Autonomous Systems: Deep Learning

Reinforcement Learning Introduction

HOCHSCHULE HEILBRONN

Prof. Dr.-Ing. Nicolaj Stache, Pascal Graf

Heilbronn University of Applied Sciences

Outline

1. Introduction and Overview
2. Literature
3. Theoretical Foundations
4. Tabular Solution Methods
5. Task 1: Grid World
6. Exploration – Exploitation Dilemma
7. Task 2: Improved Exploration

1. Introduction and Overview

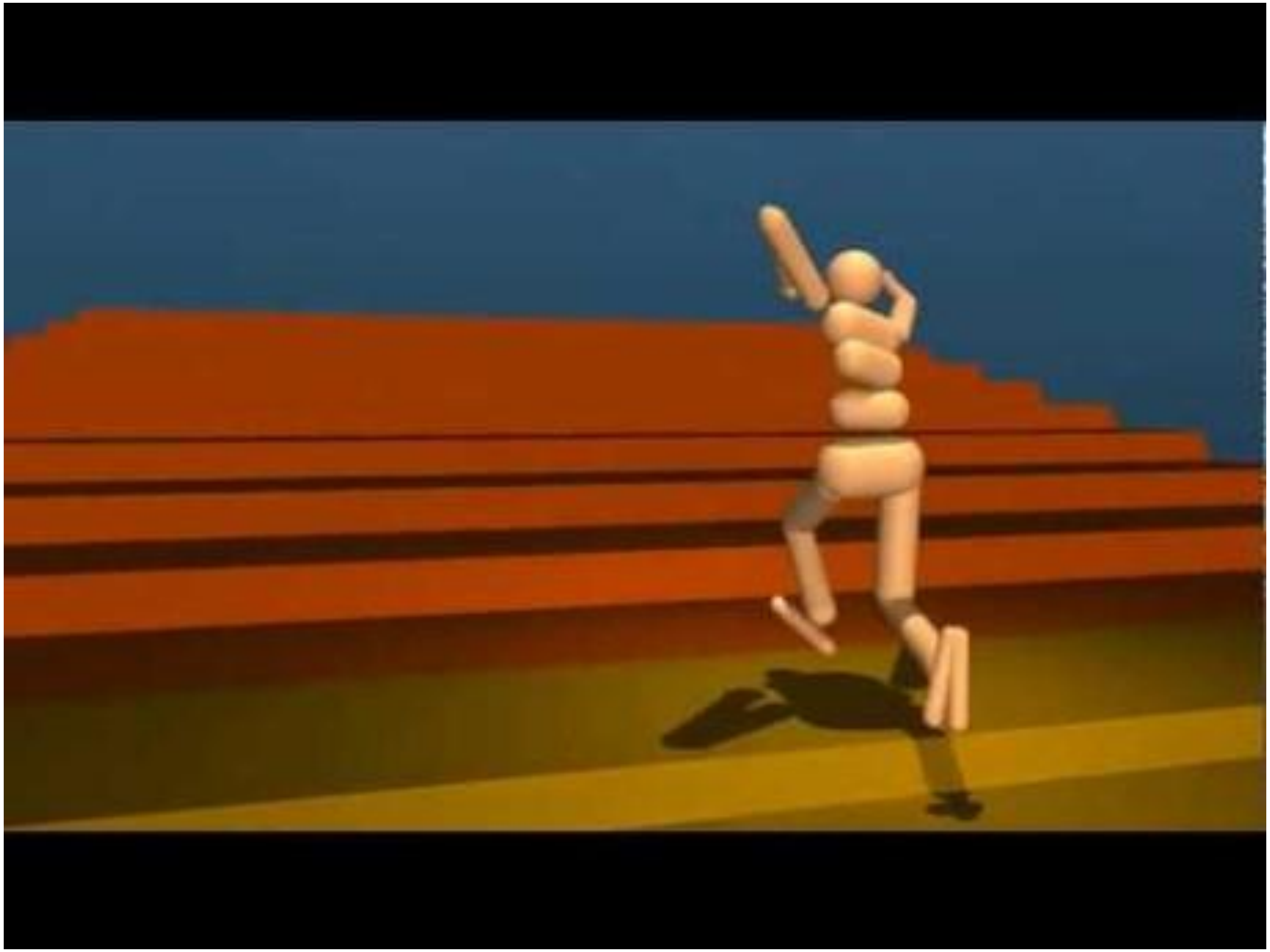
HOCHSCHULE HEILBRONN

Prof. Dr.-Ing. Nicolaj Stache, Pascal Graf
Heilbronn University of Applied Sciences

What can Reinforcement Learning do?

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

Emergence of Locomotion Behaviours in Rich Environments (Deep Mind, 2017)



https://www.youtube.com/watch?v=hx_bgoTF7bs

What can Reinforcement Learning do?

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

Multi-Agent Hide and Seek (OpenAI, 2019)

https://www.youtube.com/watch?v=kopoLzvh5jY

What can Reinforcement Learning do?

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

Solving Rubik's Cube with a Robot Hand (OpenAI, 2019)

https://www.youtube.com/watch?v=x4O8pojMF0w

What can Reinforcement Learning do?

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

From Motor Control to Team Play in Simulated Humanoid Football (Deep Mind, 2022)

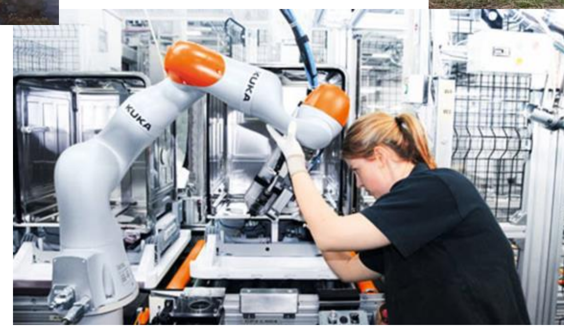
https://www.youtube.com/watch?v=KHMwq9pv7mg&ab_channel=AliEslami

What can Reinforcement Learning do?

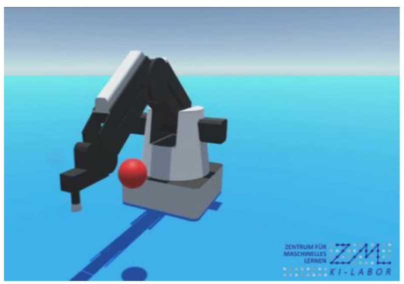
HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

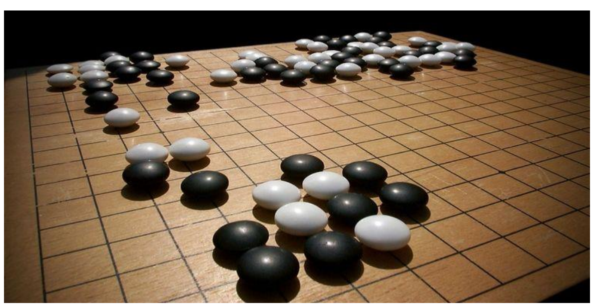
https://www.youtube.com/watch?v=X0j5FmaxrY0

[mundogeo.com]



[kuka.com]



