Q-Learning – How to?

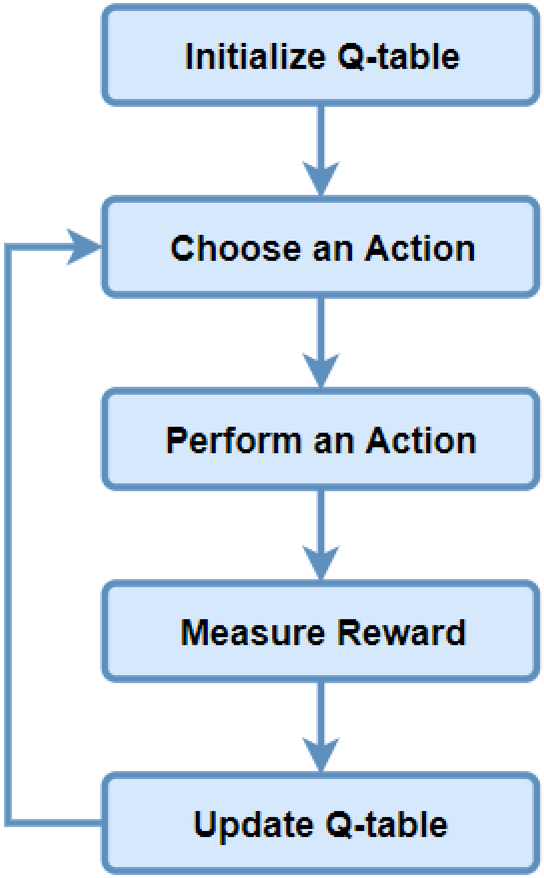
Institute of Automation and Information Systems

