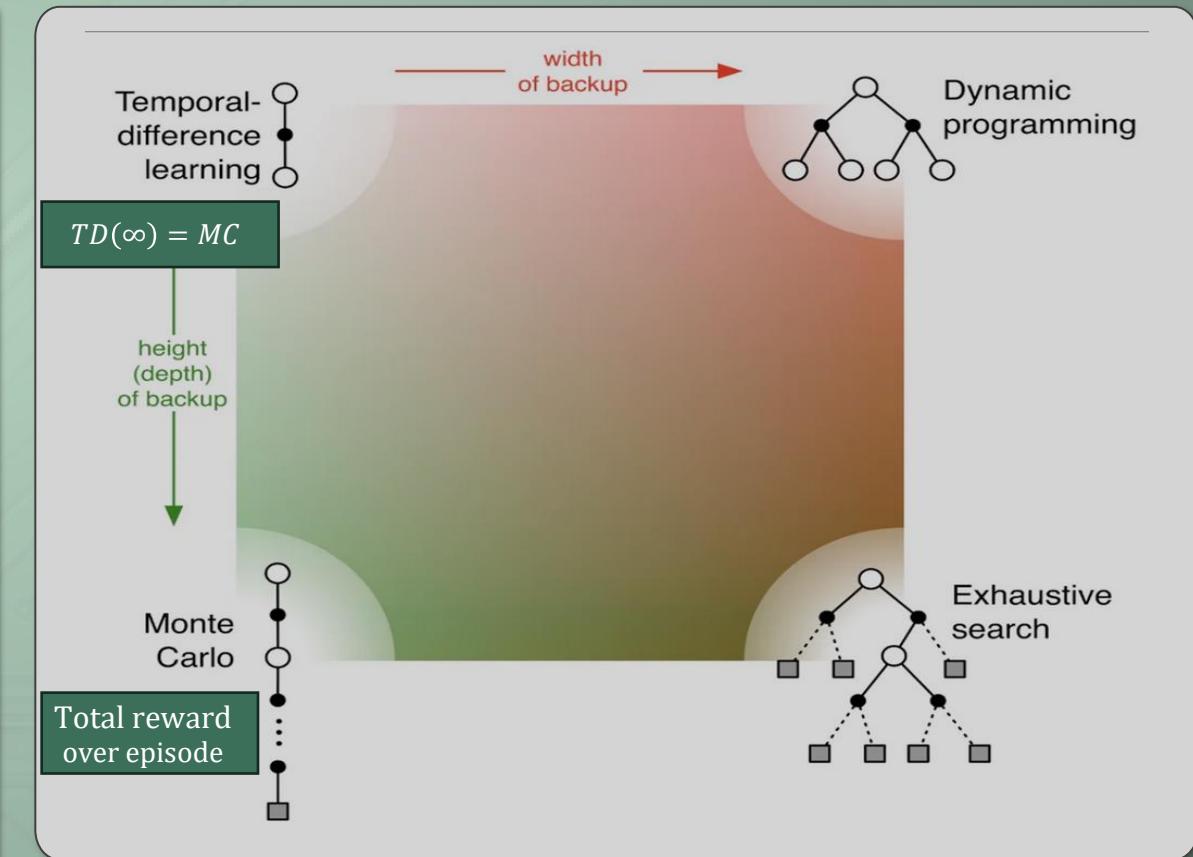
TD Target

Temporal Difference learning TD(0)

... learns directly from episodes of experience without a model of the environment. Updates value estimates based on a sample of the subsequent state.