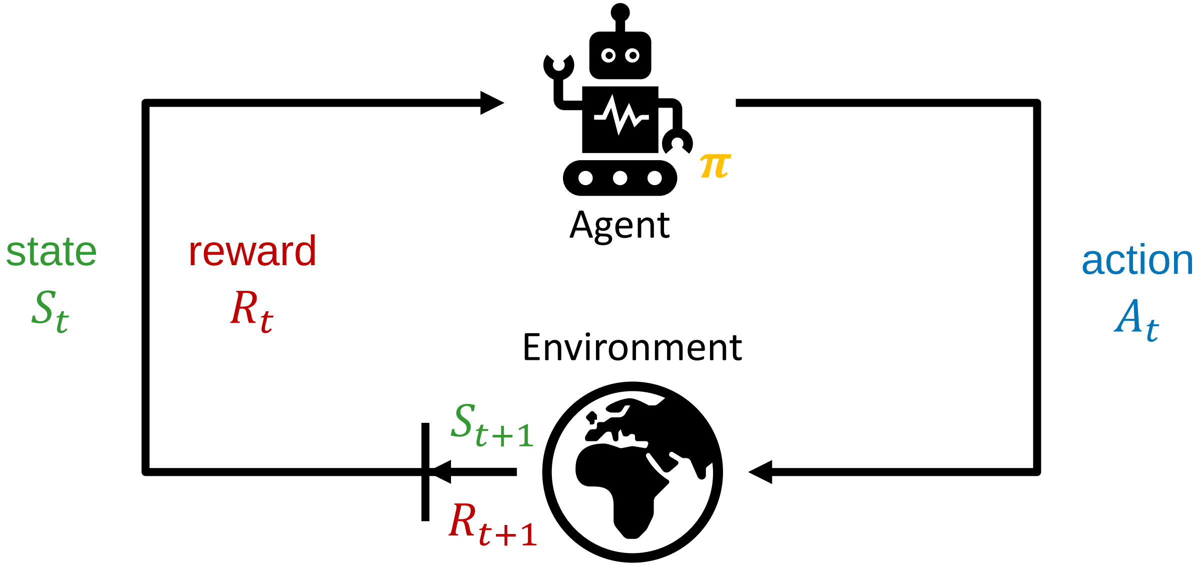


[nikcheerla.github.io]

[dailygame.at]

What is Reinforcement Learning?

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES



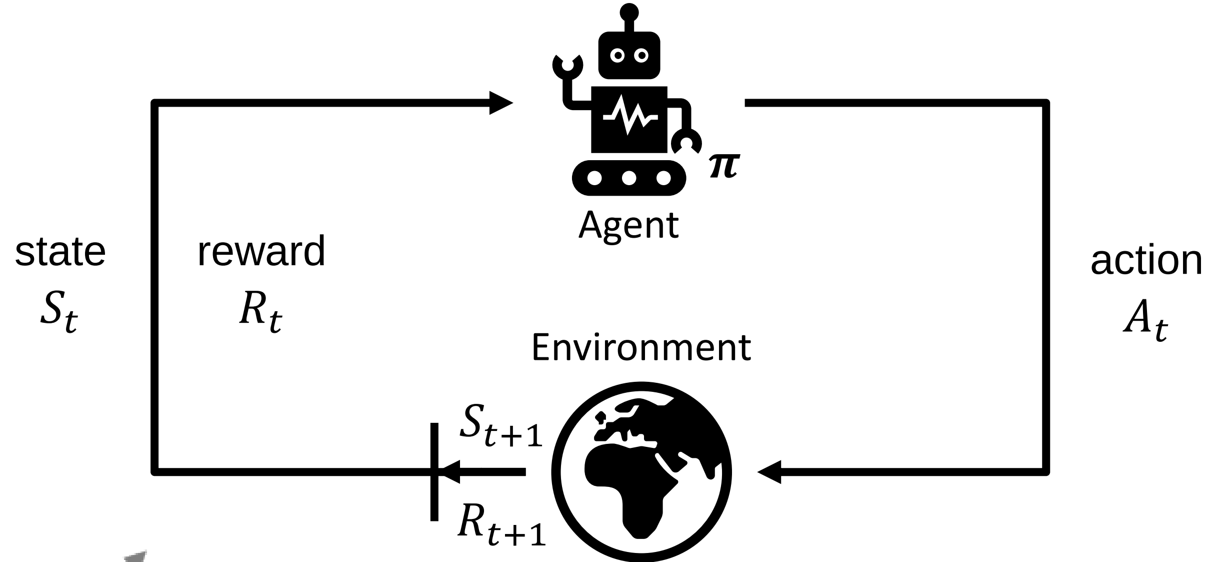
A **policy** π defines the learning agent's way of behaving at a given time. A policy is a mapping from perceived states of the environment to actions to be taken when in those states.

A **reward signal** defines the goal of a reinforcement learning problem. On each time step, the environment sends to the reinforcement learning agent a single number called the **reward**. The agent's sole objective is to maximize the total reward it receives over the long run. The reward signal thus defines what are the good and bad events for the agent.

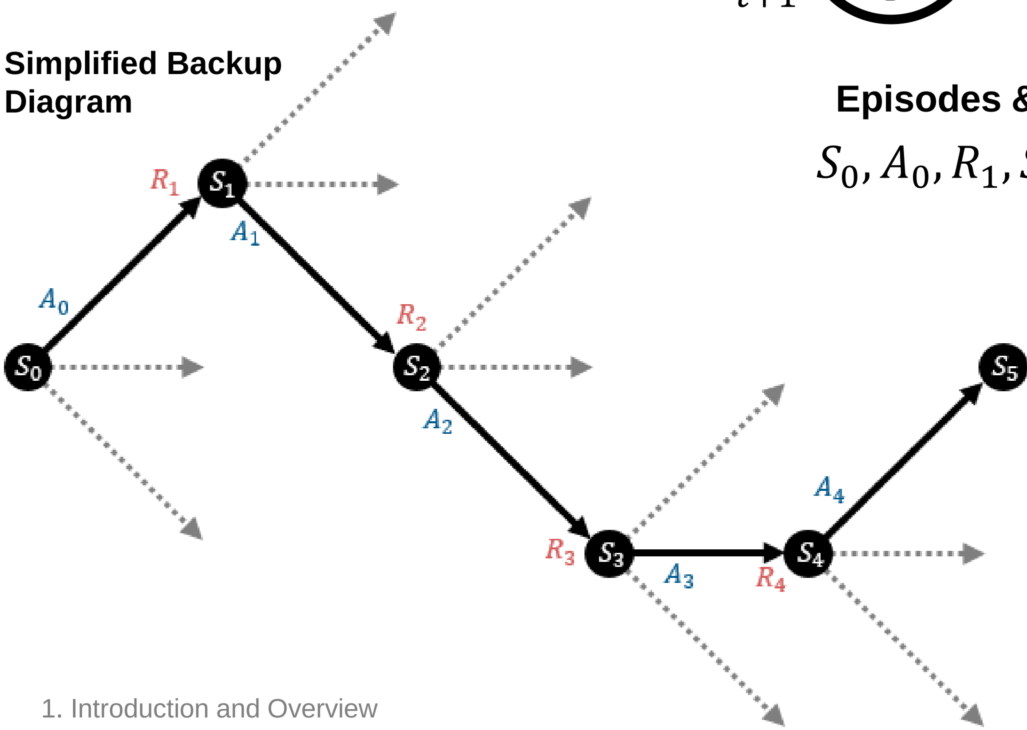
Whereas the **reward signal** indicates what is good in an immediate sense, a **value function** specifies what is good in the long run. Roughly speaking, the value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state.

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

What is Reinforcement Learning?