Technical University of Munich

Agenda

Lehrstuhl AIS
Fakultät für Maschinenwesen
Technische Universität München

Q-Learning Process Steps



Source: https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c

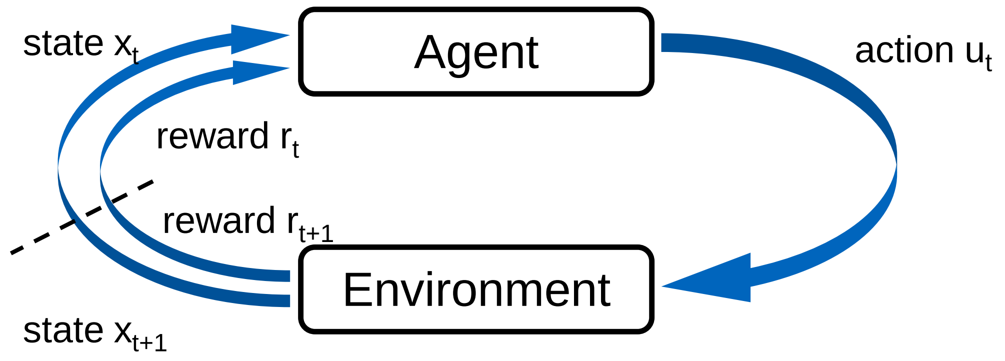
Q-Learning Process Steps

Possible Actions n

Possible States m

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

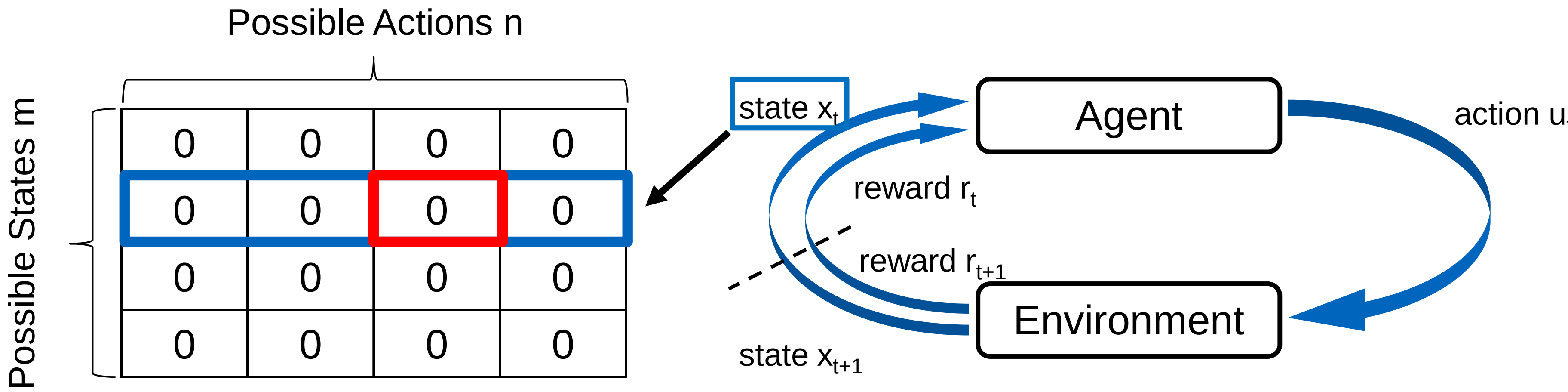
Initialize with 0



1. Initialize Q-Table

- Use discretization for environments with contiuous state variables
- If the state is described by multiple state variables $X_1, X_2, \dots, X_k$ (such as position, velocity and acceleration) with corresponding number of discretization intervals $m_1, m_2, \dots m_k$, the Q-Table is of dimension $m_1 \times m_2 \times \dots \times m_k \times n$

Q-Learning Process Steps



2. Choose an Action:

- Get the current state x_t
- Perform ϵ-greedy strategy:

$$\text{Action} = \begin{cases} \max\limits_{u} Q(x,u), & R > \epsilon \\ \text{Random } u, & R \leq \epsilon \end{cases}$$, R is uniform random value in $[0,1]$ and $\epsilon \in [0,1]$

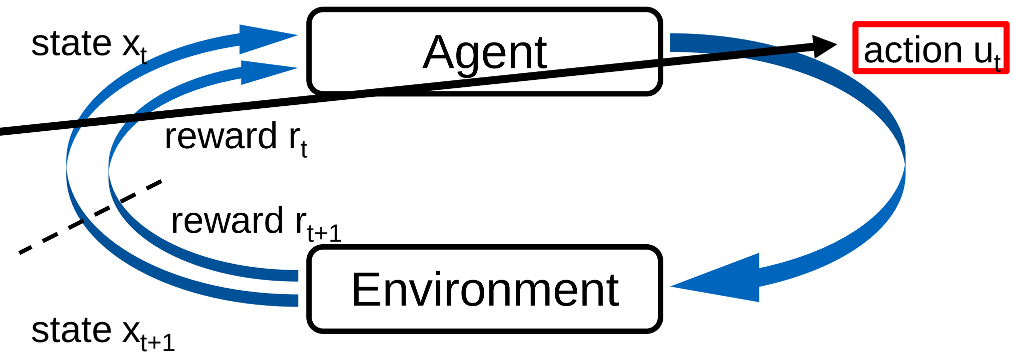
→ when highest Q-Value occurs more than once, choose randomly from actions with this value

Q-Learning Process Steps

Possible Actions n

Possible States m

0	0	0	0
0	0	$\boxed{0}$	0
0	0	0	0
0	0	0	0



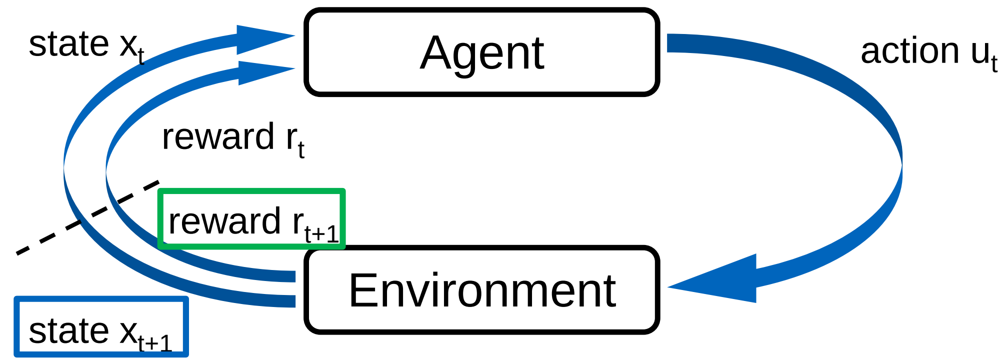
3. Perform an Action

Q-Learning Process Steps

Possible Actions n

Possible States m

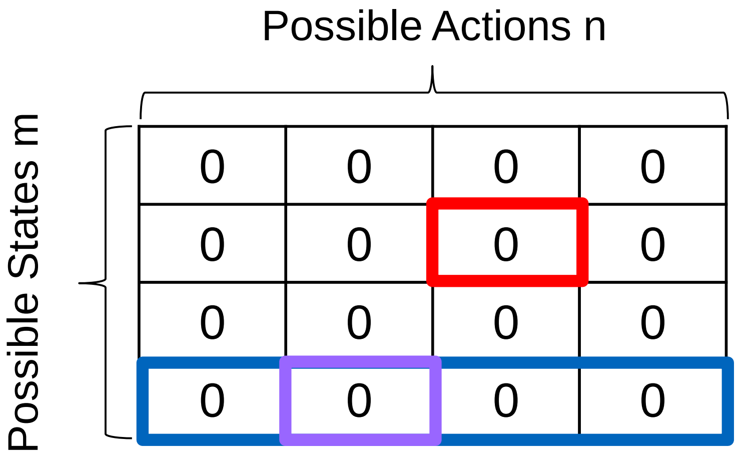
0	0	0	0
0	0	$\boxed{0}$	0
0	0	0	0
0	0	0	0