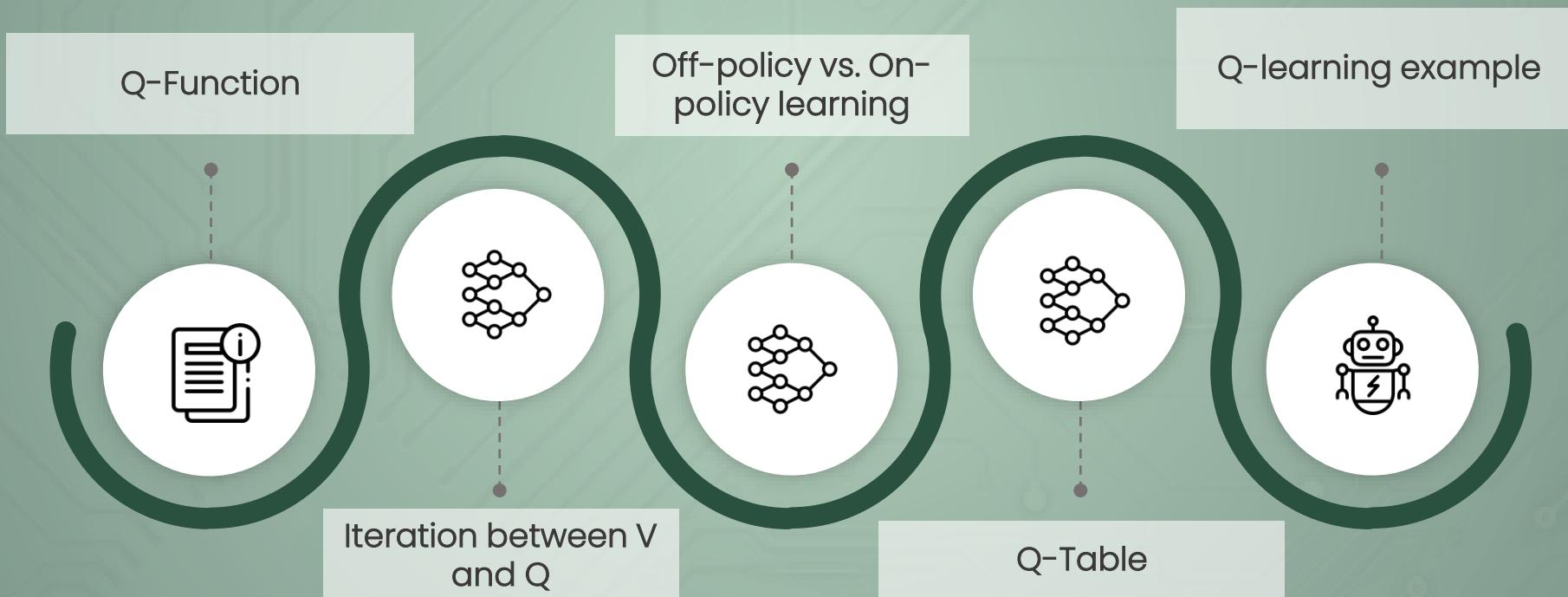
Comparison to other controlling approaches:

- Exhaustive search:
 - evaluates all possible paths and strategies, which can be computationally expensive or infeasible.
 - TD Learning incrementally adjusts value estimates and does not require exploring all possibilities.
- Dynamic Programming:
 - requires a complete and accurate model of the environment.
 - TD Learning does not require a model and learns from incomplete and imperfect episodes by learning online after every step
- Monte Carlo:
 - updates only occur after an entire episode has completed.
 - TD Learning updates values after each time step, leading to faster learning.
 - TD is more suitable for continuing (non-episodic) tasks.



Q-Learning

... is an off-policy value-based method that uses a TD approach to train its action-value function



Q-Function

... expected return of taking action a in state s . Captures immediate rewards and potential future gains. Q-Values: updated iteratively based on agent's experiences for deriving optimal policy

$$\text{New } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \operatorname{argmax}_{a'} Q(s', a') - Q(s, a)]$$

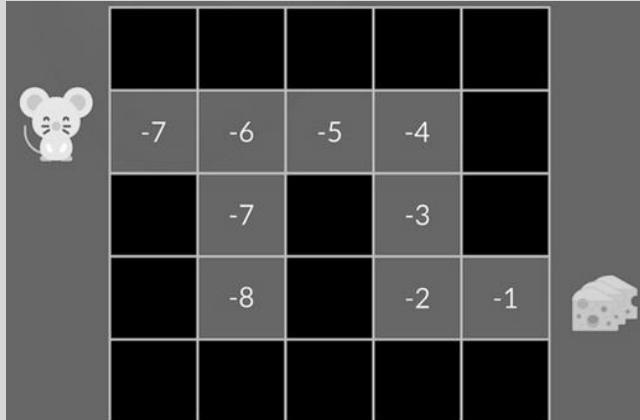
- █ New Q Value for that state and the action
- █ Learning Rate
- █ Reward for taking that action at that state
- █ Current Q Values
- █ Maximum expected future reward given the new state (s') and all possible actions at that new state choosing the best available action regardless of policy function
- █ Discount Rate

Iteration between V and Q

... i.e. between value of state and value of state-action

State-Value-Function

Calculate the value of state



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

Value of state s

Expected return

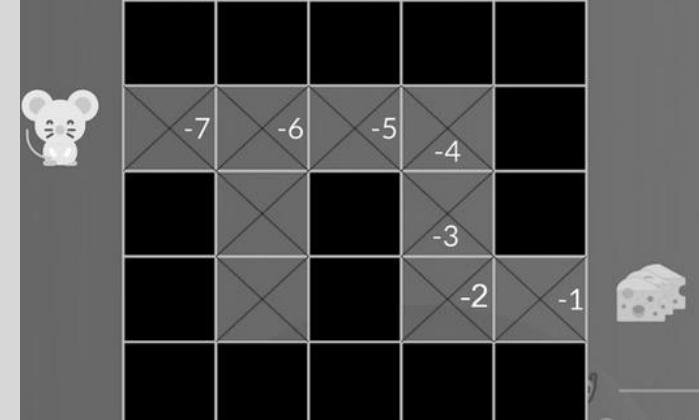
If the agent starts at state s

And uses the policy to choose its actions for all time steps

For each state, the state-value function outputs the expected return if the agent starts in that state and then follows the policy forever after.

