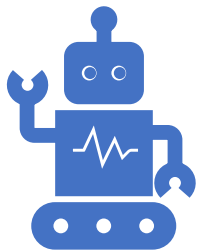




UiO : **University of Oslo**



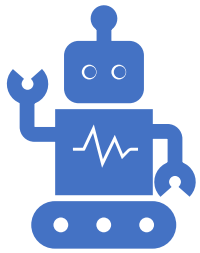
IN3050/IN4050, Lecture 12

Reinforcement learning

Weria Khaksar and Kai Olav Ellefsen



UiO : **University of Oslo**



IN3050/IN4050, Lecture 12

Reinforcement learning

1: The reinforcement learning problem

Weria Khaksar

Next video: Reward and action selection



[The Commuter \(2018\)](#)



[Ghostbusters \(1984\)](#)

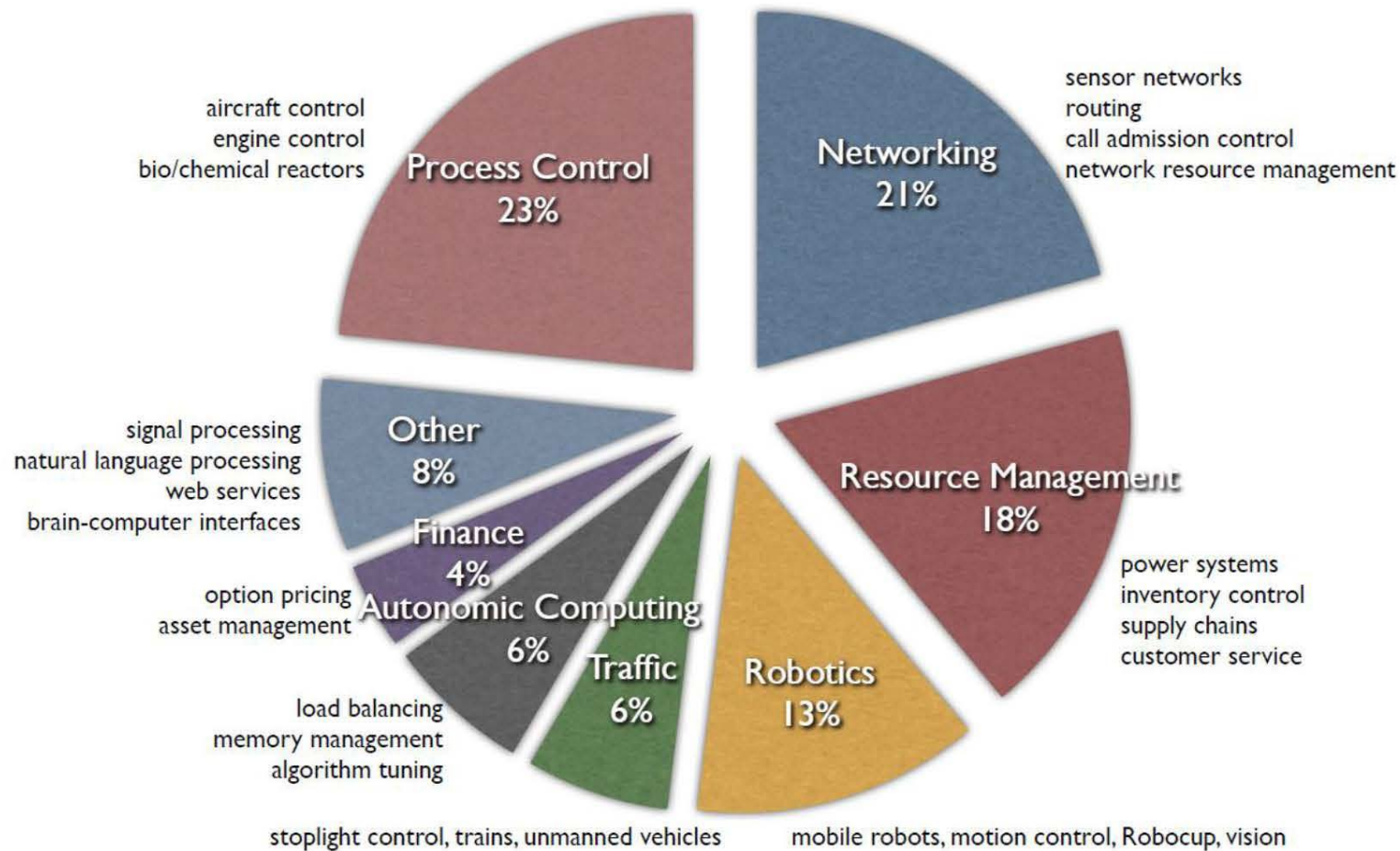
"It would be in vain for one Intelligent Being , to set a Rule to the Actions of another, if he had not in his Power, to reward the compliance with, and punish deviation from his Rule, by some Good and Evil, that is not the natural product and consequence of the action itself."

(Locke, "Essay", 2.28.6)

"The use of punishments and rewards can at best be a part of the teaching process. Roughly speaking, if the teacher has no other means of communicating to the pupil, the amount of information which can reach him does not exceed the total number of rewards and punishments applied."

(Turing (1950) "Computing Machinery and Intelligence")

Applications of RL:



From: [“Deconstructing Reinforcement Learning”, ICML 2009](#)

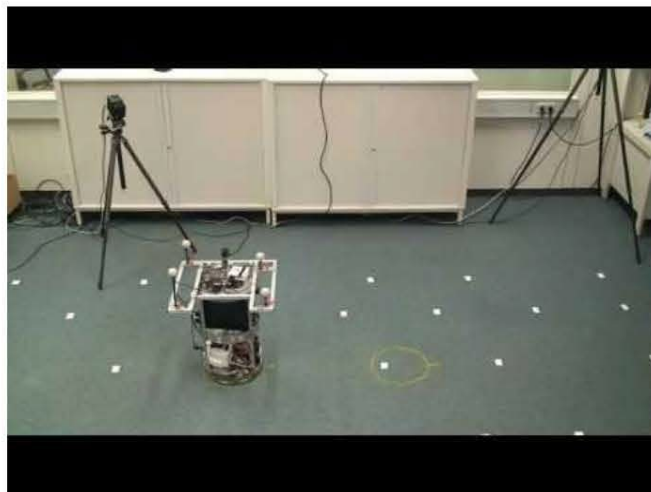
Examples:



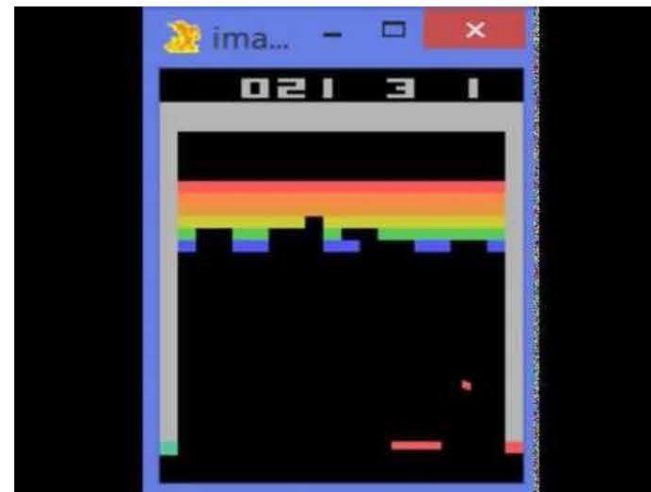
[Barrett WAM robot learning to flip pancakes by reinforcement learning](#)



[Socially Aware Motion Planning with Deep Reinforcement Learning](#)

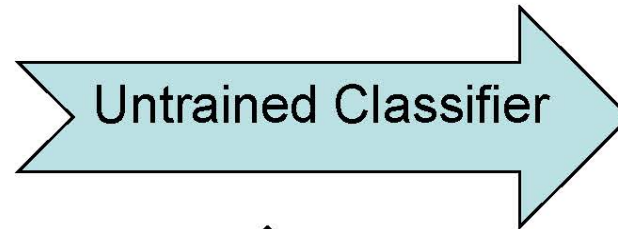


[Hierarchical Reinforcement Learning for Robot Navigation](#)

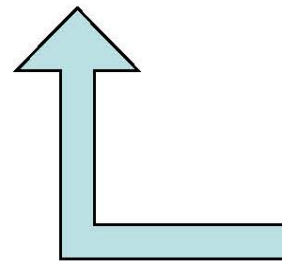


[Google DeepMind's Deep Q-learning playing Atari Breakout](#)

Last time: Supervised learning

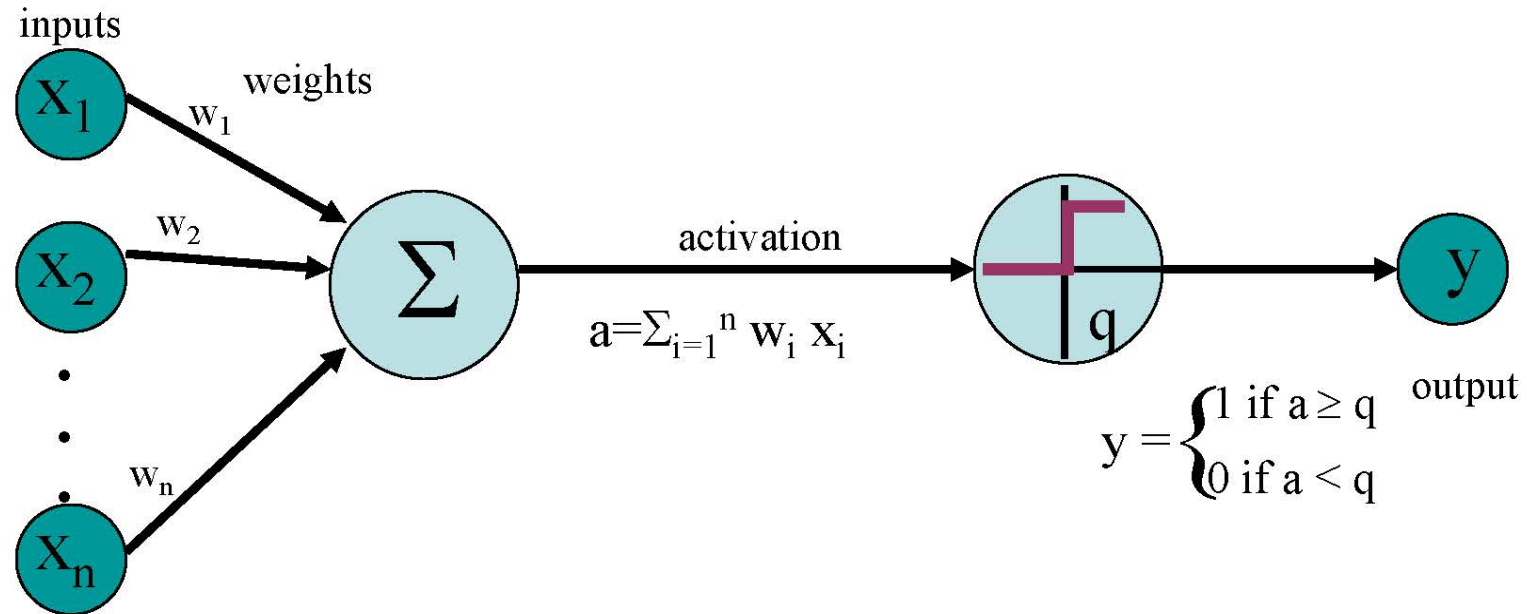


"CAT"