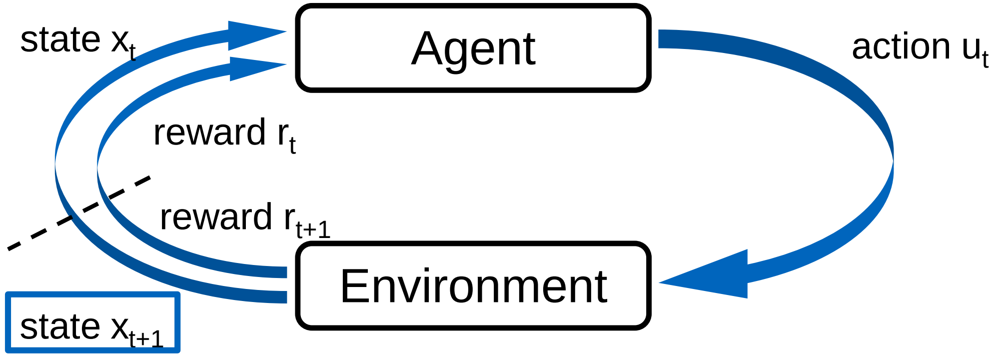


4. Measure reward

And remember new state x_{t+1}

Q-Learning Process Steps





5. Update Q-Table

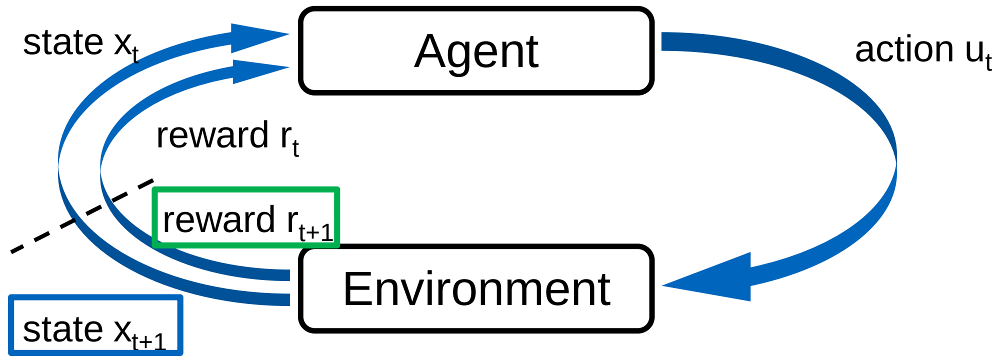
- Go to new state x_{t+1}
- Find the highest possible Q-value of state x_{t+1} (Q-value = expected future reward)

Q-Learning Process Steps

Possible Actions n

Possible States m

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



5. Update Q-Table

- Go to new state x_{t+1}
- Find the highest possible Q-value of state x_{t+1} (Q-value = expected future reward)
- Update Q-value for the original state-action pair:

$$\boxed{Q^{\pi}_{k+1}(x_t, u_t)} = (1-\alpha)\cdot \boxed{Q^{\pi}_{k}(x_t, u_t)} + \alpha \cdot \left(\boxed{r_{t+1}} + \gamma \cdot \boxed{\underset{u}{\operatorname{argmax}}\, Q(\boxed{x_{t+1},} u)}\right)$$

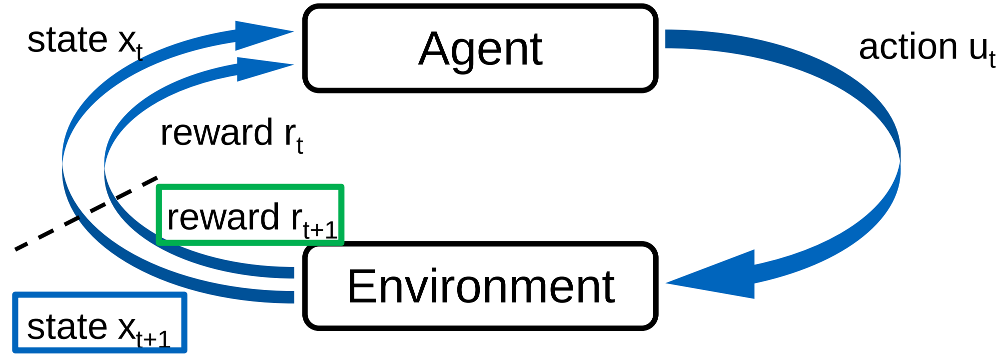
$$r_{t+1} = 20, \qquad \alpha = 10^{-3}, \qquad \gamma = 0.95$$