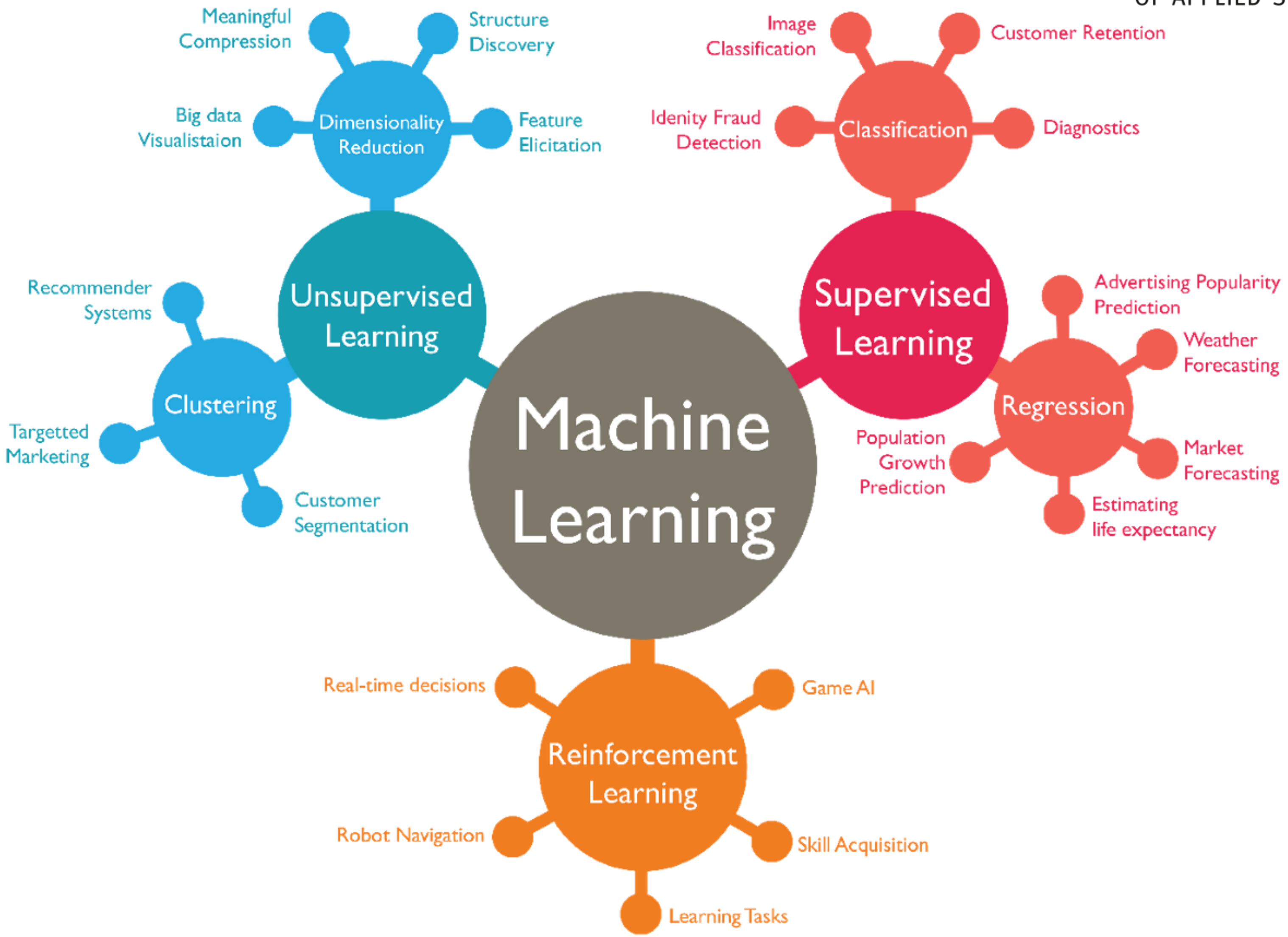
Simplified Backup Diagram



Episodes & Trajectories

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots \quad (1)$$

Why Reinforcement Learning?



https://cdn-images-1.medium.com/max/1200/0*_cgWPP25djXBauNZ.png

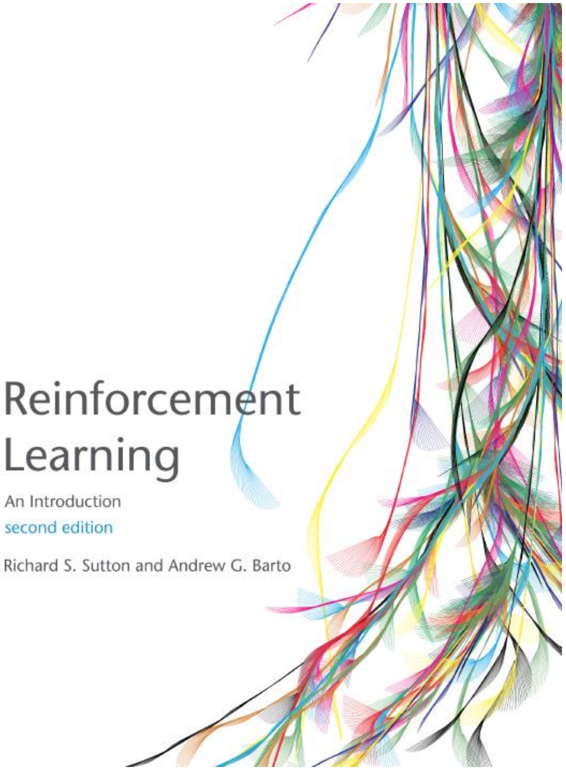
2. Literature

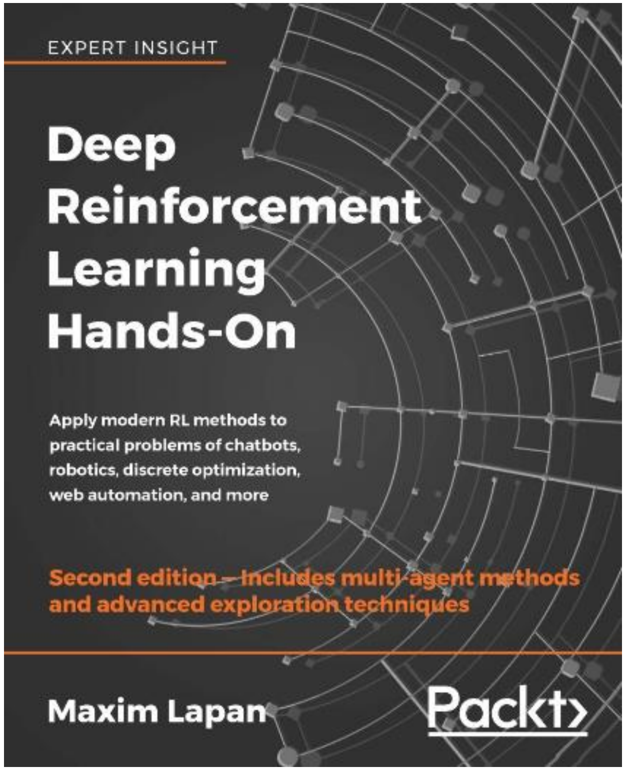
HOCHSCHULE HEILBRONN

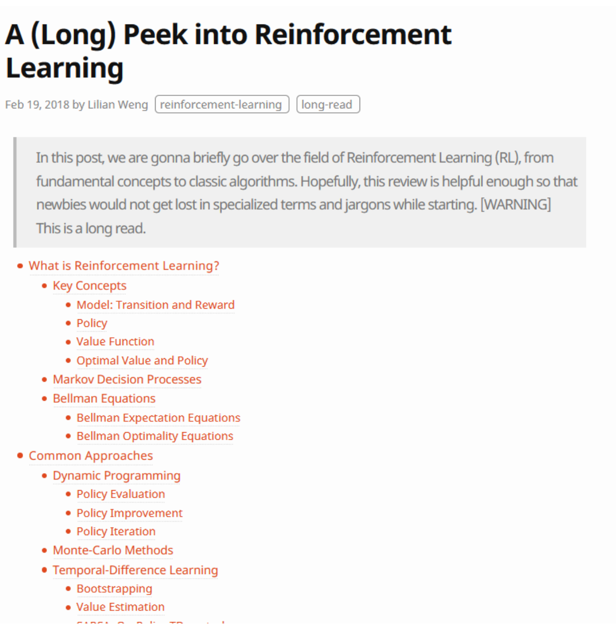
Prof. Dr.-Ing. Nicolaj Stache, Pascal Graf
Heilbronn University of Applied Sciences

Literature

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES







http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

https://www.packtpub.com/data/deep-reinforcement-learning-hands-on-second-edition

https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

3. Theoretical Foundations

HOCHSCHULE HEILBRONN

Prof. Dr.-Ing. Nicolaj Stache, Pascal Graf

Heilbronn University of Applied Sciences

Markov Decision Processes (MDPs)