No, it was a dog.
Adjust classifier
parameters

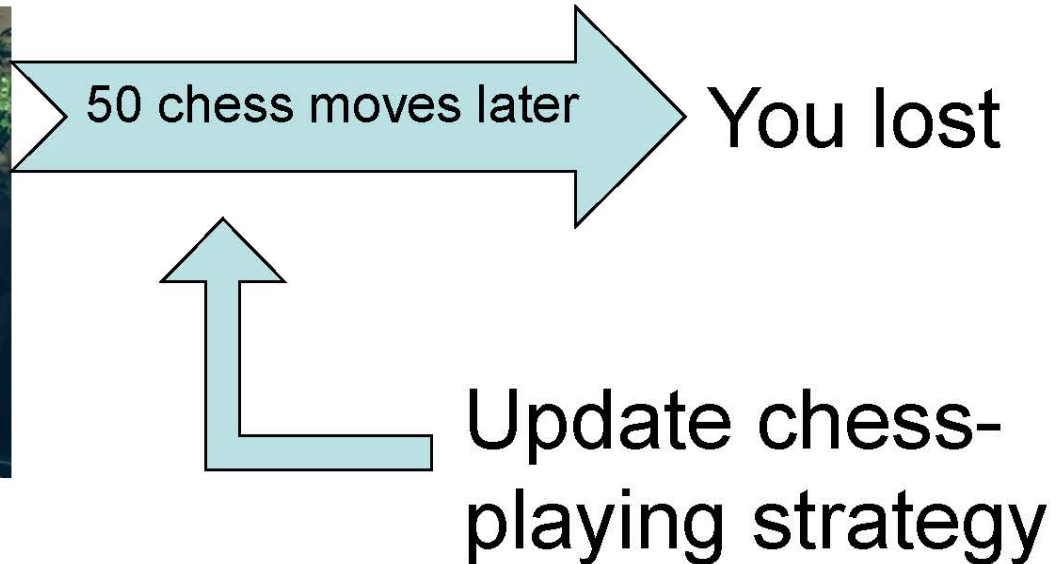
Supervised learning: Weight updates



$$\Delta w_{ij} = \eta \cdot (t_j - y_j) \cdot x_i$$

Learning rate (points to η)
Input (points to x_i)
Desired output (points to t_j)
Actual output (points to y_j)
Error (points to $t_j - y_j$)

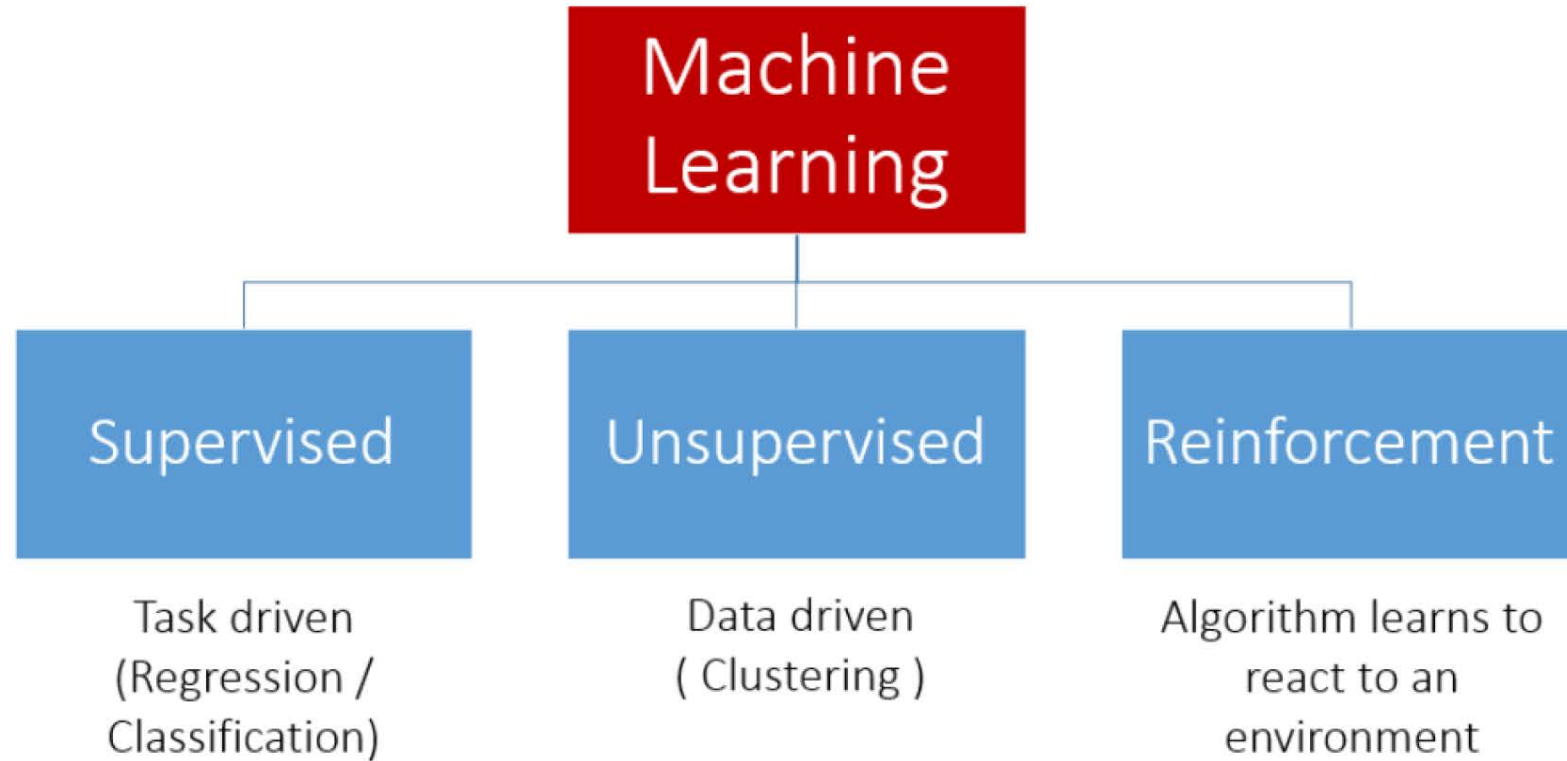
Reinforcement Learning: Infrequent Feedback



How do we update our system now? We don't know the error!

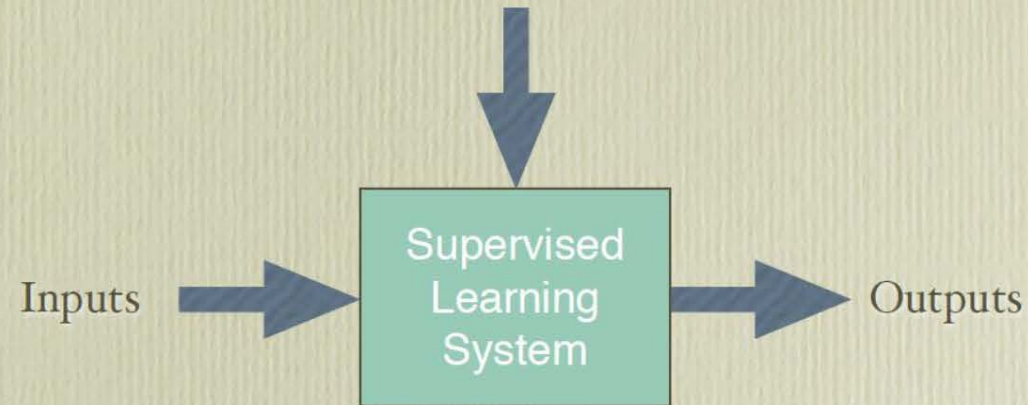
$$\Delta w_{ij} = \eta \cdot (t_j - y_j) \cdot x_i$$

The diagram shows the equation $\Delta w_{ij} = \eta \cdot (t_j - y_j) \cdot x_i$ enclosed in a black rectangular box. Red arrows point from text labels to parts of the equation: 'Learning rate' points to η , 'Input' points to x_i , 'Desired output' points to t_j , and 'Actual output' points to y_j . A red arrow labeled 'Error' points to the entire term $(t_j - y_j)$.



Supervised Learning

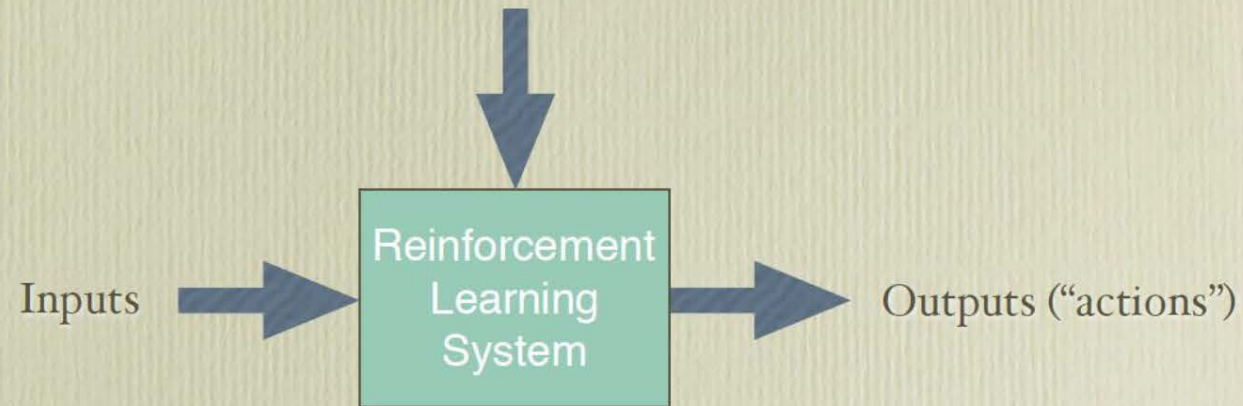
Training Info = desired (target) outputs



Error = (target output - actual output)