Action-Value-Function

Calculate the value of state action pair



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

Value of state-action pair s,a

Expected return

If the agent starts at state s

and chooses action a

And then uses the policy to choose its actions for all time steps

For each state and action, the action-value function outputs the expected return if the agent starts in that state and takes the action and then follows the policy forever after.

Off-policy vs. On-policy learning

On-Policy: Algorithm learns, evaluates policy that it follows to make decisions → SARSA

Off-Policy: Algorithm learns optimal policy independently of policy it follows → Q-Learning.

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:
 Initialize S
 Loop for each step of episode:
 Choose A from S using policy derived from Q (e.g., ε -greedy)
 Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$$

$$S \leftarrow S'$$

 until S is terminal

Off policy

- Using a different policy for acting and for updating
- “Playing the best move regarding short-term incentive and updating policy, if improvement took place”
- More explorative, less stable

Focus on
Q-Learning

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:
 Initialize S
 Choose A from S using policy derived from Q (e.g., ε -greedy)
 Loop for each step of episode:
 Take action A , observe R, S'
 Choose A' from S' using policy derived from Q (e.g., ε -greedy)

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R - \gamma Q(S', A') - Q(S, A)]$$

$$S \leftarrow S'; A \leftarrow A'$$

 until S is terminal

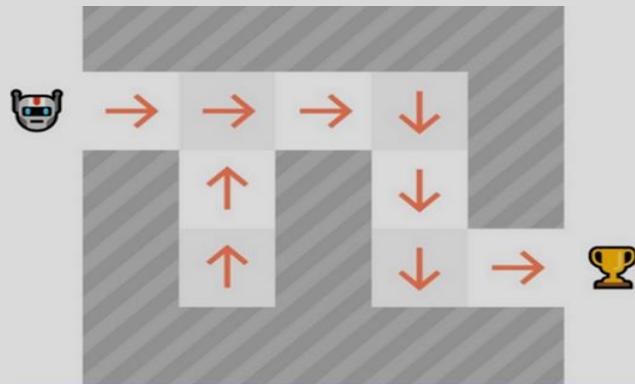
On policy

- Using a same policy for acting and for updating
- “Playing the best available move from existing policy”
- Less explorative, more stable

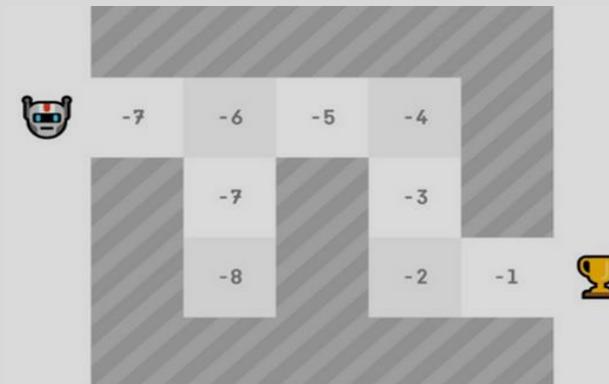
Off-policy vs. On-policy learning

Policy vs. value based methods

Policy Based Method: Train the agent to learn which action to take given a state



Value-Based Methods : Train the agent to learn which state is more valuable and take the action that leads to it.



Focus on
Q-Learning

Difference:

- In policy-based training, the optimal policy (denoted π^*) is found by training the policy directly.
- In value-based training, finding an optimal value function (denoted Q^* or V^* , we'll study the difference below) leads to having an optimal policy.

Cliff Walking Example

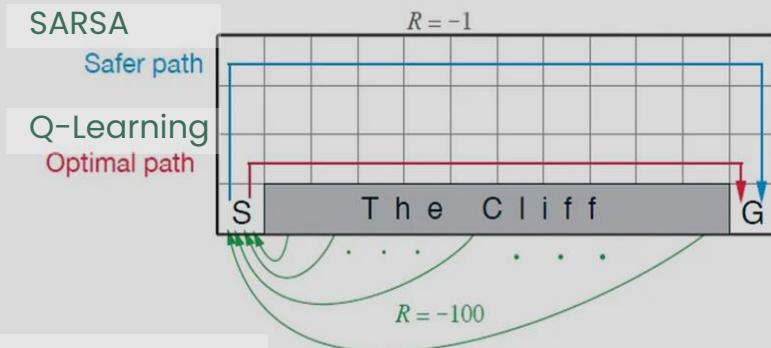
... is a standard gridworld environment used to illustrate the difference between on-policy and off-policy methods like SARSA and Q-learning

- Agent can take actions to move in one of four directions: up, down, left, right.
- Moving into "Cliff" state incurs large negative reward (e.g., -100), sends agent back to "Start" state.
- Each other move typically has small negative reward (e.g., -1), incentivizing agent to reach goal quickly

Comparing SARSA and Q-Learning:

- SARSA:
 - agent tends to take longer but safer route
 - avoiding edge adjacent to cliff, because it considers future action which might be exploratory and lead it into the cliff.
- Q-Learning:
 - agent usually learns optimal policy to skirt dangerously close to cliff for shortest path to goal
 - but it may occasionally fall into cliff during exploration due to greedy nature of its learning