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

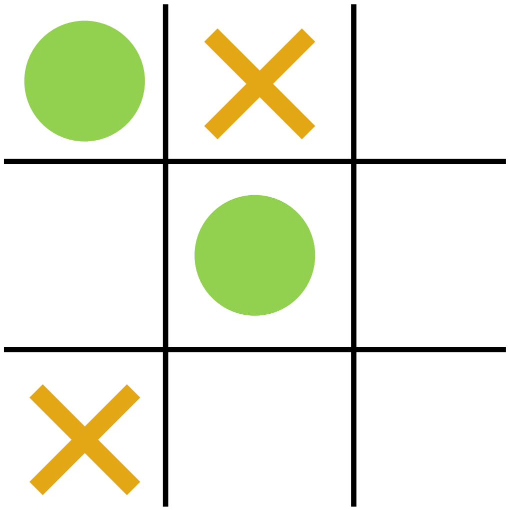
MARKOV PROPERTY AND MARKOV CHAIN

Markov Property: For any given time, the conditional distribution of future states of the process given present and past states depends only on the present state and not at all on the past states (*memoryless property*).
A random process with this property is called **Markov process**.
A **Markov chain** is a Markov process with discrete time and discrete state space.

https://towardsdatascience.com/brief-introduction-to-markov-chains-2c8cab9c98ab

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, S_2, \dots, S_t] \tag{2}$$

Tic-Tac-Toe



Markov Decision Processes (MDPs)

MARKOV PROPERTY AND MARKOV CHAIN

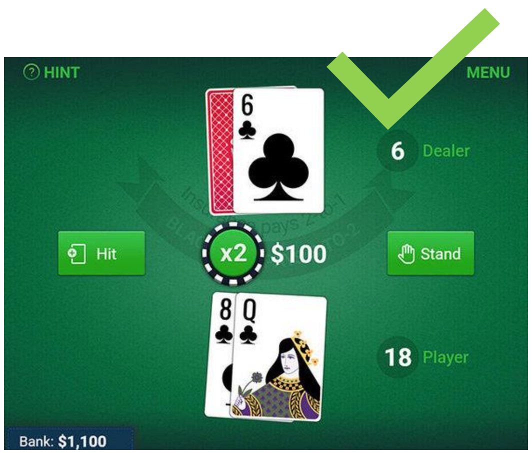
Markov Property: For any given time, the conditional distribution of future states of the process given present and past states depends only on the present state and not at all on the past states (*memoryless property*).
A random process with this property is called **Markov process**.
A **Markov chain** is a Markov process with discrete time and discrete state space.

https://towardsdatascience.com/brief-introduction-to-markov-chains-2c8cab9c98ab

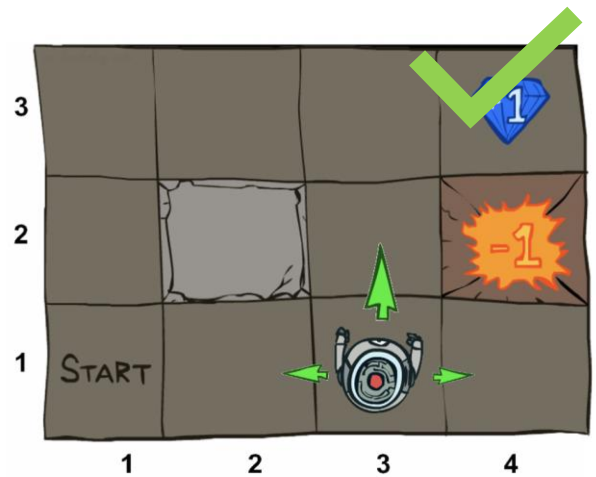
Markov Property:

https://i.ytimg.com/vi/VHqCAaFXpbc/maxresdefault.jpg

's Turn



https://spiele.rtl.de/kartenspiele/black-jack.html



https://cdn-images-1.medium.com/max/1024/1*-oiL7isNsMmktFCNalhTqQ.png

Markov Decision Processes (MDPs)

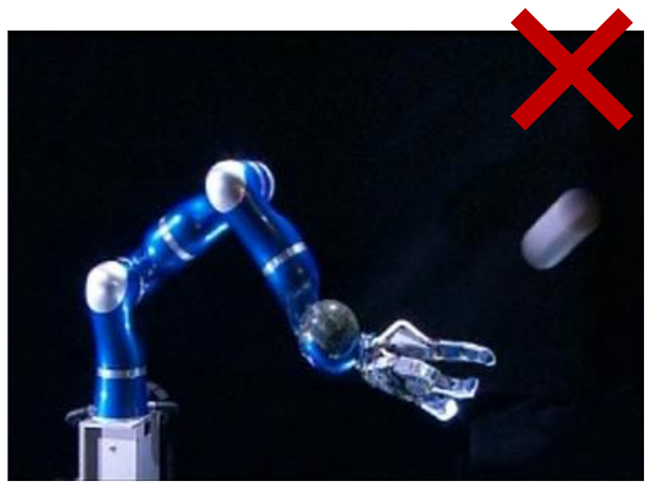
MARKOV PROPERTY AND MARKOV CHAIN

Markov Property: For any given time, the conditional distribution of future states of the process given present and past states depends only on the present state and not at all on the past states (*memoryless property*).
A random process with this property is called **Markov process**.
A **Markov chain** is a Markov process with discrete time and discrete state space.

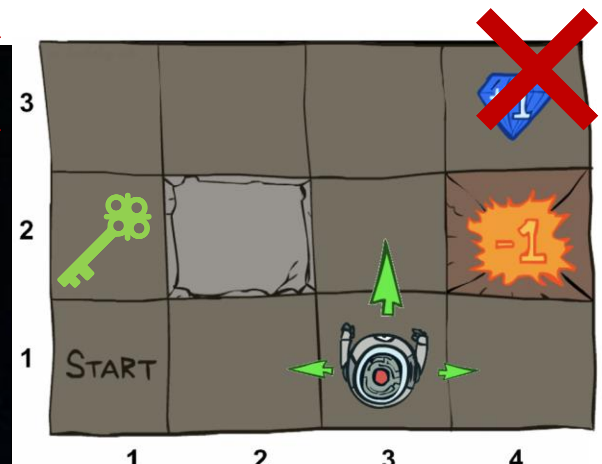
https://towardsdatascience.com/brief-introduction-to-markov-chains-2c8cab9c98ab

Markov Property:

https://www.retrogames.cz/play_222-Atari2600.php



https://www.researchgate.net/profile/Thomas-Wimboeck/publication/225021162/figure/fig1/AS:302781194883082@1449200070953/The-hand-arm-system-catching-a-ball-The-system-is-built-from-the-DLR-LWR-III-arm-with-N.png

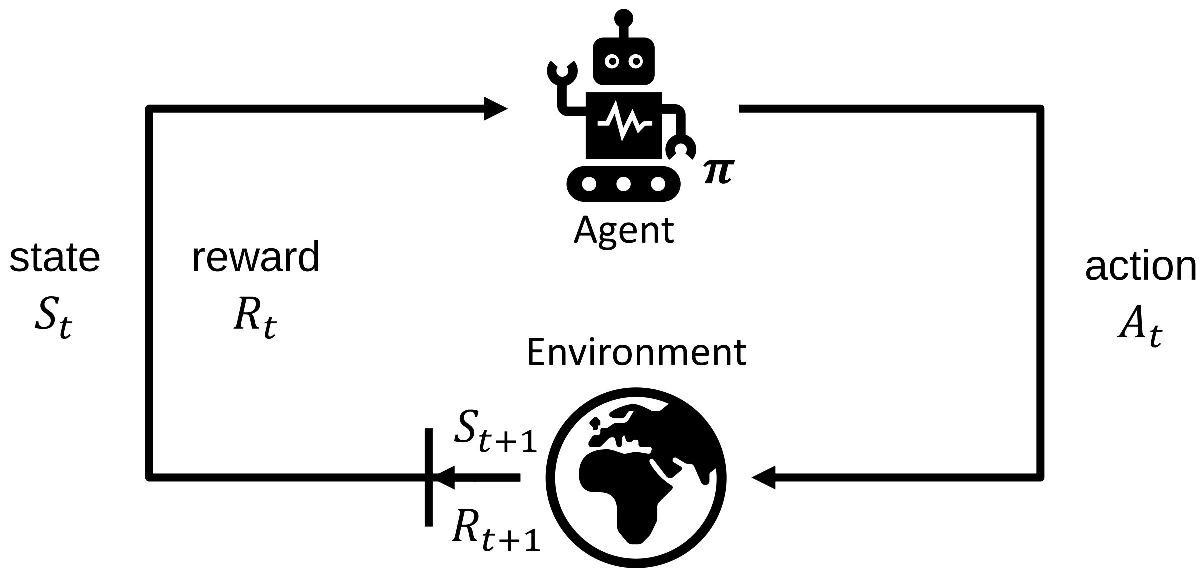


https://cdn-images-1.medium.com/max/1024/1*-oiL7isNsMmktFCNalhTqQ.png

Markov Decision Processes (MDPs)