Q-Learning Process Steps

Possible Actions n

Possible States m

0	0	0	0
0	0	0.02	0
0	0	0	0
0	0	0	0



5. Update Q-Table

- Go to new state x_{t+1}
- Find the highest possible Q-value of state x_{t+1} (Q-value = expected future reward)
- Update Q-value for the original state-action pair:

$$\boxed{Q^{\pi}_{k+1}(x_t, u_t)} = (1-\alpha) \cdot \boxed{Q^{\pi}_{k}(x_t, u_t)} + \alpha \cdot \left(\boxed{r_{t+1}} + \gamma \cdot \boxed{\operatorname*{argmax}_u Q(\boxed{x_{t+1}}, u)} \right) = \boxed{0.02}$$

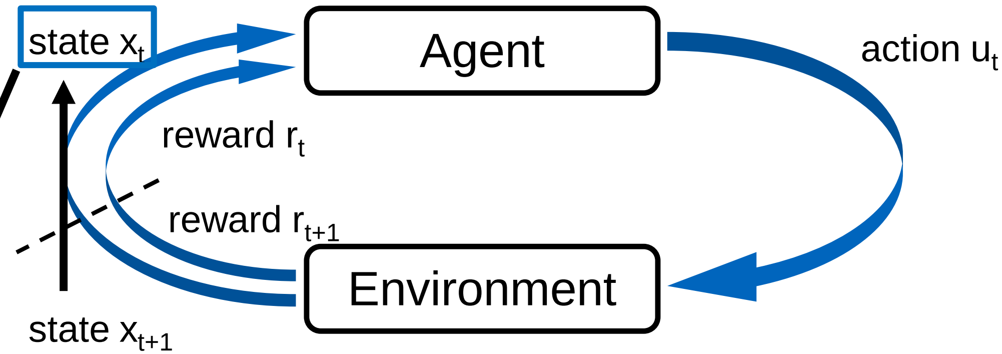
$$r_{t+1} = 20, \qquad \alpha = 10^{-3}, \qquad \gamma = 0.95$$

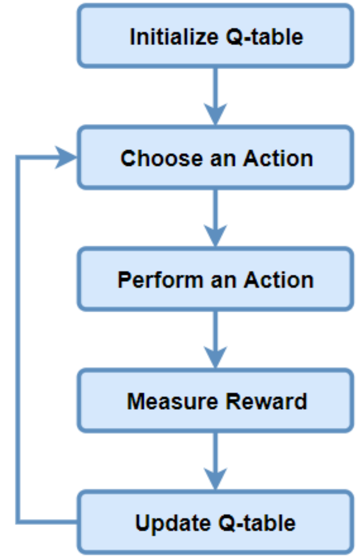
Q-Learning Process Steps

Possible Actions n

Possible States m

0	0	0	0
0	0	0.02	0
0	0	0	0
0	0	0	0





Go back to step 2 and continue

Agenda

1 **Q-Learning Process Steps**

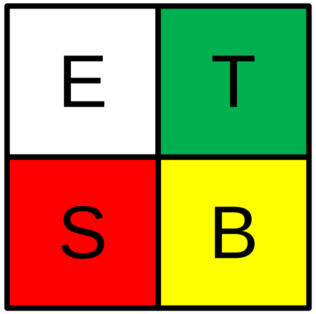
2 **Exercise: Grid World**

3 **Exercise: Crane-Simulation**

Grid World – Task 1

- Small Grid World: 2x2
- Go from a starting point (S) to a target location (T)
- Possible actions: up, down, left or right (crashing into the boundary → position remains the same, but counts as new step for reward calculation)
- Rewards:

x_{t+1}	$r_{t+1}(x_{t+1})$
E	-1
T	+100
B	-0.5
S	-1

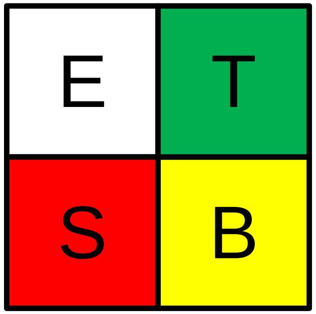


- Task 1:
  - Perform the first iteration of Q-Learning (Steps 1-5) by hand
  - Use learning rate $\alpha = 0.1$ and discount factor $\gamma = 1$
  - Neglect epsilon-greedy strategy