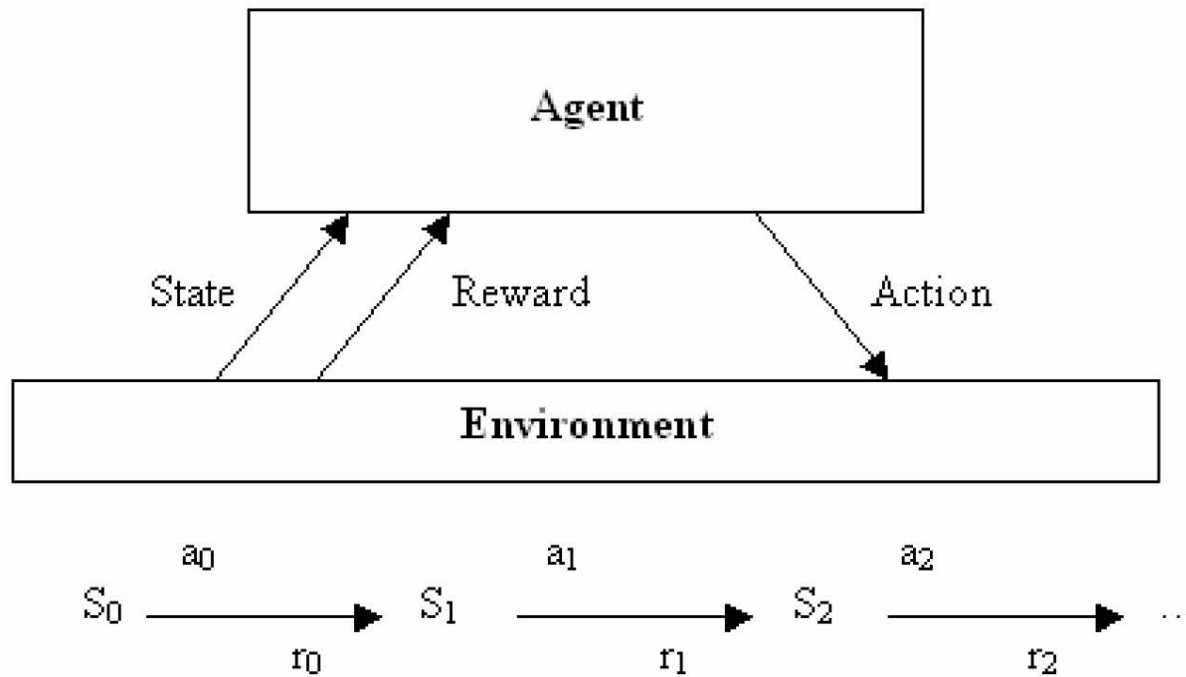
Reinforcement Learning

Training Info = evaluations (“rewards” / “penalties”)

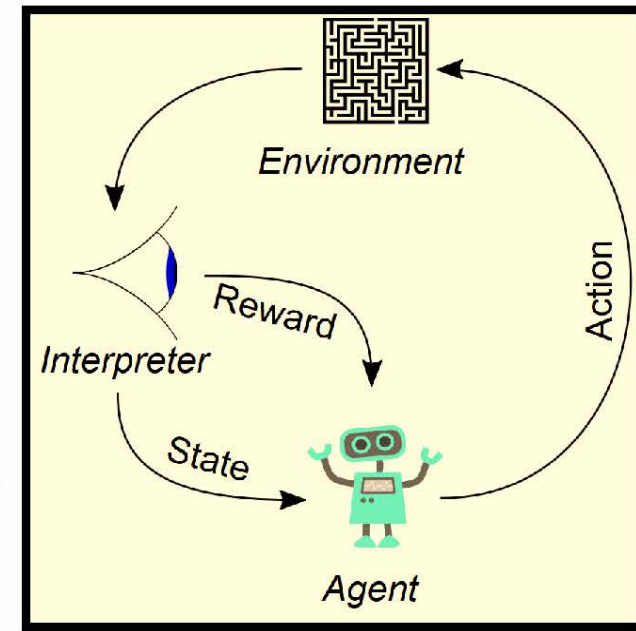


Objective: get as much reward as possible

The reinforcement learning problem: *State, Action and Reward*



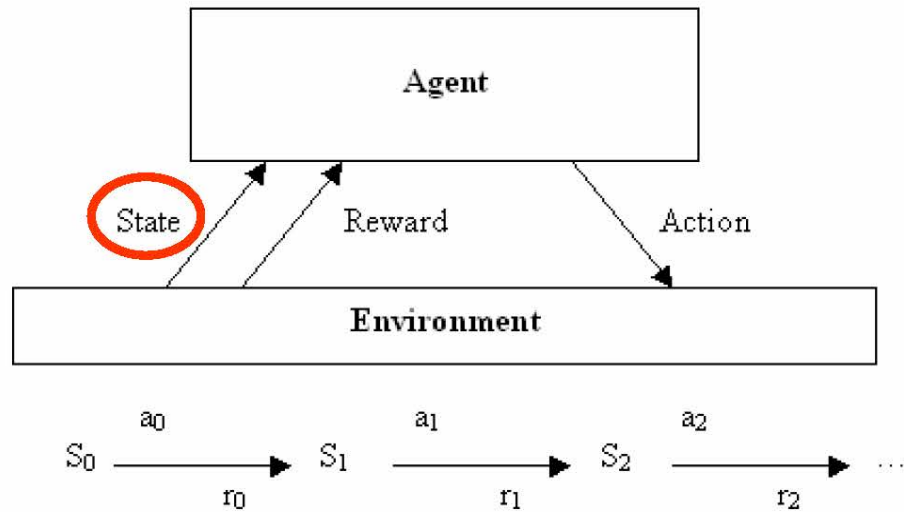
Goal: learn to choose actions that maximize:
 $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$, where $0 \leq \gamma < 1$



The reinforcement learning problem: *State, Action and Reward*



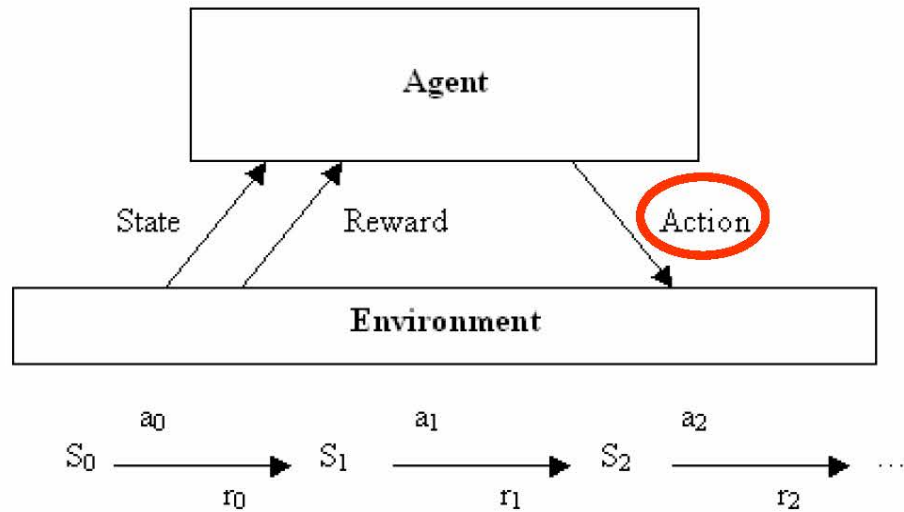
The reinforcement learning problem: *State, Action and Reward*



Goal: learn to choose actions that maximize:
 $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$, where $0 \leq \gamma < 1$



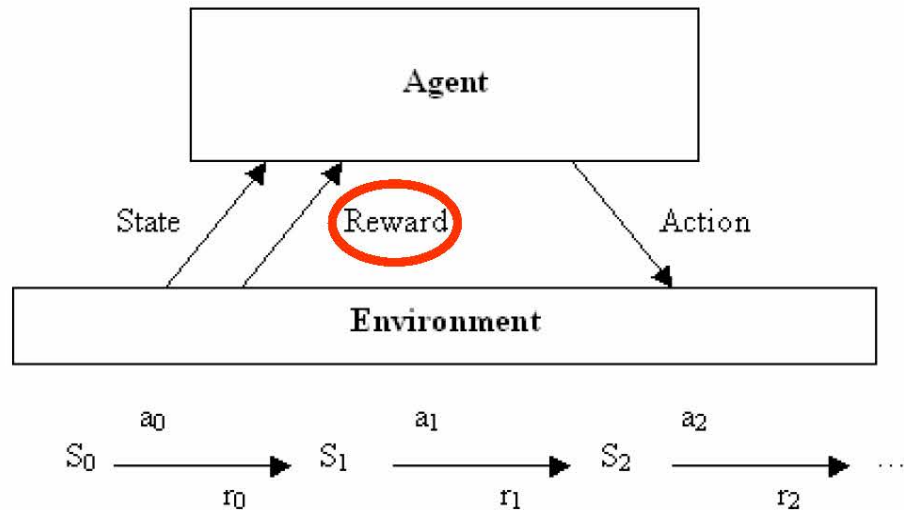
The reinforcement learning problem: *State, Action and Reward*



“Move piece from J1 to H1”

Goal: learn to choose actions that maximize:
 $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$, where $0 \leq \gamma < 1$