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

AGENT-ENVIRONMENT INTERFACE



State-Transition Probabilities

Probability

of ending up in state s'

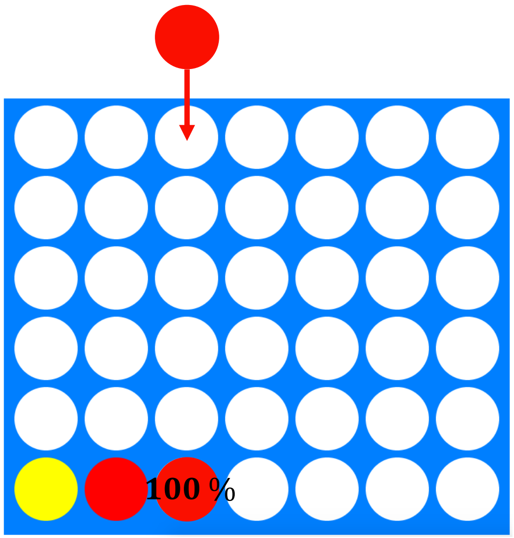
given the case we started in state s and took action a

$$p(s' \mid s, a) = \mathbb{P}\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in R} p(s', r \mid s, a) \quad (3)$$

Markov Decision Processes (MDPs)

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

AGENT-ENVIRONMENT INTERFACE



https://i.stack.imgur.com/v455k.png

https://spiele.rtl.de/kartenspiele/black-jack.html

State-Transition Probabilities

Probability

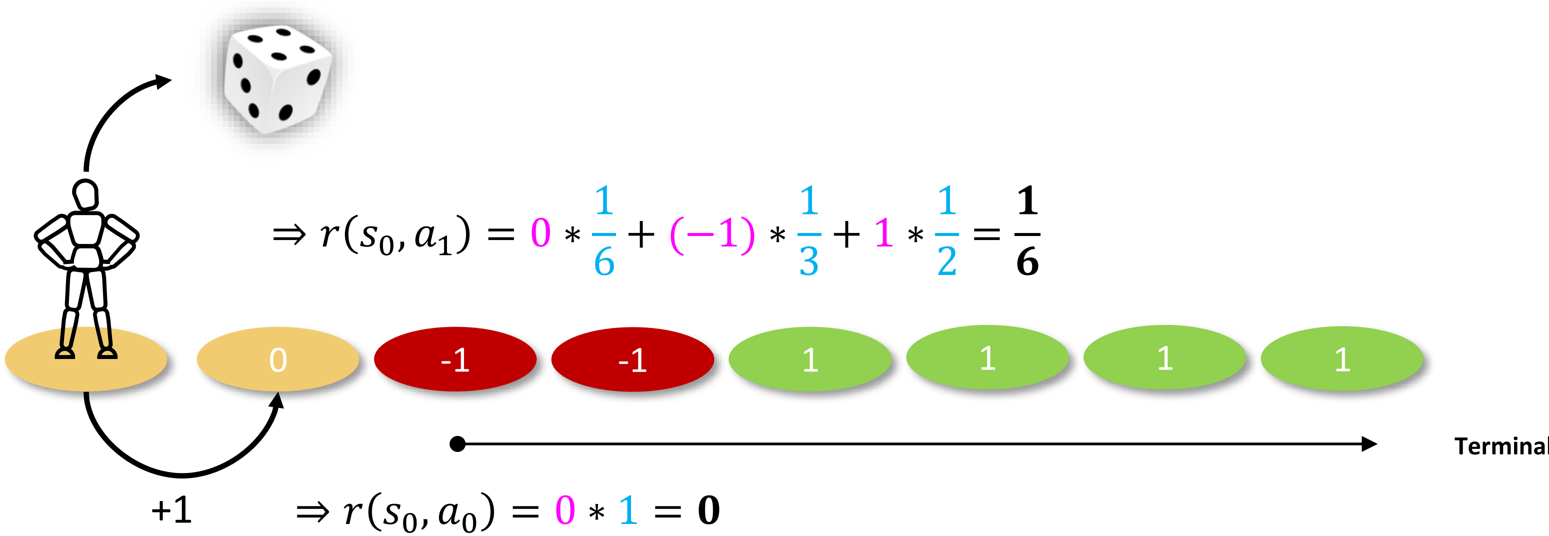
of ending up in state s'

given the case we started in state s and took action a

$$p(s' \mid s, a) = \mathbb{P}\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in R} p(s', r \mid s, a) \tag{3}$$

Markov Decision Processes (MDPs)

AGENT-ENVIRONMENT INTERFACE



$$\Rightarrow r(s_0, a_1) = 0 * \frac{1}{6} + (-1) * \frac{1}{3} + 1 * \frac{1}{2} = \mathbf{\frac{1}{6}}$$

$$\Rightarrow r(s_0, a_0) = 0 * 1 = \mathbf{0}$$

State-Transition Probabilities

$$p(s' \mid s, a) = \mathbb{P}\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in R} p(s', r \mid s, a) \qquad (3)$$

Expected Reward

Expected Reward

when in state s taking action a

$$r(s, a) = \mathbb{E}[\mathrm{R_t} | \mathrm{S_{t-1}} = \mathrm{s}, \mathrm{A_{t-1}} = \mathrm{a}] = \sum_{r \in R} r \sum_{s' \in S} p(s', r | s, a) \qquad (4)$$

Markov Decision Processes (MDPs)

REWARDS AND RETURNS

Return

In general, we seek to maximize the expected return G_t. In the simplest case the return is the sum of the rewards where T is the final time step.

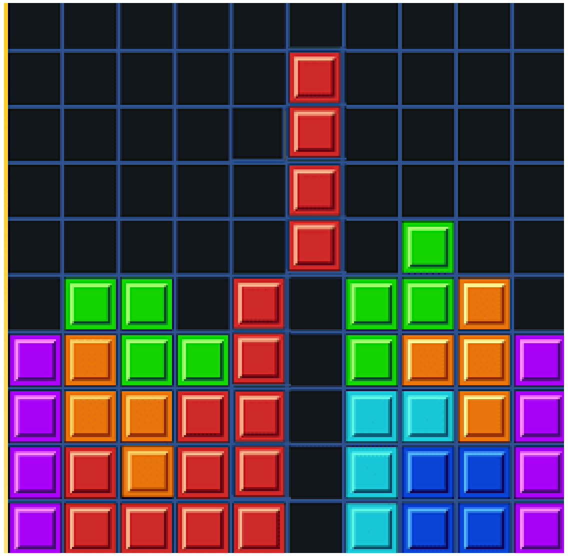
$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ \dots + R_T \tag{5}$$

Discounted Return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{6}$$

Recursive Property

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \dots \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \ \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{7}$$



http://images.sftcdn.net/images/t_optimized,f_auto/p/db2558b0-5fd0-11e7-98e4-117ef89d3ee9/2596915323/classic-tetris-logo.png

Markov Decision Processes (MDPs)

POLICY AND VALUE FUNCTION

State-Value Function

The value function of a state s under a policy π, denoted $v_\pi(s)$, is the expected return when starting in s and following π thereafter.