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
做人	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	+10
骷髅	骷髅	骷髅	骷髅	骷髅	骷髅	骷髅	骷髅	骷髅	骷髅	骷髅	骷髅	骷髅	旗帜



Focus on
Q-Learning

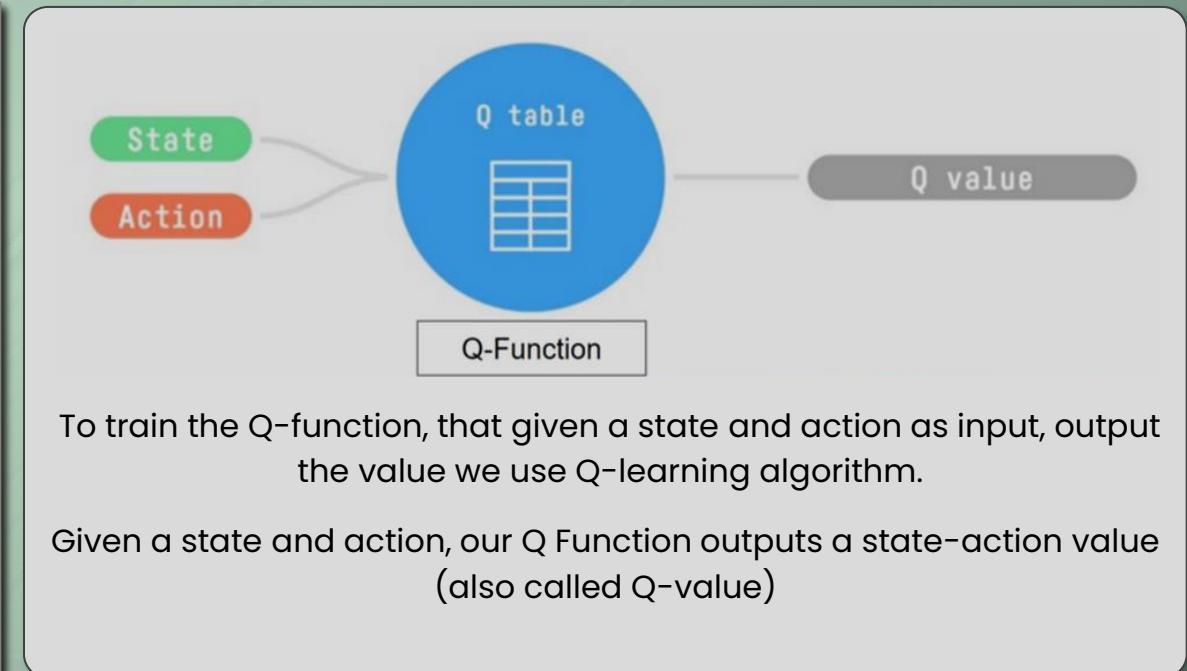
Implementing Q-Function with Q-Table

Implementing the Q-Function:

- The Q-Table essentially implements the Q-Function.
- The agent consults the Q-Table to decide which action to take in a given state, usually choosing the action with the highest Q-Value.
- Over time, with enough exploration and consistent updates, the Q-Table converges to the optimal Q-Function, $Q^*(s, a)$.

Be aware of exploration-exploitation dilemma:

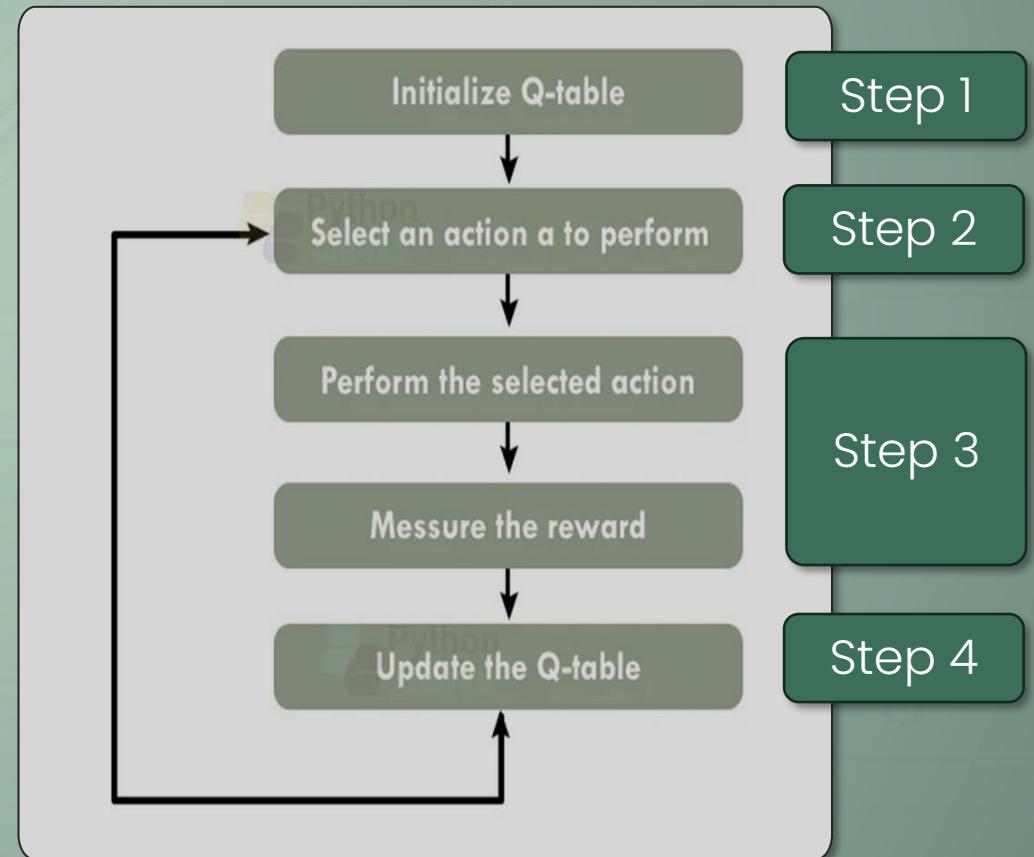
- Ensure adequate exploration of the state-action space to find the optimal policy
- i.e. do not always use Q-Table for determining next step, but to it randomly in small share of cases ε