Grid World – Task 2

- Task 2:
  - After 5 completed episodes with epsilon-greedy strategy ($\epsilon = 0.05$) the Q-Table looks like shown below
  - Compute the first iteration of Q-Learning for episode 6, starting again at (S) by hand
  - As before: Neglect epsilon-greedy strategy

	Up	Down	Left	Right
E	0	-0.1	-0.1	0
T	0	0	0	0
S	-0.1	-0.1	-0.1	7.91
B	40.95	0	-0.1	0

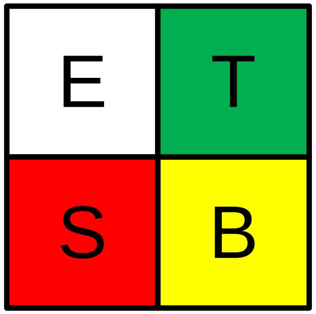


Grid World – Task 3

- Imagine an epsilon-greedy strategy where ϵ is computed as a function of the current episode:
  → $\epsilon(\text{episode}) = 0.9999^{\text{episode}}$
- Epsilon gets very small over time

- Task 3: What does the Q-Table look like after an infinite amount of episodes?

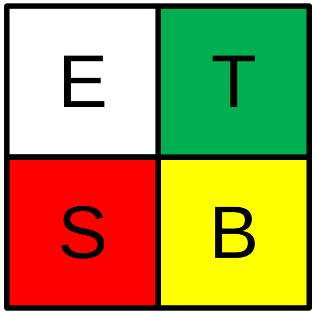
$$Q_{k+1}^{\pi}(x_t, u_t) = (1-\alpha) \cdot Q_k^{\pi}(x_t, u_t) + \alpha \cdot \left(r_{t+1} + \gamma \cdot \underset{u}{\operatorname{argmax}}\, Q(x_{t+1}, u)\right)$$



Grid World – Task 4

– Task 4: What happens, if we apply the following reward structure to the Grid World problem?

x_{t+1}	$r(x_{t+1})$
E	-1
T	+100
B	+1
S	-1



Agenda

1 Q-Learning Process Steps

2 Exercise: Grid World

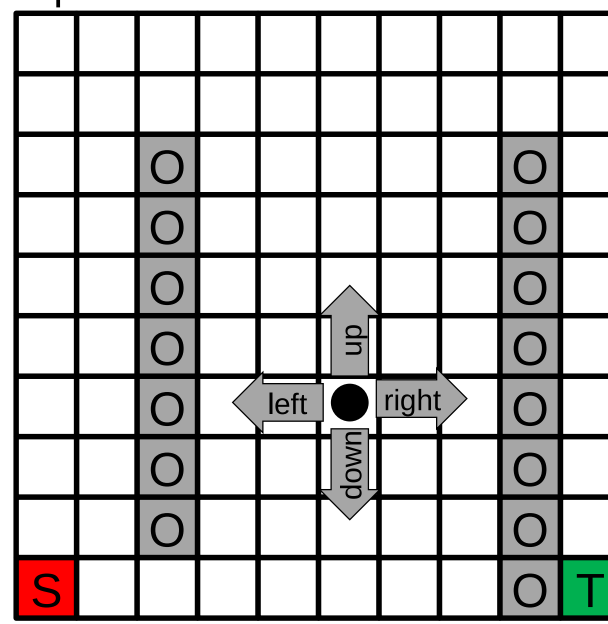
3 **Exercise: Crane-Simulation**

Crane Simulation

- Quadratic Warehouse → discretized into a 10 x 10 grid
- A product has to be transported with a crane from a starting point (S) to a target location (T)
- When the crane crashes into an high rack storage (O), the game ends
- Possible actions: up, down, left or right (crashing into the boundary → position remains the same)
- Rewards:

x_{t+1}	$r_{t+1}(x_{t+1})$
White field / (S)	-1
(T)	+100
(O)	-100

Top view:



Crane Simulation - Tasks

Tasks:

- Open *crane_simulation_Template.py*
- In class *CraneSim:* define the grid and all required variables
- In class *Q_Agent*: implement the epsilon-greedy policy and the update function for the Q-Table
- In function *play*: implement all necessary steps to run one episode of the simulation

		O						O	
		O						O	
		O						O	
		O						O	
		O						O	
		O						O	
		O						O	
S								O	T

Source: https://www.vettercranes.com/produkte/profi-portalkrane/zweischienenportale/