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \tag{8}$$

Action-Value Function

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \tag{9}$$

Bellman Equation

$$\boxed{v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a)[r + \gamma v_\pi(s')]} \tag{10}$$

$$\boxed{q_\pi(s, a) = \sum_{s',r} p(s', r | s, a)[r + \gamma v_\pi(s')]} \tag{11}$$

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

Bellman Optimality Equations

$$v_*(s) = \max_a \mathbb{E}_{\pi_*}[G_t | S_t = s, A_t = a] \quad (12)$$
$$\boxed{= \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')]}$$

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] \quad (13)$$
$$\boxed{= \sum_{s',r} p(s', r | s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right]}$$

4. Tabular Solution Methods

HOCHSCHULE HEILBRONN

Prof. Dr.-Ing. Nicolaj Stache, Pascal Graf
Heilbronn University of Applied Sciences

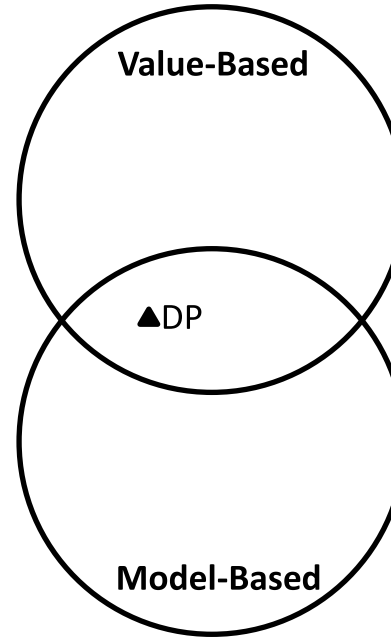
Dynamic Programming

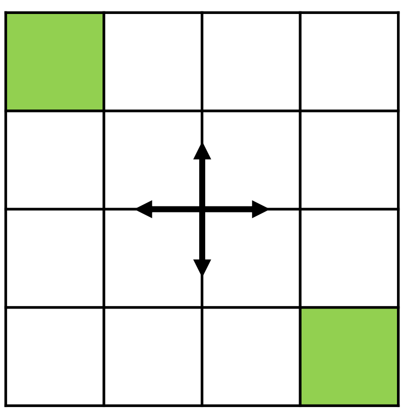
HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

POLICY EVALUATION (PREDICTION)

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \qquad (10)$$

$$\boxed{v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')]} \qquad (14)$$





$R_t = -1$
on all transitions
$\gamma = 1$

Dynamic Programming

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

POLICY EVALUATION (PREDICTION)

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 0$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$k = 1$

0	-1.75	-2	-2
-1.75	-2	-2	-2
-2	-2	-2	-1.75
-2	-2	-1.75	0

$k = 2$

0	-2.44	-2.94	-3
-2.44	-2.88	-3	-2.94
-2.94	-3	-2.88	-2.44
-3	-2.94	-2.44	0

$k = 3$

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')] \quad (14)$$

$$\begin{aligned} v_1(s_{(0,1)}) = &\ 0.25 * 1 * [-1+0] \\ &+ 0.25 * 1 * [-1+0] \\ &+ 0.25 * 1 * [-1+0] \\ &+ 0.25 * 1 * [-1+0] \end{aligned}$$

$$\begin{aligned} v_2(s_{(0,1)}) = &\ 0.25 * 1 * [-1+0] \\ &+ 0.75 * 1 * [-1+(-1)] \end{aligned}$$

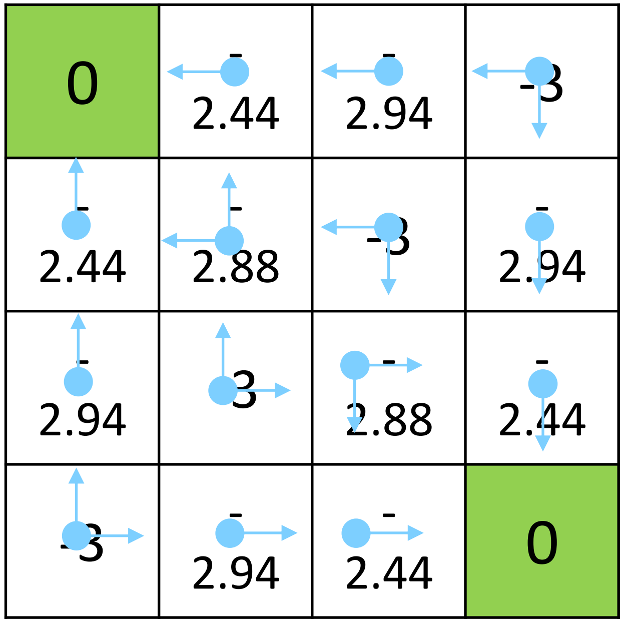
$$\begin{aligned} v_3(s_{(0,1)}) = &\ 0.25 * 1 * [-1+0] \\ &+ 0.25 * 1 * [-1+(-1.75)] \\ &+ 0.5 * 1 * [-1+(-2)] \end{aligned}$$

Dynamic Programming

POLICY IMPROVEMENT

Bellman Equations

$$\pi'(s) = \mathrm{argmax}_{\mathrm{a}} \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \qquad (15)$$



$k = 3$

Dynamic Programming

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

POLICY ITERATION

$$\pi_0 \xrightarrow{\mathrm{E}} v_{\pi_0} \xrightarrow{\mathrm{I}} \pi_1 \xrightarrow{\mathrm{E}} v_{\pi_1} \xrightarrow{\mathrm{I}} \pi_2 \xrightarrow{\mathrm{E}} \cdots \xrightarrow{\mathrm{I}} \pi_* \xrightarrow{\mathrm{E}} v_*$$

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

PROBLEMS & CONSTRAINTS

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \quad (10)$$

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_*(s')\right] \quad (12)$$

https://i.ytimg.com/vi/fWJSzIZIowl/maxresdefault.jpg

Model Dependance

$7.7 * 10^{45}$ States

https://i.ytimg.com/vi/VHqCAaFXpbc/maxresdefault.jpg

Number of States

Monte Carlo Methods (MC)

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

https://i.ytimg.com/vi/fWJSzIZlowl/maxresdefault.jpg

Model Dependance

https://i.ytimg.com/vi/VHqCAaFXpbc/maxresdefault.jpg

Number of States

1. Learning the Action-Value

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} q_*(s',a')\right] \qquad (13)$$

2. Learning from Experience

MC Update: $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t - Q(S_t, A_t)] \qquad (16)$$

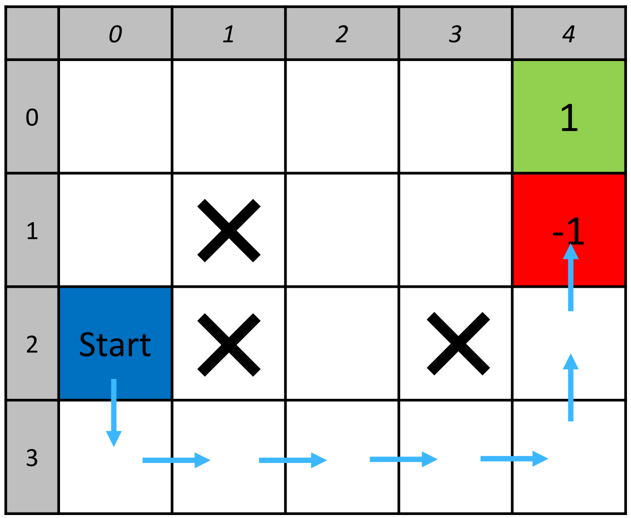
Monte Carlo Methods (MC)

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

MC Update: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t - Q(S_t, A_t)]$ (16)

Learning Rate Error

Acting with an $\epsilon - greedy\ policy$



$R_t = -0.1$
on all other transitions

State	*Left*	*Right*	*Top*	*Bottom*
…	…	…	…	…
(2,0)	0	0	0	0
(3,0)	0	0	0	0
(0,1)	0	0	0	0
(3,1)	0	0	0	0
(0,2)	0	0	0	0
…	…	…	…	…

$\gamma = 1; \alpha = 0.1$

$T_{R_0} = (-0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -1)$

$G_0 = (-1.6, -1.5, -1.4, -1.3, -1.2, -1.1, -1)$

Monte Carlo Methods (MC)