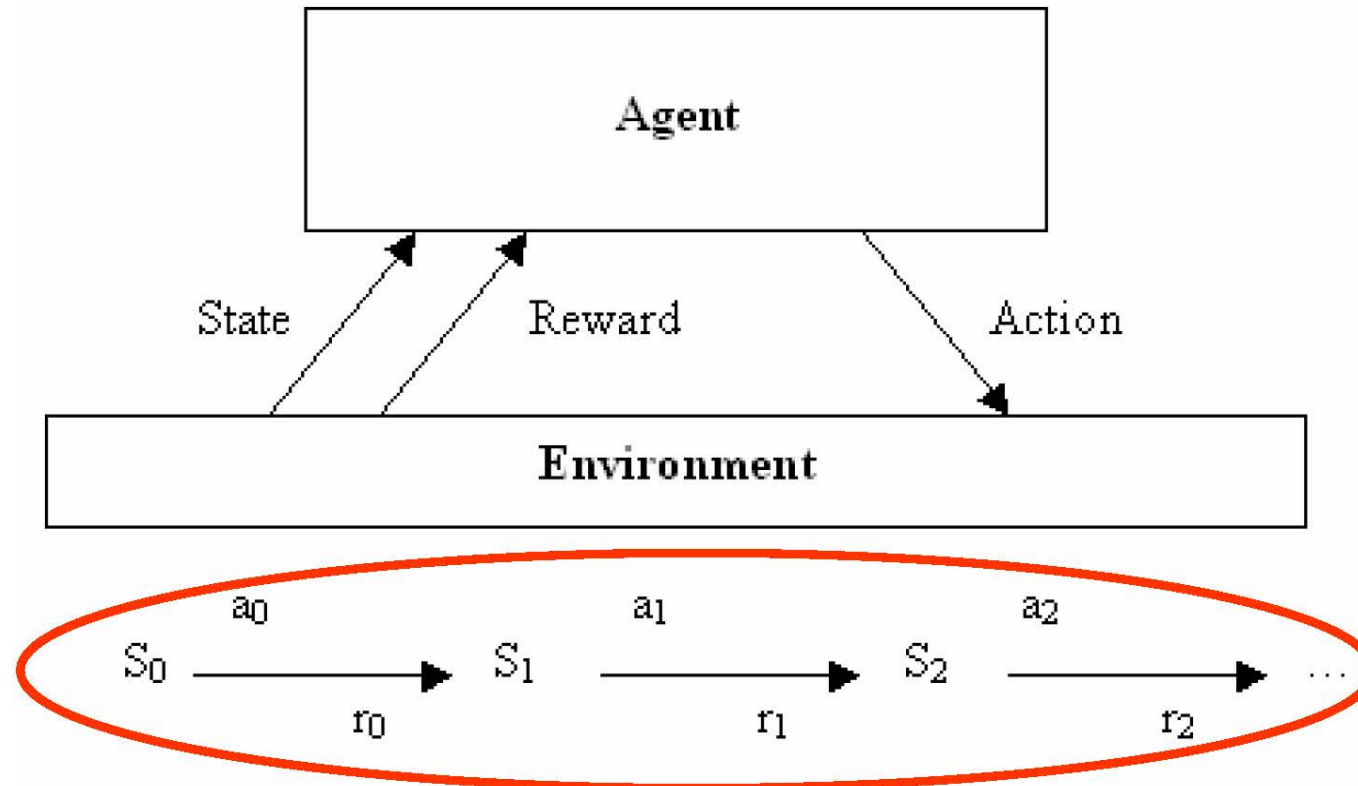
The reinforcement learning problem: *State, Action and Reward*



You took an opponent's piece.
Reward=1

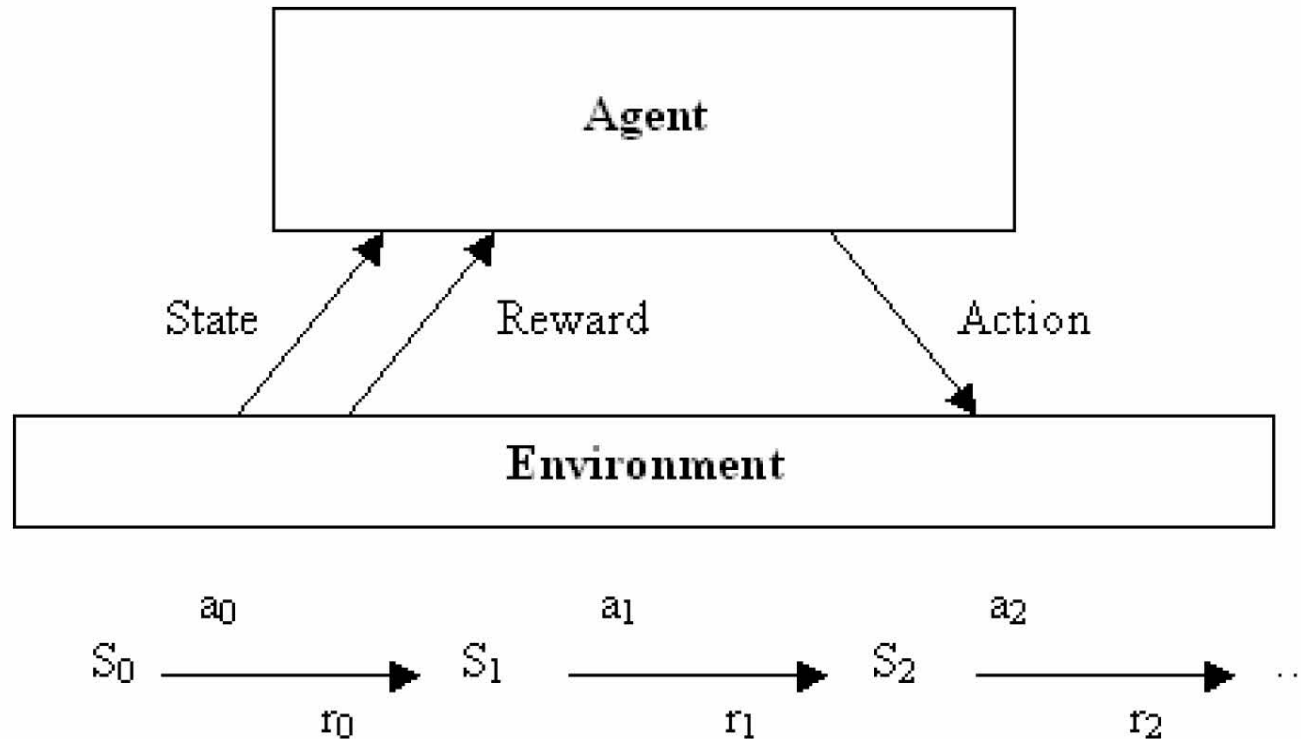
Goal: learn to choose actions that maximize:
 $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$, where $0 \leq \gamma < 1$

The reinforcement learning problem: *State, Action and Reward*



Goal: learn to choose actions that maximize:
 $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$, where $0 \leq \gamma < 1$

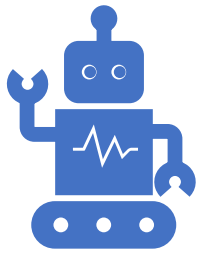
The reinforcement learning problem: *State, Action and Reward*



Goal: learn to choose actions that maximize:
 $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$, where $0 \leq \gamma < 1$



UiO : **University of Oslo**



IN3050/IN4050, Lecture 12

Reinforcement learning

2: Reward and action selection

Weria Khaksar

Next video: Policy and value

The reinforcement learning problem:

State, Action and Reward

Learning is guided by the reward

- An infrequent numerical feedback indicating how well we are doing.
- Problems:
 - The reward does not tell us *what we should have done!*
 - The reward may be *delayed* – does not always indicate *when we made a mistake.*

The reinforcement learning problem:

The reward Function

- Corresponds to the fitness function of an evolutionary algorithm.
- r_{t+1} is a function of (s_t, a_t) .
- The reward is a numeric value. Can be negative (“punishment”).
- Can be given throughout the learning episode, or only in the end.
- Goal: Maximize total reward.

The reinforcement learning problem:

Maximizing total reward

- Total reward:

$$R = \sum_{t=0}^{N-1} r_{t+1}$$

Future rewards may be uncertain and we might care more about rewards that come soon. Therefore, we discount future rewards:

$$R = \sum_{t=0}^{\infty} \gamma^t \cdot r_{t+1}, \quad 0 \leq \gamma \leq 1$$

or

$$R = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1}, \quad 0 \leq \gamma \leq 1$$

The reinforcement learning problem:

Maximizing total reward

- Future reward:

$$R = r_1 + r_2 + r_3 + \dots + r_n$$

$$R_t = r_t + r_{t+1} + r_{t+2} + \dots + r_n$$

- Discount future rewards (environment is stochastic)

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^{n-t} r_n$$

$$= r_t + \gamma (r_{t+1} + \gamma (r_{t+2} + \dots))$$

$$= r_t + \gamma R_{t+1}$$

- A good strategy for an agent would be to always choose an action that **maximizes the (discounted) future reward**.

The reinforcement learning problem:

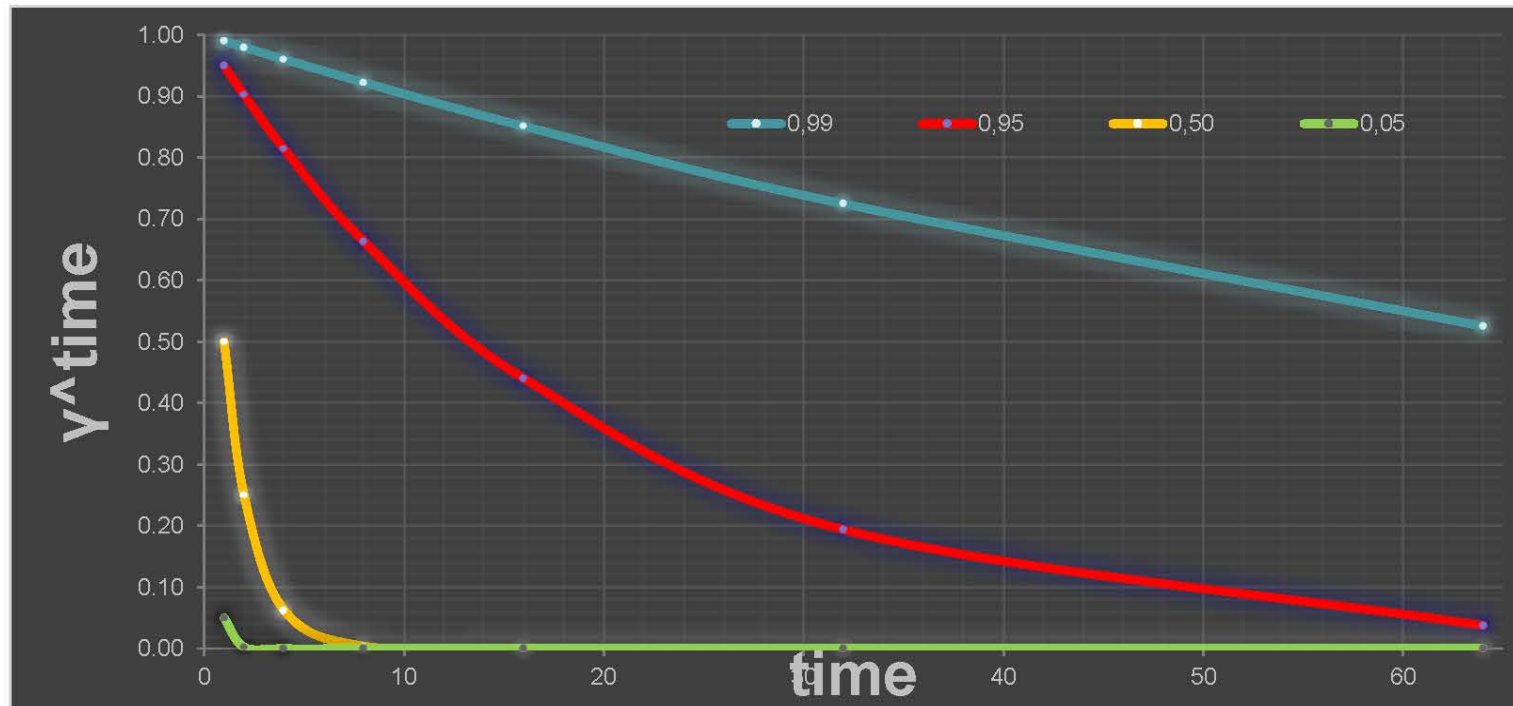
Discounted rewards example

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1}, \quad 0 \leq \gamma \leq 1$$

t	0.99^t	0.95^t	0.50^t	0.05^t
1	0,990000	0,950000	0,500000	0,050000
2	0,980100	0,902500	0,250000	0,002500
4	0,960596	0,814506	0,062500	0,000006
8	0,922745	0,663420	0,003906	0,000000
16	0,851458	0,440127	0,000015	0,000000
32	0,724980	0,193711	0,000000	0,000000
64	0,525596	0,037524	0,000000	0,000000