Q-Table loop

... or how the Q-Table is trained

- **Initialize Q-table:**
The process begins with initializing the Q-table.
- **Select an action "a" to perform:**
After initialization, agent selects an action based on current state and Q-table. Exception: With small prob ϵ random action is selected
- **Perform the selected action:**
Agent then performs chosen action in environment.
- **Measure the reward:**
After performing action, agent receives and measures reward based on outcome.
- **Update the Q-table:**
Using obtained reward and new state, Q-table is updated to better estimate action values.
- **Loop Structure:**
Process from selecting action to updating Q-table is depicted as loop, indicating repetition until termination condition (not shown in the image).

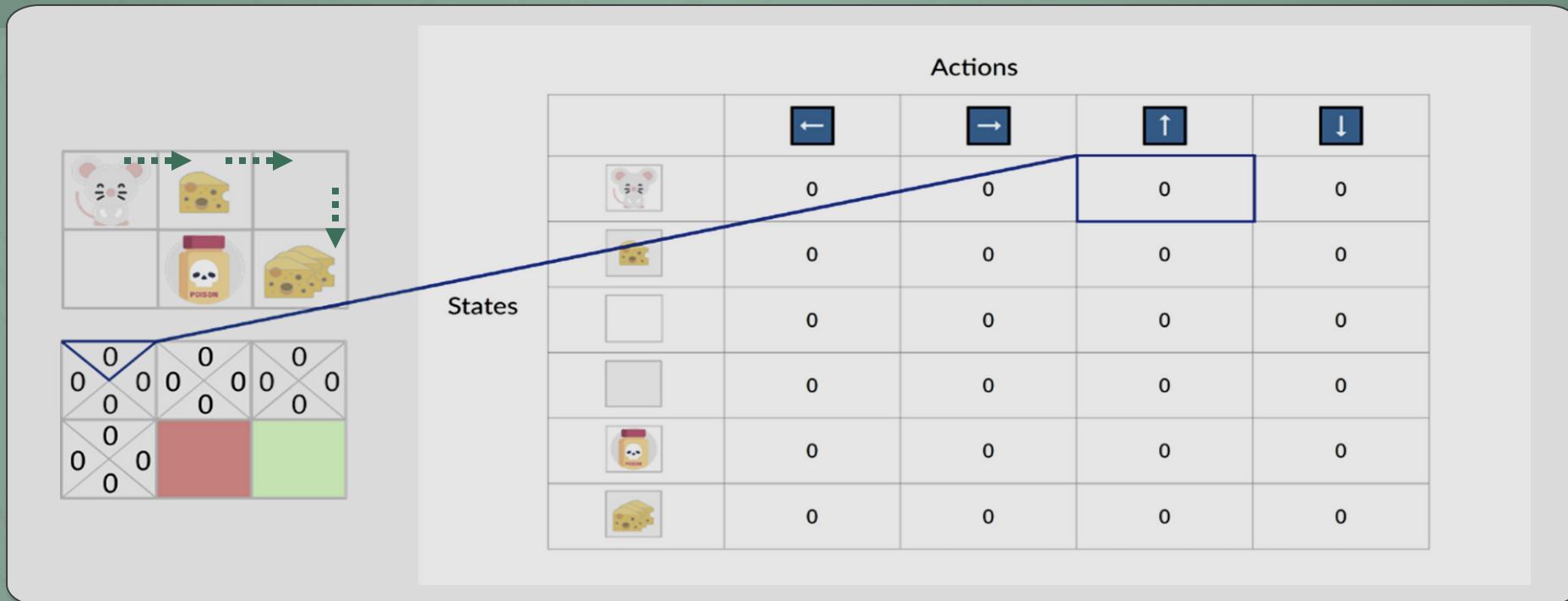


Q-learning example

... is a tabular form for storing Q-Values corresponding to state-action pairs.