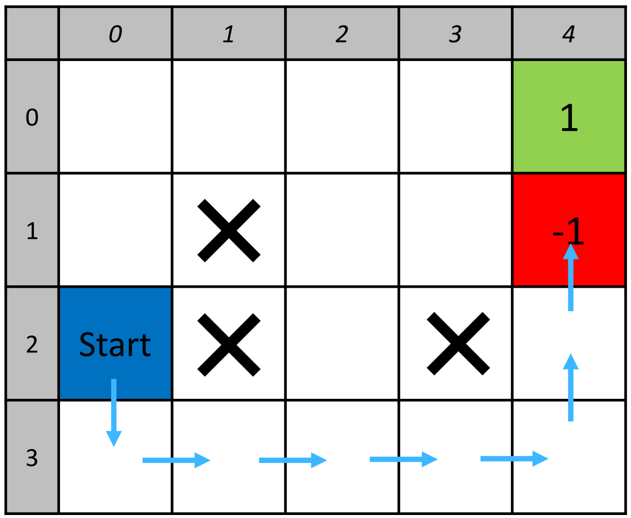
HEILBRONN UNIVERSITY OF APPLIED SCIENCES

MC Update: $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t - Q(S_t, A_t)] \quad (16)$$

Learning Rate

Error

Acting with an $\epsilon - greedy\ policy$



$R_t = -0.1$
on all other transitions

State	*Left*	*Right*	*Top*	*Bottom*
…	…	…	…	…
(2,0)	0	0	0	-0.16
(3,0)	0	-0.15	0	0
(0,1)	0	0	0	0
(3,1)	0	-0.14	0	0
(0,2)	0	0	0	0
…	…	…	…	…

$$\gamma = 1; \alpha = 0.1$$

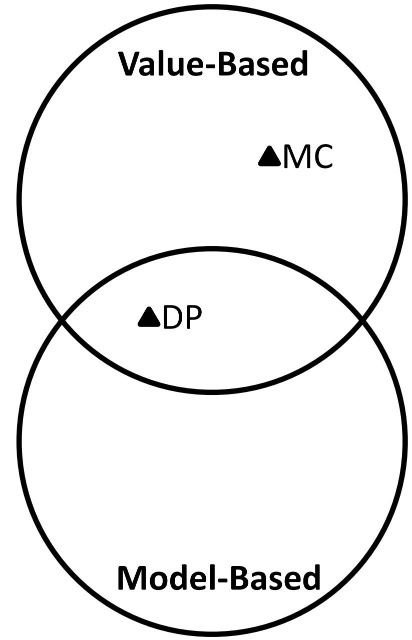
$$T_{R_0} = (-0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -1)$$

$$G_0 = (-1.6, -1.5, -1.4, -1.3, -1.2, -1.1, -1)$$

Monte Carlo Methods (MC)

MC Update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t - Q(S_t, A_t)] \qquad (16)$$



On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

  $\pi \leftarrow$ an arbitrary ε-soft policy

  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

  Generate an episode following π: $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

  $G \leftarrow 0$

  Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

    $G \leftarrow \gamma G + R_{t+1}$

    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \dots, S_{t-1}, A_{t-1}$:

      Append G to $Returns(S_t, A_t)$

      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

      $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

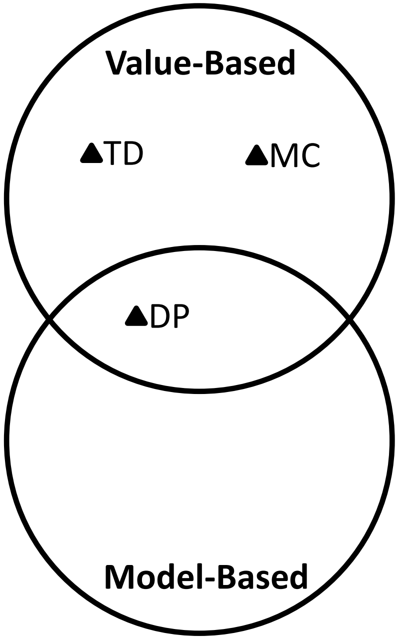
      For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

TD PREDICTION



MC Update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t - Q(S_t, A_t)] \quad (16)$$

TD Update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \quad (17)$$

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

ADVANTAGES OF TD PREDICTION METHODS

- TD methods update their estimates based in part on other estimates. They learn a guess from a guess - they **bootstrap**.
- TD methods have an advantage over DP methods in that they do not require a model of the environment, of its reward and next-state probability distributions.
- TD methods are naturally implemented in an online, fully incremental fashion. With Monte Carlo methods one must wait until the end of an episode, because only then is the return known, whereas with TD methods one need wait only one time step. Some applications have very long episodes, so that delaying all learning until the end of the episode is too slow. Other applications are continuing tasks and have no episodes at all.

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

Temporal-Difference Learning

Q-LEARNING: OFF-POLICY TD CONTROL

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{19}$$

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 $Q(terminal, \cdot) = 0$

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

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

5. Task 1: Grid World

HOCHSCHULE HEILBRONN

Prof. Dr.-Ing. Nicolaj Stache, Pascal Graf
Heilbronn University of Applied Sciences

Grid World

Task:

Implement a Q-Learning algorithm to find an optimal policy in the Grid World environment described underneath.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \quad (19)$$

	0	*1*	*2*	*3*	*4*
0					1
1		X			-1
2	Start	X		X	
3					

$R_t = -0.1$
on all other transitions

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

6. Exploration - Exploitation Dilemma

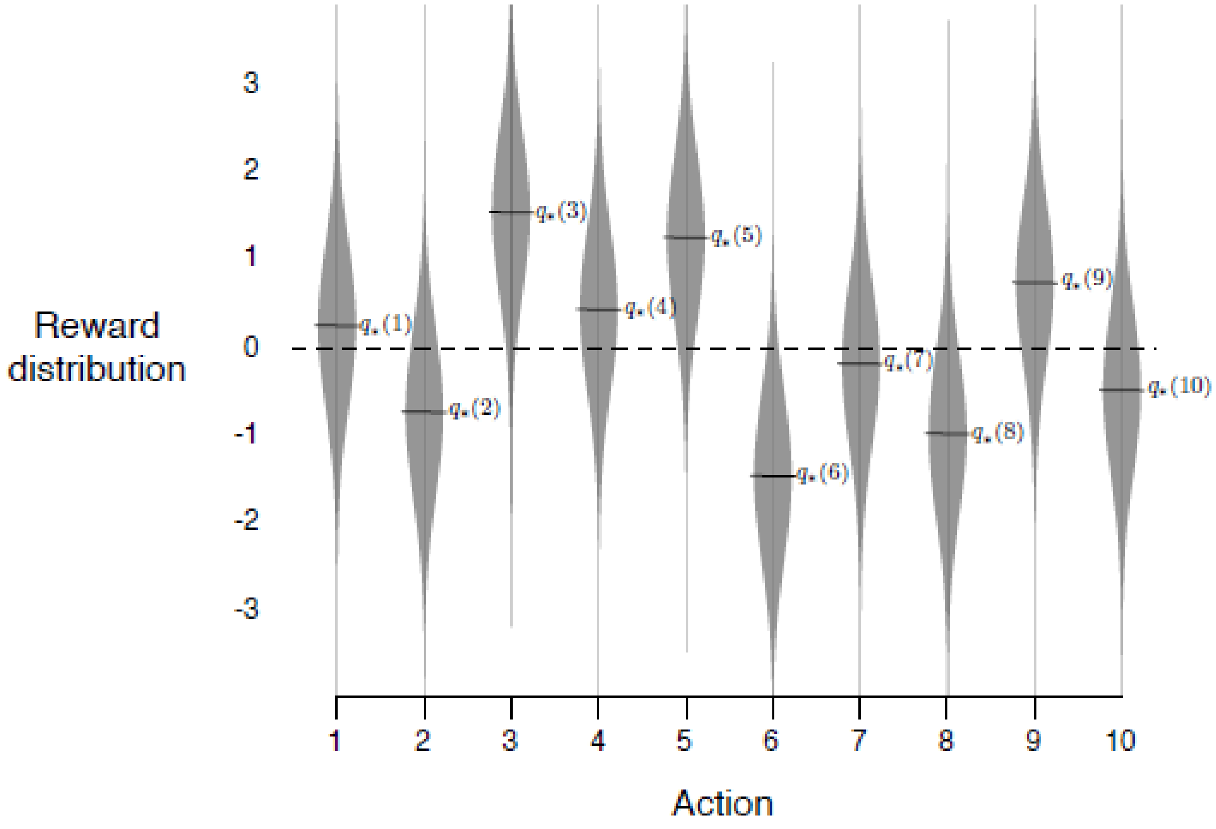
HOCHSCHULE HEILBRONN

Prof. Dr.-Ing. Nicolaj Stache, Pascal Graf

Heilbronn University of Applied Sciences

K-Armed Bandit Problem

Objective: Maximize the expected total reward over some time period, for example, over 1000 action selections, or time steps.