The reinforcement learning problem: *Discounted rewards example*

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1}, \quad 0 \leq \gamma \leq 1$$



The reinforcement learning problem:

Action Selection

- At each learning stage, the RL algorithm looks at the possible actions and calculates the expected average reward.

$$Q_{s,t}(a)$$

- Based on $Q_{s,t}(a)$, an action will be selected using:

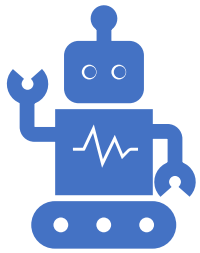
➤ **Greedy strategy:** pure exploitation

➤ **ϵ -Greedy strategy:** exploitation with a little exploration

➤ **Soft-Max strategy:** $P(Q_{s,t}(a)) = \frac{e^{(Q_{s,t}(a)/\tau)}}{\sum_b e^{(Q_{s,t}(b)/\tau)}}$



UiO : **University of Oslo**



IN3050/IN4050, Lecture 12

Reinforcement learning

3: Policy and value
Weria Khaksar

Next video: The Q-learning algorithm

The reinforcement learning problem:

Policy (π) and Value (V)

- The set of actions we took define our policy (π).
- The expected rewards we get in return, defines our value (V).

The reinforcement learning problem: *Markov Decision Process*

- If we only need to know the current state, the problem has the *Markov property*.



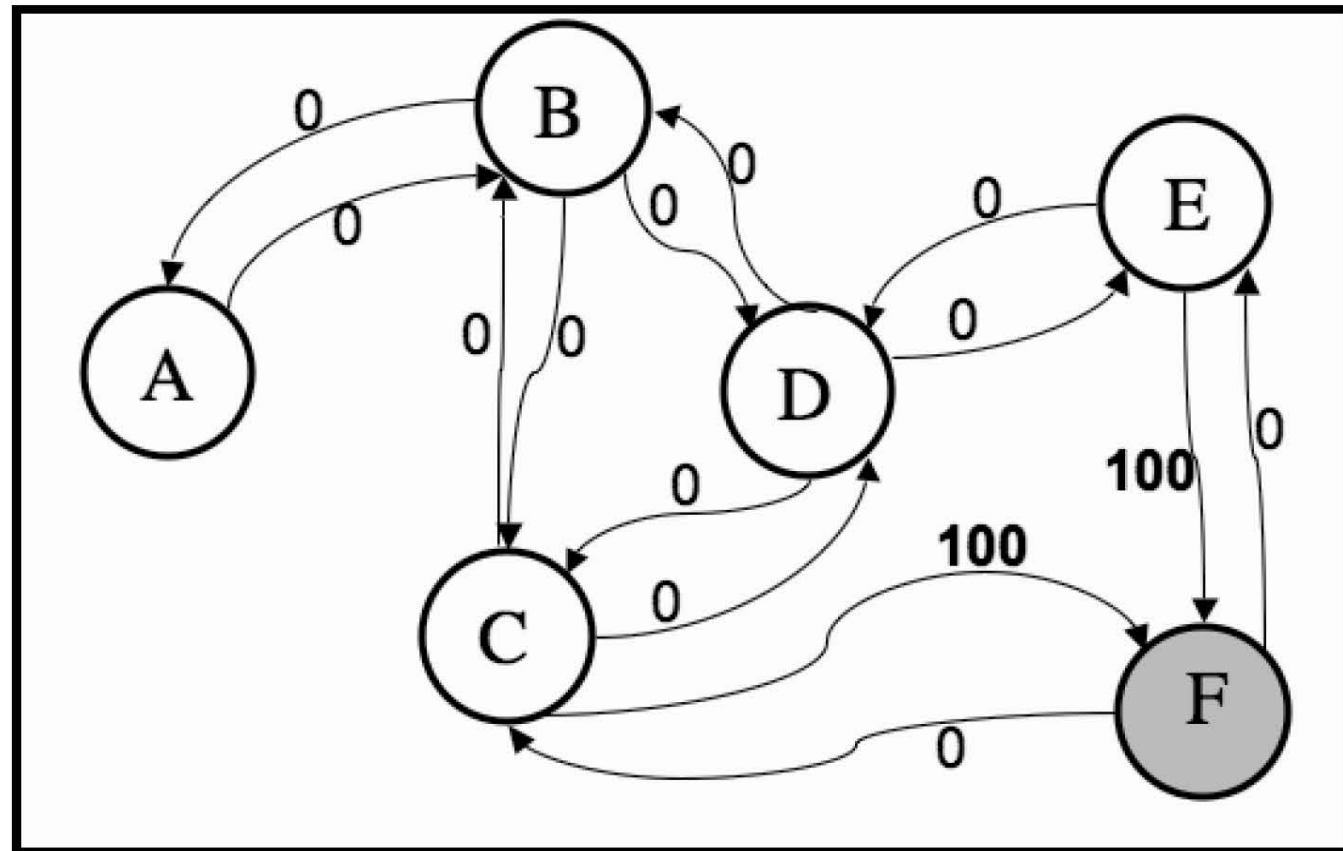
- No Markov Property:

$$P_r(r_t = r', s_{t+1} = s' | s_t, a_t, r_{t-1}, \dots, r_1, s_1, a_1, s_0, a_0)$$

- Markov Property:

$$P_r(r_t = r', s_{t+1} = s' | s_t, a_t)$$

The reinforcement learning problem: *Markov Decision Process*



A simple example of a Markov Decision Process

The reinforcement learning problem:

Value

- The expected future reward is known as the **value**.
- Two ways to compute the value:
 - The value of a state, $V(s)$, averaged over all possible actions in that state.
(state-value function)

$$V(s) = E(r_t | s_t = s) = E \left\{ \sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i+1} | s_t = s \right\}$$

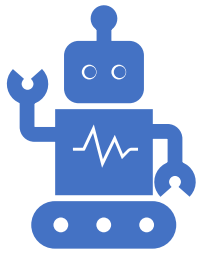
- The value of a state/action pair $Q(s, a)$. **(action-value function)**

$$Q(s, a) = E(r_t | s_t = s, a_t = a) = E \left\{ \sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i+1} | s_t = s, a_t = a \right\}$$

- Q and V are initially unknown, and learned iteratively as we gain experience.



UiO : **University of Oslo**



IN3050/IN4050, Lecture 12

Reinforcement learning

4: The Q-learning algorithm
Weria Khaksar

Next video: Q-learning example

The reinforcement learning problem: *The Q-Learning Algorithm*

The Q-Learning Algorithm

- Initialisation
 - set $Q(s, a)$ to small random values for all s and a
 - Repeat:
 - initialise s
 - repeat:
 - * select action a using ϵ -greedy or another policy
 - * take action a and receive reward r
 - * sample new state s'
 - * update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
 - * set $s \leftarrow s'$
 - For each step of the current episode
 - Until there are no more episodes
-

The reinforcement learning problem: *The Q-Learning Algorithm*

The Q-Learning Algorithm

- Initialisation

- set $Q(s, a)$ to small random values for all s and a

- Repeat:

- initialise s

- repeat:

- * select action a using ϵ -greedy or another policy
- * take action a and receive reward r
- * sample new state s'
- * update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
- * set $s \leftarrow s'$

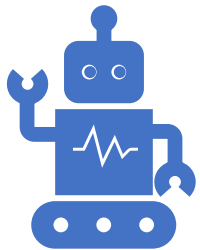
- For each step of the current episode

- Until there are no more episodes

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$



UiO : **University of Oslo**



IN3050/IN4050, Lecture 12

Reinforcement learning

5: Q-learning example

Kai Olav Ellefsen

Next video: On-policy and off-policy learning

Q-learning

- Values are learned by “backing up” values from the current state to the previous one:

$$\underbrace{Q(s_t, a_t)} \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

The diagram illustrates the Q-learning update rule. The left side shows the updated value $Q(s_t, a_t)$ being assigned to the current value. The right side shows the update calculation: the current value $Q(s_t, a_t)$ (labeled "old value") is added to the product of the learning rate μ and the difference between the current value and the target value. The target value is the sum of the reward r_{t+1} (labeled "reward") and the discounted maximum value of the next state, $\gamma \max_a Q(s_{t+1}, a)$ (labeled "estimate of optimal future value"). The term $\max_a Q(s_{t+1}, a)$ is also labeled "learned value".