How can the mouse reach both cheese fields without running into the poison field?

Step 1



Structure of Q-Table:

- Rows represent all possible states.
- Columns represent all possible actions.
- Entries in the table store the Q-Values, $Q(s, a)$.

Learning with Q-Table:

- Agents select actions based on Q-Values in table, typically choosing action with highest Q-Value in current state.
- After taking an action and observing the reward and next state, the Q-Value is updated

Q-learning example

... or how the Q-Table is trained

Step 2

Choose action using ϵ -greedy policy, i.e. choose action A_t using policy derived from Q :

- ϵ -greedy
- Exploitation:
selects optimal ("greedy") action
 - Exploration:
selects random action

Step 3

Take action A_t and observe:

- reward R_t
- State S_t

Step 4

Update $Q(S_t, A_t)$:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

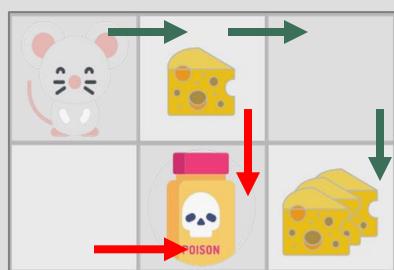
New value of state t Former estimation of value of state t Learning Rate Discounted value of next state
TD Target

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

New Q-value estimation Former Q-value estimation Learning Rate Immediate Reward Discounted Estimate optimal Q-value of next state
TD Target TD Error

Q-learning example

The trained Q-Table tells the mouse where to go and where not to go.



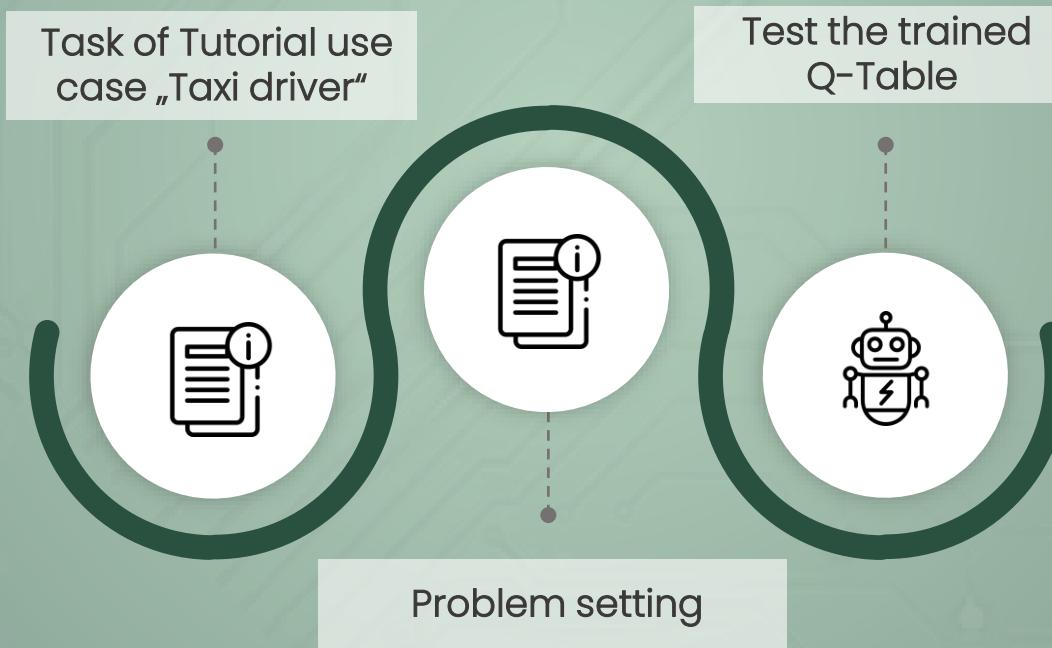
	⬅	➡	↑	↓
Mouse	0	0	0	0
Cheese	0	0	0	0
Empty	0	0	0	0
Empty	0	0	0	0
Poison	0	0	0	0
Empty	0	0	0	0

Training

	⬅	➡	↑	↓
Mouse	0	10.8	0	0
Cheese	0	9.9	0	-10
Empty	0	0	0	10
Empty	0	-10	0	0
Poison	0	0	0	0
Empty	0	0	0	0