http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

K-Armed Bandit Problem

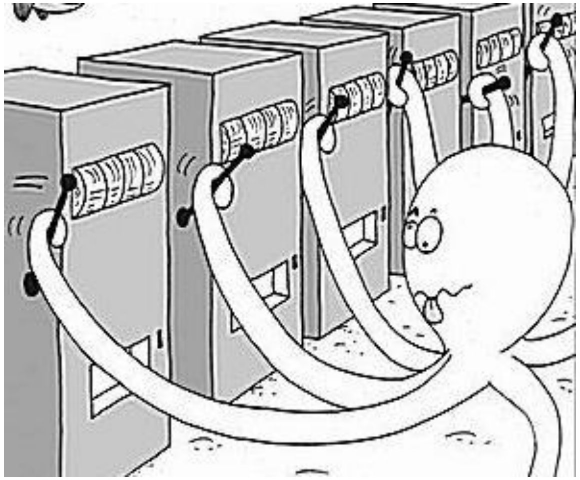
HEILBRONN UNIVERSITY OF APPLIED SCIENCES

The value of an arbitrary action a, denoted $q_*(a)$, is the expected reward given that a is selected:

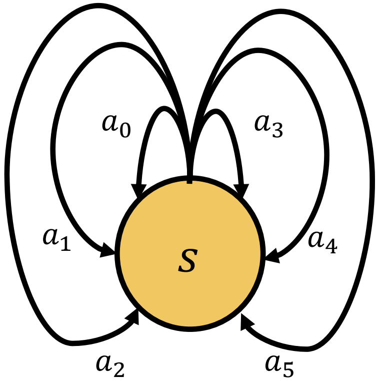
$$q_*(a) = E[R_t | A_t = a]. \qquad (1)$$

The estimated value of action a at time step t is $Q_t(a)$. We would like $Q_t(a)$ to be close to $q_*(a)$.

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf



https://blogs.mathworks.com/loren/2016/10/10/multi-armed-bandit-problem-and-exploration-vs-exploitation-trade-off/



Greedy vs. Non-Greedy

When maintaining estimates of the action values, at any time step there is at least one action whose estimated value is greatest. We call these the **greedy** actions. When you select one of these actions, we say that you are **exploiting** your current knowledge. If instead you select one of the **nongreedy** actions, then we say you are **exploring**.

Estimating the action value by averaging:

$$Q_t(a) = \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} \quad (2)$$

We now turn to the question of how these averages can be computed in a computationally efficient manner.

$$Q_{n+1} = \frac{R_1 + \mathrm{R}_2 + \mathrm{R}_3 + \cdots + \mathrm{R}_\mathrm{n}}{\mathrm{n}} = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} R_i \quad (3)$$

$$\boxed{Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]} \quad (4)$$

$$NewEstimate \leftarrow OldEstimate + StepSize * [Target - OldEstimate]$$

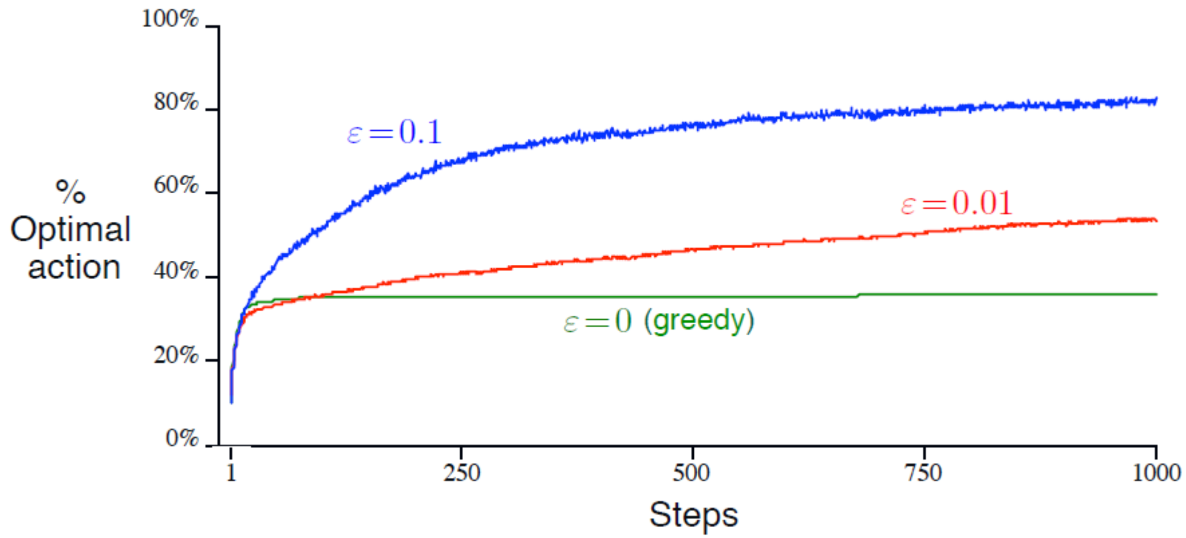
http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

ϵ –Greedy

Greedy Acting:

$$A_t = argmax_a \, Q_t(a). \tag{5}$$

Epsilon-Greedy:
With small probability ϵ, select randomly from among all the actions with equal probability.

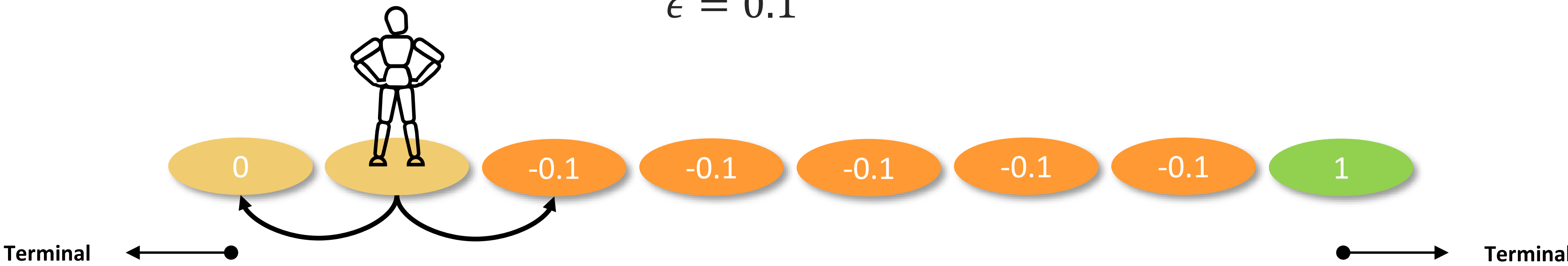


http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

ϵ –Greedy

Constraints

$$\epsilon = 0.1$$



https://i.ytimg.com/vi/VHqCAaFXpbc/maxresdefault.jpg

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

Upper Confidence Bound (UCB)

HEILBRONN UNIVERSITY OF APPLIED SCIENCES

Rather than performing exploration by simply selecting an **arbitrary action**, chosen with a probability that remains constant, the UCB algorithm **changes its exploration-exploitation balance** as it gathers more knowledge of the environment.

$$A_t = argmax_a \left[\underbrace{Q_t(a)}_{\text{Exploitation}} + \underbrace{c \sqrt{\frac{\log t}{N_t(a)}}}_{\text{Exploration}} \right]$$

$Q_t(a)$	Estimated Action Value
$N_t(a)$	Number of times action *a* has been selected before
c	Confidence value controlling the level of exploration
t	Number of times state *s* has been visited before

http://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

7. Task 2: Improved Exploration

HOCHSCHULE HEILBRONN

Prof. Dr.-Ing. Nicolaj Stache, Pascal Graf
Heilbronn University of Applied Sciences