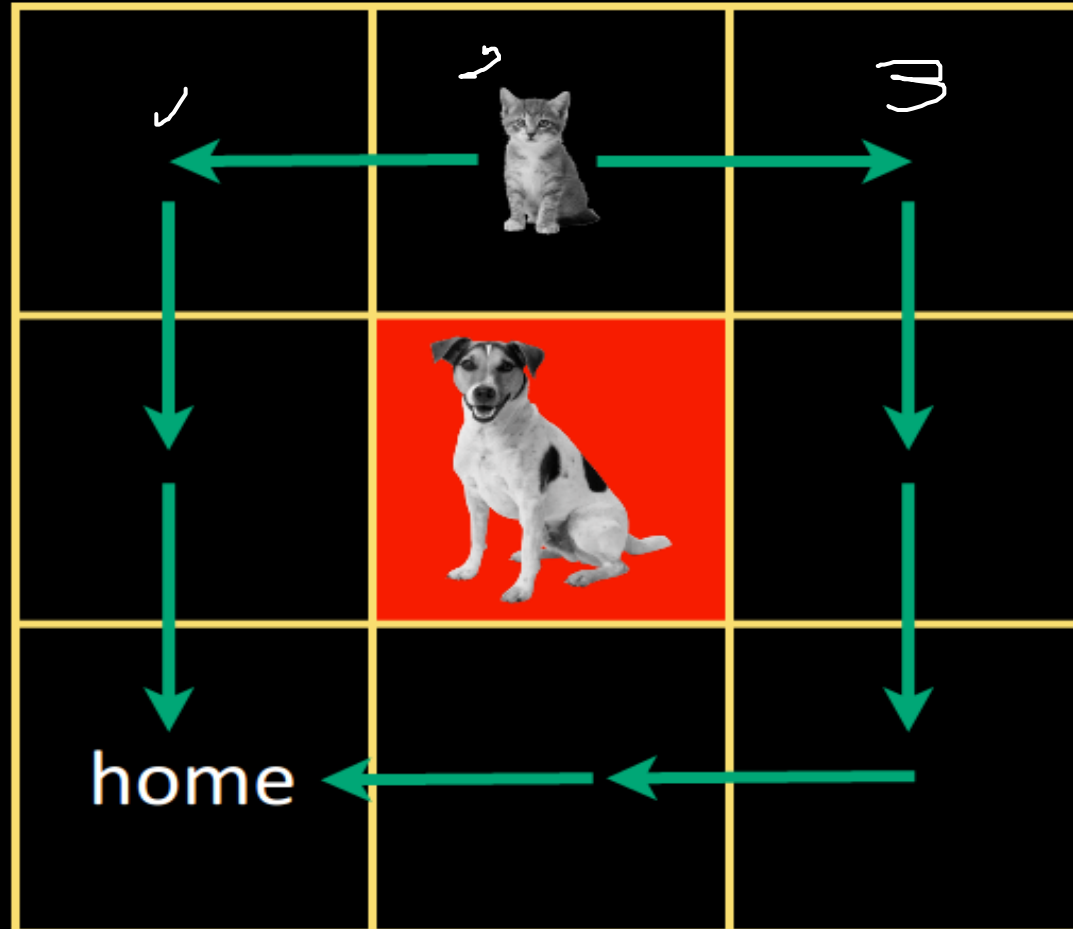
- The same can be done for v-values:

$$V(s_t) \leftarrow V(s_t) + \mu(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

Q-learning example

- Credits: Arjun Chandra

toy problem



expected long term value of taking
 some action in each state,
 under some action selection scheme?



$E\{R\}$	$E\{R\}$	$E\{R\}$
$E\{R\}$ $E\{R\}$	$E\{R\}$ $E\{R\}$	$E\{R\}$ $E\{R\}$
$E\{R\}$	$E\{R\}$	$E\{R\}$
$E\{R\}$	$E\{R\}$	$E\{R\}$
$E\{R\}$ $E\{R\}$	$E\{R\}$ $E\{R\}$	$E\{R\}$ $E\{R\}$
$E\{R\}$	$E\{R\}$	$E\{R\}$
$E\{R\}$	$E\{R\}$	$E\{R\}$
$E\{R\}$ $E\{R\}$	$E\{R\}$ $E\{R\}$	$E\{R\}$ $E\{R\}$
$E\{R\}$	$E\{R\}$	$E\{R\}$



our toy problem

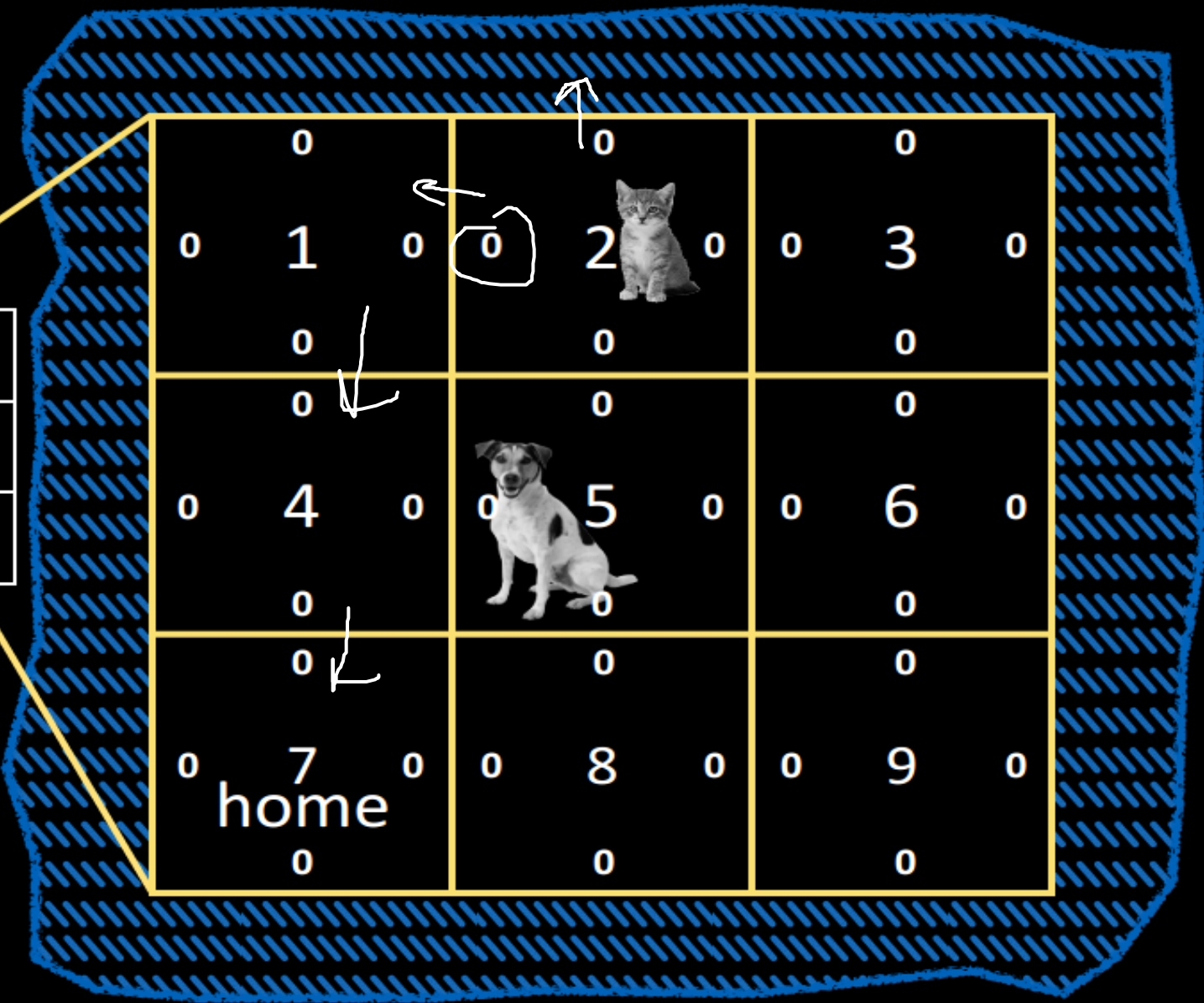
lookup table

		
		
home		

	0			0			0	
0	1	0	0	2 	0	0	3	0
	0			0			0	
	0			0			0	
0	4	0	0	5 	0	0	6	0
			0	0			0	
	0			0			0	
0	7	0	0	8	0	0	9	0
	home							
	0			0			0	



reward structure?



move...
to any cell except 5 and 7:
-1

out of bounds:
-5



to 5:
-10

to 7/home:
10

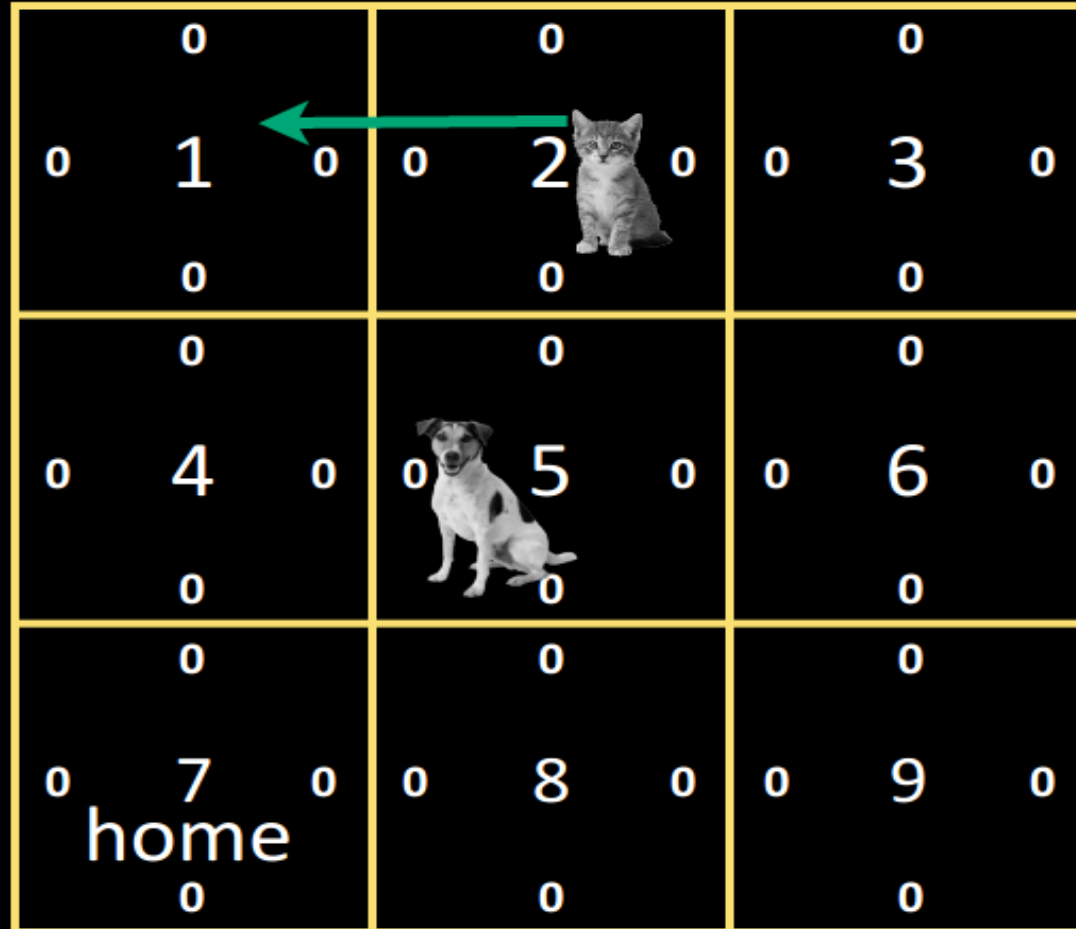


let's fix $\mu = 0.1$, $\gamma = 0.5$

		
		
home		

	0			0			0	
0	1	0	0	2 	0	0	3	0
	0			0			0	
	0			0			0	
0	4	0	0	5 	0	0	6	0
	0			0			0	
	0			0			0	
0	7	0	0	8	0	0	9	0
	0			0			0	





episode 1 begins...