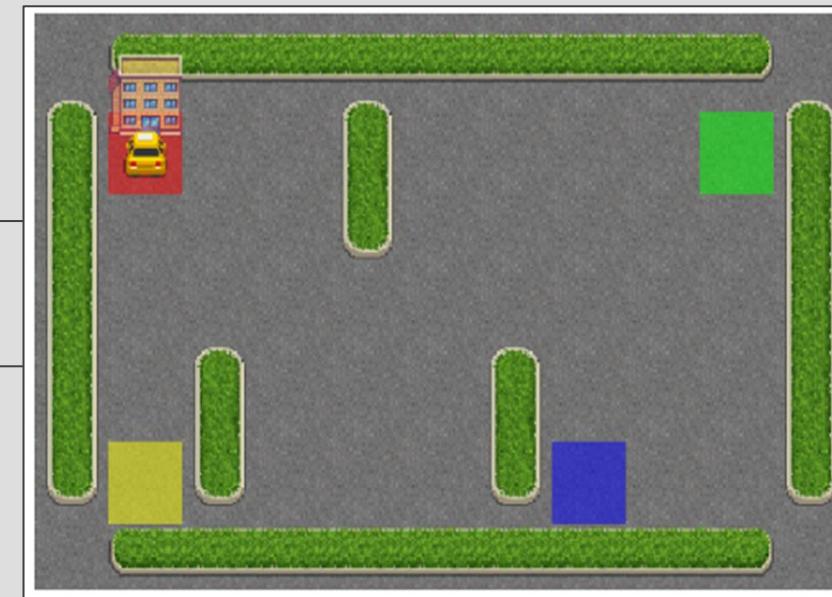
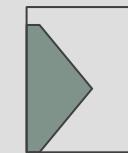
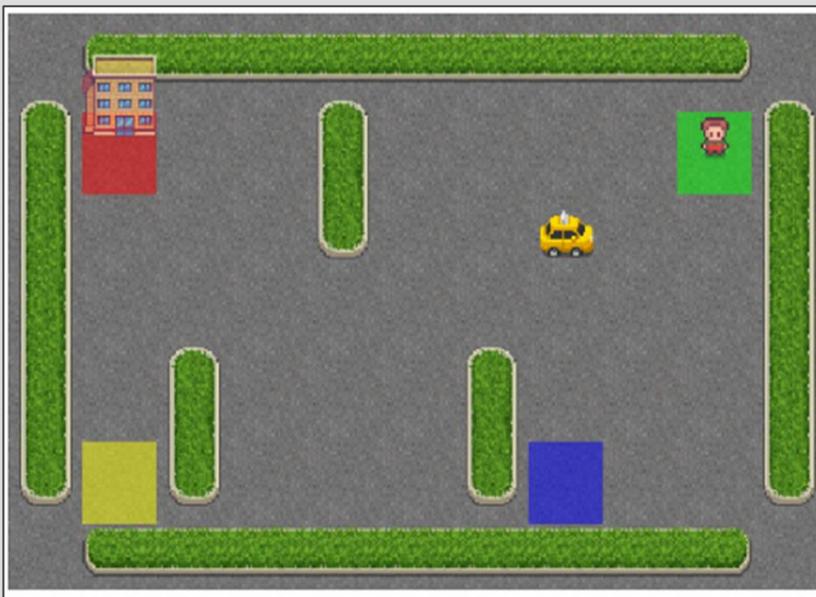
Use Case: Taxi driver

...a taxi should pick up passengers and drop them at their destination on a parking lot



Task of Tutorial use case „Taxi driver“

...a taxi should pick up passengers and drop them at their destination on a parking lot



Problem setting

State space:

- Grid, number of fields: 25 (5*5)
 - Pickup positions: 5 (Y, R, G, B or in the taxi)
 - Possible destinations: 4
- Number of states: 500

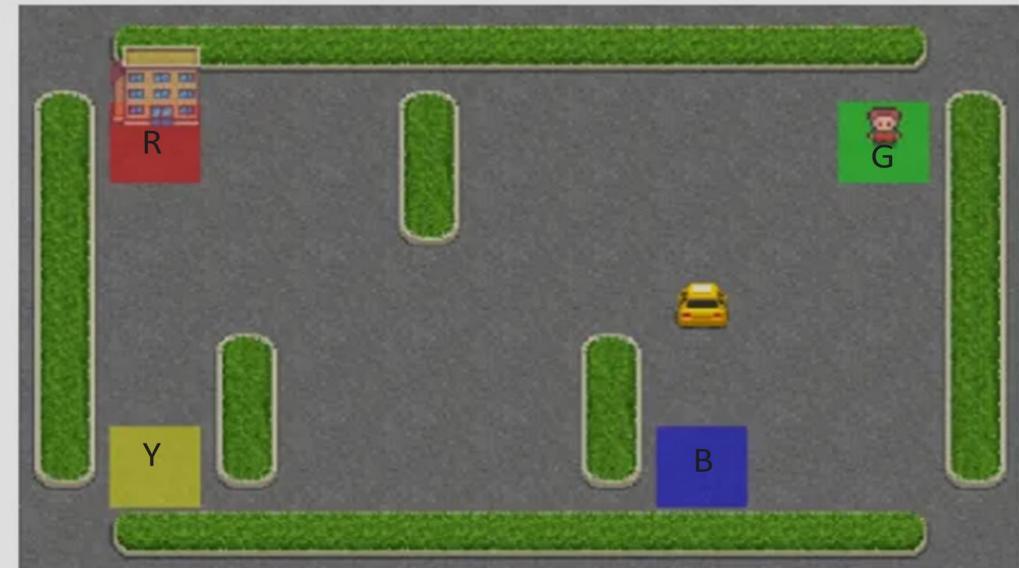
Action space:

- Down, Up, Left, Right
- Drop, Pick up passenger

Reward function:

- Move: -1
- Failed drop-off: -10
- Successful drop-off: 10

State space: Discrete(500)
Action space: Discrete(6)
State: 364
Action: 1
Action mask: [1 1 1 0 0 0]
Reward: -1



Test the trained Q-Table



Links

i.e. sources for self-learning

	Title	Link
Overview	Sutton, Barto: Reinforcement Learning: An Introduction	http://incompleteideas.net/book/bookdraft2017nov5.pdf
	Artificial Intelligence: Reinforcement Learning in Python	https://www.udemy.com/course/artificial-intelligence-reinforcement-learning-in-python/
	Fundamentals of Reinforcement Learning	https://levelup.gitconnected.com/fundamental-of-reinforcement-learning-markov-decision-process-8ba98fa66060
	Reinforcement Learning Series: Overview of Methods	https://www.youtube.com/watch?app=desktop&v=i7q8bISGwMQ
	Reinforcement Learning in Machine Learning	https://pythongeeks.org/reinforcement-learning-in-machine-learning/
	Easy Introduction to Reinforcement Learning	https://www.scribbr.com/ai-tools/reinforcement-learning/
	A (Long) Peek into Reinforcement Learning	https://lilianweng.github.io/posts/2018-02-19-rl-overview/

	Title	Link
Basic concepts of RL	The Exploration/Exploitation trade-off	https://huggingface.co/learn/deep-rl-course/unit1/exp-exp-tradeoff
	Dynamic Programming RL	https://shirsho-12.github.io/blog/rl_dp/
	Elucidating Policy Iteration in Reinforcement Learning – Jack's Car Rental Problem	https://towardsdatascience.com/elucidating-policy-iteration-in-reinforcement-learning-jacks-car-rental-problem-d41b34c8aec7

Links

i.e. sources for self-learning

	Title	Link
Q-Learning	Temporal-Difference (TD) Learning	https://towardsdatascience.com/introduction-to-reinforcement-learning-rl-part-6-temporal-difference-td-learning-2a12f0aba9f9
	Reinforcement Learning 6. Temporal Difference Learning	https://www.slideshare.net/SeungJaeLee17/reinforcement-learning-an-introduction-chapter-6
	Diving deeper into Reinforcement Learning with Q-Learning	https://medium.com/free-code-camp/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe
	Fundamentals of Reinforcement Learning: Navigating Cliffworld with SARSA and Q-learning	https://medium.com/gradientcrescent/fundamentals-of-reinforcement-learning-navigating-cliffworld-with-sarsa-and-q-learning-cc3c36eb5830
	Reinforcement Learning: SARSA and Q-Learning	https://arshren.medium.com/reinforcement-learning-sarsa-and-q-learning-e11ebe87dca9
	An introduction to Q-Learning: reinforcement learning	https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/
	Walking Off The Cliff With Off-Policy Reinforcement Learning	https://towardsdatascience.com/walking-off-the-cliff-with-off-policy-reinforcement-learning-7fdbcdfe31ff
	A Beginners Guide to Q-Learning	https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c
	Q-Learning: Model Free Reinforcement Learning and Temporal Difference Learning	https://www.youtube.com/watch?v=0iqz4tcKN58

Links

i.e. sources for self-learning

	Title	Link
Tutorial Applications	Solving The Taxi Environment With Q-Learning — A Tutorial	https://towardsdatascience.com/solving-the-taxi-environment-with-q-learning-a-tutorial-c76c22fc5d8f
	Text-Flappy Bird	https://aspram.medium.com/learning-flappy-bird-agents-with-reinforcement-learning-d07f31609333
	Practical Reinforcement Learning using Python - 8 AI Agents	https://www.udemy.com/course/practical-reinforcement-learning/

	Title	Link
Real world Applications	9 awesome real world applications of Reinforcement Learning	https://medium.com/@mlblogging.k/9-awesome-applications-of-reinforcement-learning-e1306ed25c09
	Reinforcement Learning and its Real-Life Applications	https://blogs.skillovilla.com/reinforcement-learning-and-its-real-life-applications/
	Mastering the game of Go with deep neural networks and tree search	https://www.nature.com/articles/nature16961.pdf



About me

Dr. Harald Stein

- Data Scientist ~ 8 years experience
 - Algotrader ~ 4 years experience
 - Ph.D. in Economics, Game Theory
-
- LinkedIn: <https://www.linkedin.com/in/harald-stein-phd-1648b51a>
 - ResearchGate: <https://www.researchgate.net/profile/Harald-Stein>