0	0	0
0	1 	0
0	0	0
0	0	0
0	4	0
0		5
0	0	0
0	7	0
home	0	8
0	0	0
0	0	9
0	0	0

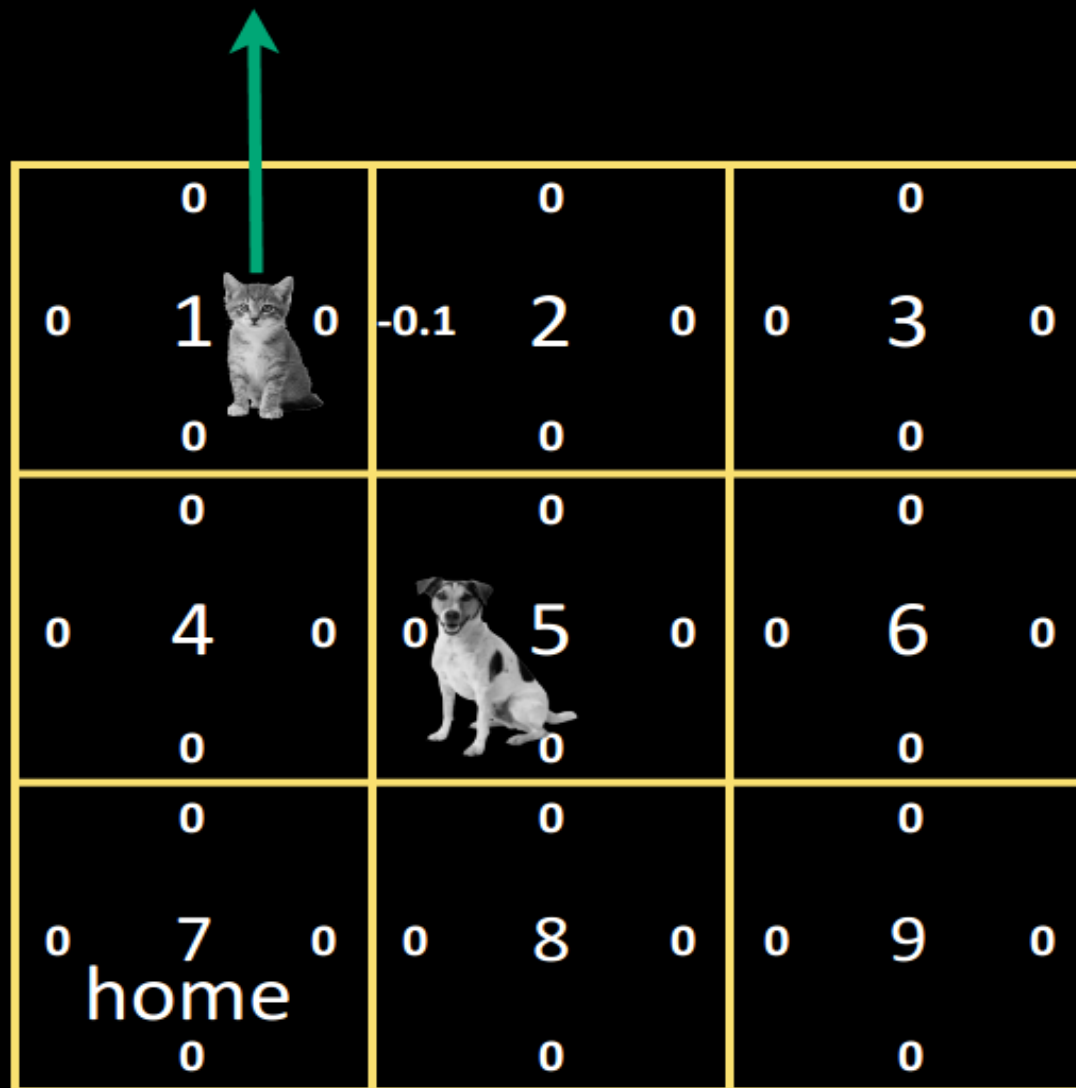
-1

$$\underbrace{-0.1}_{Q(s_t, a_t)} \leftarrow \underbrace{0}_{Q(s_t, a_t)}_{\text{old value}} + \underbrace{0.1}_{\mu}_{\text{learning rate}} \cdot \left(\underbrace{-1}_{\text{reward}} + \underbrace{0.5}_{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{0}_{Q(s_t, a_t)}_{\text{old value}} \right)$$



0	0	0
0	1  0	-0.1 2 0
0	0	0 3 0
0	0	0
0	4 0	0 5 0
0	0	0 6 0
0	0	0
0	7 0	0 8 0
0	home	0 9 0
0	0	0





0 0 1 0	0 -0.1 2 0	0 0 3 0
0 4 0	0 5 0	0 6 0
0 7 home 0	0 8 0	0 9 0

The table contains the following elements:



- Cell (1,1):** Contains the number 1 and a small image of a kitten. A green arrow points upwards from the top of this cell.
- Cell (1,2):** Contains the number 2.
- Cell (1,3):** Contains the number 3.
- Cell (2,1):** Contains the number 4.
- Cell (2,2):** Contains the number 5 and a small image of a dog.
- Cell (2,3):** Contains the number 6.
- Cell (3,1):** Contains the number 7 and the word "home".
- Cell (3,2):** Contains the number 8.
- Cell (3,3):** Contains the number 9.



?	0	0
0 1  0	-0.1 2 0	0 3 0
0	0	0
0	0	0
0 4 0	0  5 0	0 6 0
0	0	0
0 7 0	0 8 0	0 9 0
home	0	0

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{0.1}_{\text{learning rate}} \cdot \left(\underbrace{-0.5}_{\text{reward}} + \underbrace{0.5}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$





-0.5 0 1  0 0	0 -0.1 2 0 0	0 0 3 0 0
0 0 4 0 0	0  5 0 0	0 0 6 0 0
0 0 7 home 0 0	0 0 8 0 0	0 0 9 0 0



-0.5 0 1 0 0	0 -0.1 2 0 0	0 0 3 0 0
0 0 4 0 0	0 0 5 0 0	0 0 6 0 0
0 0 7 0 home 0	0 0 8 0 0	0 0 9 0 0

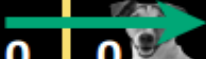


-0.5	0	0
0 1 0	-0.1 2 0	0 3 0
?	-1 0	0 0
0	0	0
0 4  0	0  5 0	0 6 0
0	0	0
0 7 0	0 8 0	0 9 0
home	0	0

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$



	-0.5		0		0		0	
0	1	0	-0.1	2	0	0	3	0
	-0.1		0		0		0	
0	4	0	0	5	0	0	6	0
	0						0	
0	7	0	0	8	0	0	9	0
	home						0	
	0						0	



-0.5	0	0
0 1 0	-0.1 2 0	0 3 0
-0.1	0	-10
0	0	0
0 4 ?	0 5 0	0 6 0
0	0	0
0 7	0 8 0	0 9 0
0 home	0	0

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$



-0.5	0	0
0 1 0	-0.1 2 0	0 3 0
-0.1	0	0
0	0	0
0 4 -1	0 5 0	0 6 0
0	0	0
0	0	0
0 7 0	0 8 0	0 9 0
home	0	0
0	0	0





-0.5	0	0
0 1 0	-0.1 2 0	0 3 0
-0.1	0	0
0	0	0
0 4 -1	0 5 0	0 6 0
0	0	0
0	0	0
0 7 0	0 8 0	0 9 0
home	0	0
0	0	0





-0.5	0	0
0 1 0	-0.1 2 0	0 3 0
-0.1	0	0
0	0	0
0 4 -1	0 5 0	0 6 0
0	?	-1
0	0	0
0 7 0	0 8 0	0 9 0
home		
0	0	0

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$




-0.5	0	0
0 1 0	-0.1 2 0	0 3 0
-0.1	0	0
0	0	0
0 4 -1	0 5 0	0 6 0
0	 -0.1	0
0	0	0
0 7 0	0 8 0	0 9 0
home		
0	0	0



	-0.5		0		0		0	
0	1	0	-0.1	2	0	0	3	0
	-0.1		0		0		0	
0	4	-1	0	5	0	0	6	0
	0			-0.1			0	
0	0		0		0		0	
0	7	0	0	8	0	0	9	0
	home						0	
	0		0		0		0	



	-0.5		0		0		0	
0	1	0	-0.1	2	0	0	3	0
	-0.1		0		0		0	
0	4	-1		5	0	0	6	0
	0		-0.1		0		0	
0	7	0	?	8	0	0	9	0
	home							
	0		10		0		0	
	0		0		0		0	

$$\begin{aligned}
 & 1.0 \\
 Q(s_t, a_t) & \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)
 \end{aligned}$$



let's work out the next
episode, starting at
state 4

go WEST and then SOUTH

how does the table change?



	-0.5		0		0		0	
0	1	0	-0.1	2	0	0	3	0
	-0.1		0		0		0	
-0.5	4	-1	0	5	0	0	6	0
	1		-0.1				0	
0	7	0	1	8	0	0	9	0
	0		0				0	



$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$



and the next episode,
starting at state 3

go WEST -> SOUTH -> WEST -> SOUTH

how does the table change?



	-0.5		0		0			
0	1	0	-0.1	2	0	-0.1	3	0
	-0.1		-1				0	
	0		0				0	
-0.5	4	-1	-0.05	5	0	0	6	0
1.0	1.9			-0.1			0	
	0		0				0	
0	7	0	1	8	0	0	9	0
	0		0				0	

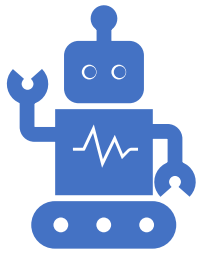
$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{0.1}_{\text{learning rate}} \cdot \left(\underbrace{-1}_{\text{reward}} + \underbrace{0.5}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

1.0 learned value
 0.5
 1.0





UiO : **University of Oslo**



IN3050/IN4050, Lecture 12

Reinforcement learning

6: On-policy and off-policy learning
Kai Olav Ellefsen

Action selection

- Estimate the *value* of each action: $Q_{s,t}(a)$
- Decide whether to:
 - Explore, or
 - exploit

	-0.5		0		0		0	
0	1	0	-0.1	2	0	-0.1	3	0
	-0.1		-1				0	
	0		0				0	
-0.5	4	-1	-0.05	5	0	0	6	0
	1.9		-0.1				0	
	0		0				0	
0	7	0	1	8	0	0	9	0
	0		0				0	



Action selection

- The function deciding which action to take in each state is called the policy, π . Examples:
 - Greedy: Always choose most valuable action
 - ϵ -greedy: Greedy, except small probability (ϵ) of choosing the action at random
- The q-learning we just saw is an example of off-policy learning.

$$\underbrace{Q(s_t, a_t)} \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

The diagram includes several annotations: a bracket under the left $Q(s_t, a_t)$ is labeled "old value"; a bracket under the μ is labeled "learning rate"; a bracket under the r_{t+1} is labeled "reward"; a bracket under the γ is labeled "discount factor"; a bracket under the $\max_a Q(s_{t+1}, a)$ is labeled "estimate of optimal future value"; a bracket under the right $Q(s_t, a_t)$ is labeled "old value"; a bracket above the entire term in parentheses is labeled "learned value".

Off-Policy Learning

The Q-Learning Algorithm

- **Initialisation**
 - set $Q(s, a)$ to small random values for all s and a
- Repeat:
 - initialise s
 - repeat:
 - * select action a using ϵ -greedy or another policy
 - * take action a and receive reward r
 - * sample new state s'
 - * update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
 - * set $s \leftarrow s'$
 - For each step of the current episode
- Until there are no more episodes

Source: Marsland

On-Policy Learning

The Sarsa Algorithm

- **Initialisation**
 - set $Q(s, a)$ to small random values for all s and a
- Repeat:
 - initialise s
 - choose action a using the current policy
 - repeat:
 - * take action a and receive reward r
 - * sample new state s'
 - * choose action a' using the current policy
 - * update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma Q(s', a') - Q(s, a))$
 - * $s \leftarrow s', a \leftarrow a'$
 - for each step of the current episode
- Until there are no more episodes



Off-Policy Learning

The Q-Learning Algorithm

- **Initialisation**

- set $Q(s, a)$ to small random values for all s and a

- Repeat:

- initialise s

- repeat:

- * select action a using ϵ -greedy or another policy
- * take action a and receive reward r
- * sample new state s'
- * update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
- * set $s \leftarrow s'$

- For each step of the current episode

- Until there are no more episodes

Source: Marsland

On-Policy Learning

The Sarsa Algorithm

- **Initialisation**

- set $Q(s, a)$ to small random values for all s and a

- Repeat:

- initialise s

- choose action a using the current policy

- repeat:

- * take action a and receive reward r
- * sample new state s'
- * choose action a' using the current policy
- * update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma Q(s', a') - Q(s, a))$
- * $s \leftarrow s', a \leftarrow a'$

- for each step of the current episode

- Until there are no more episodes



Off-Policy Learning

The Q-Learning Algorithm

- **Initialisation**

- set $Q(s, a)$ to small random values for all s and a

- Repeat:

- initialise s

- repeat:

- * select action a using ϵ -greedy or another policy

- * take action a and receive reward r

- * sample new state s'

- * update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$

- * set $s \leftarrow s'$

- For each step of the current episode

- Until there are no more episodes

Source: Marsland

On-Policy Learning

The Sarsa Algorithm

- **Initialisation**

- set $Q(s, a)$ to small random values for all s and a

- Repeat:

- initialise s

- choose action a using the current policy

- repeat:

- * take action a and receive reward r

- * sample new state s'

- * choose action a' using the current policy

- * update $Q(s, a) \leftarrow Q(s, a) + \mu(r + \gamma Q(s', a') - Q(s, a))$

- * $s \leftarrow s', a \leftarrow a'$

- for each step of the current episode

- Until there are no more episodes



On-policy vs off-policy learning

- Q-learning (off-policy):

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\mu}_{\text{learning rate}} \cdot \left(\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

The equation for Q-learning is shown with several annotations. A bracket above the term $r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a)$ is labeled "learned value". A bracket below $\max_a Q(s_{t+1}, a)$ is labeled "estimate of optimal future value".

- Sarsa (on-policy):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \mu [r_{t+1} + \gamma \underbrace{Q(s_{t+1}, a_{t+1})}_{\text{estimate of optimal future value}} - Q(s_t, a_t)]$$

The term $Q(s_{t+1}, a_{t+1})$ in the Sarsa equation is circled in the original image.

On-policy vs off-policy learning

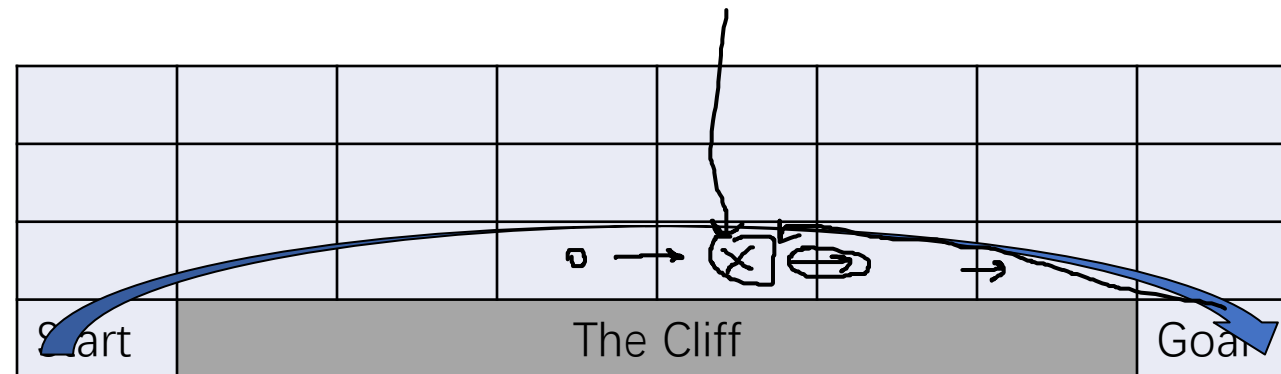
- Reward structure: Each move: -1. Move to cliff: -100.
- Policy: 90% chance of choosing best action (exploit). 10% chance of choosing random action (explore).

Start	-100	The Cliff					Goal



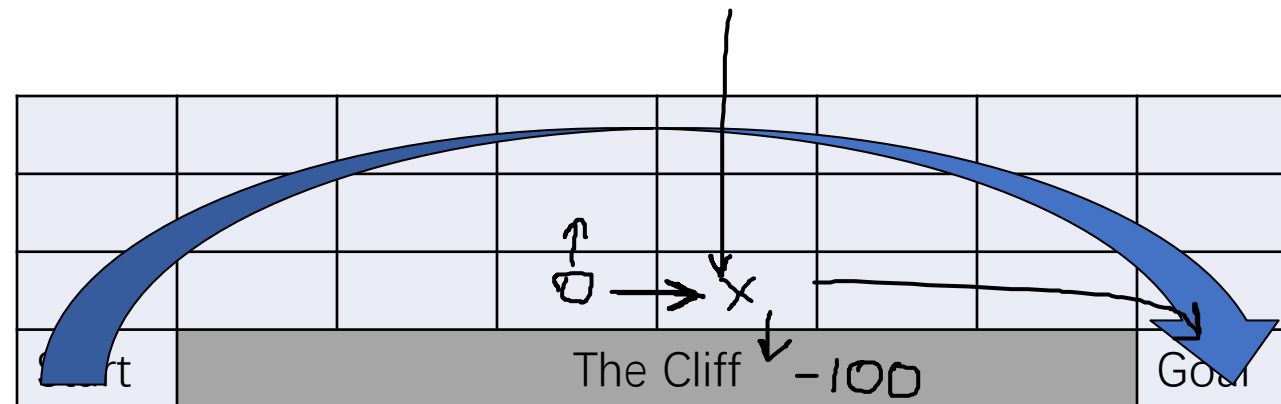
On-policy vs off-policy learning: Q-learning

- Always assumes optimal action -> does not visit cliff often while learning. Therefore, does not learn that cliff is dangerous.
- Resulting path is efficient, but risky.



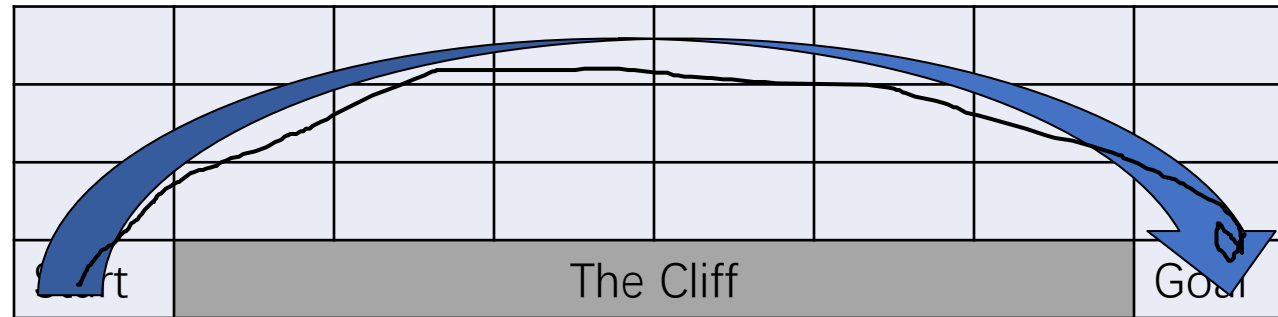
On-policy vs off-policy learning: sarsa

- During learning, we more frequently end up outside the cliff (due to the 10% chance of exploring in our policy).
- That info propagates to all states, generating a safer plan.

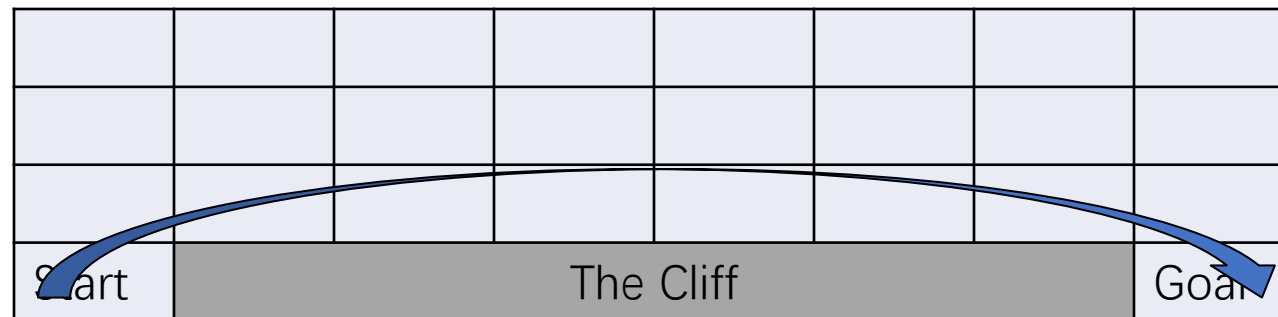


Which plan is better?

- sarsa (on-policy):

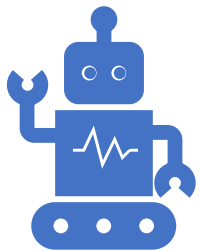


- Q-learning (off-policy):





UiO : **University of Oslo**



IN3050/IN4050, Lecture 12

Reinforcement learning

Weria Khaksar and Kai Olav Ellefsen