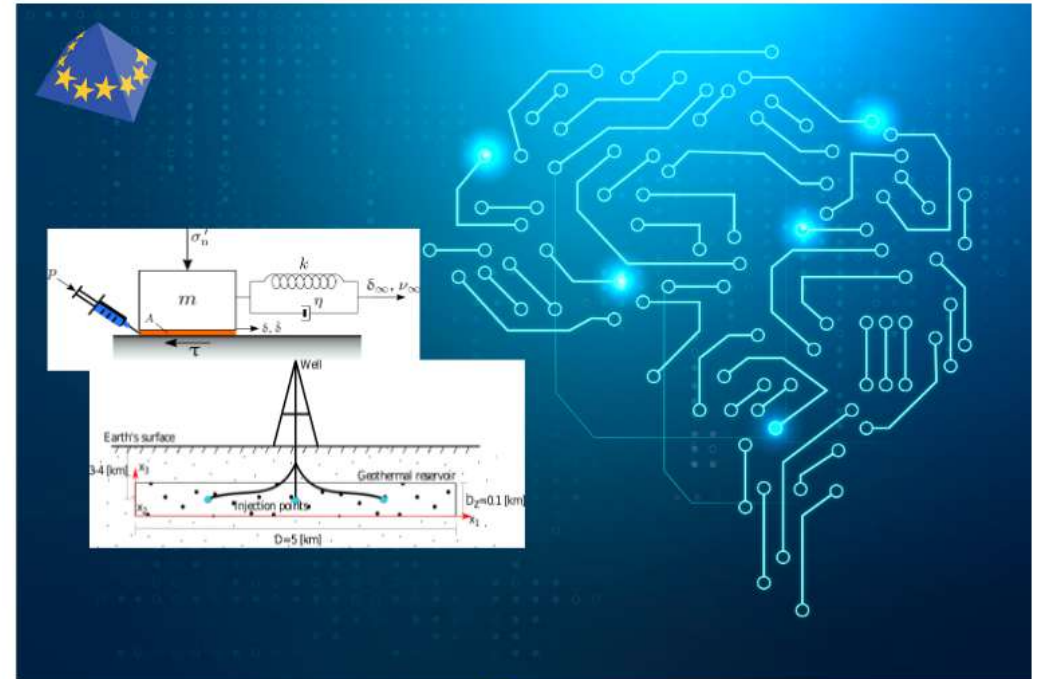


# Introduction to Reinforcement Learning with Applications in Geomechanics

Alexandros Stathas,  
Diego Gutierrez-Oribio,  
Ioannis Stefanou

**ALERT Doctoral School 2023**





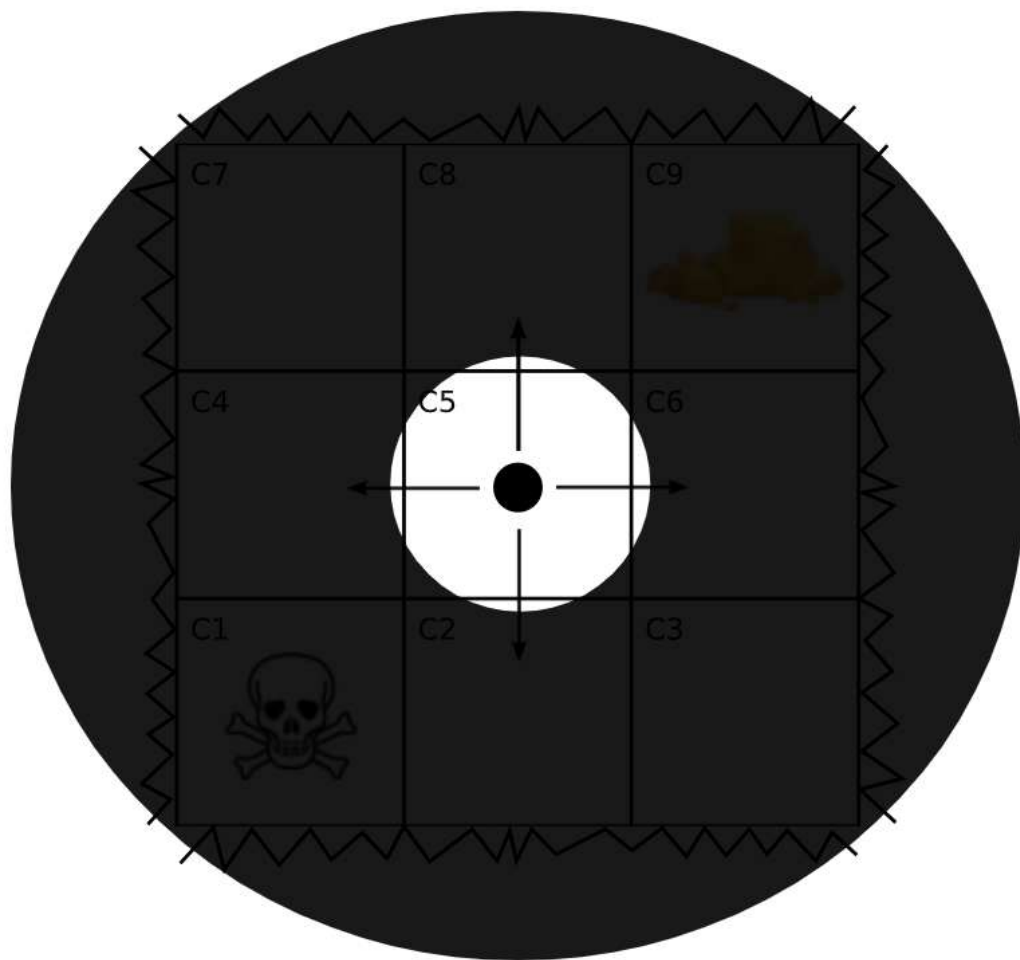
## Training a dog 101

**Example:** Ioannis gives a treat to Loupi when he sits down, fetches, etc...

**Summing up:** Use of **rewards** and/or **punishments** to encourage and/or discourage certain **actions**.

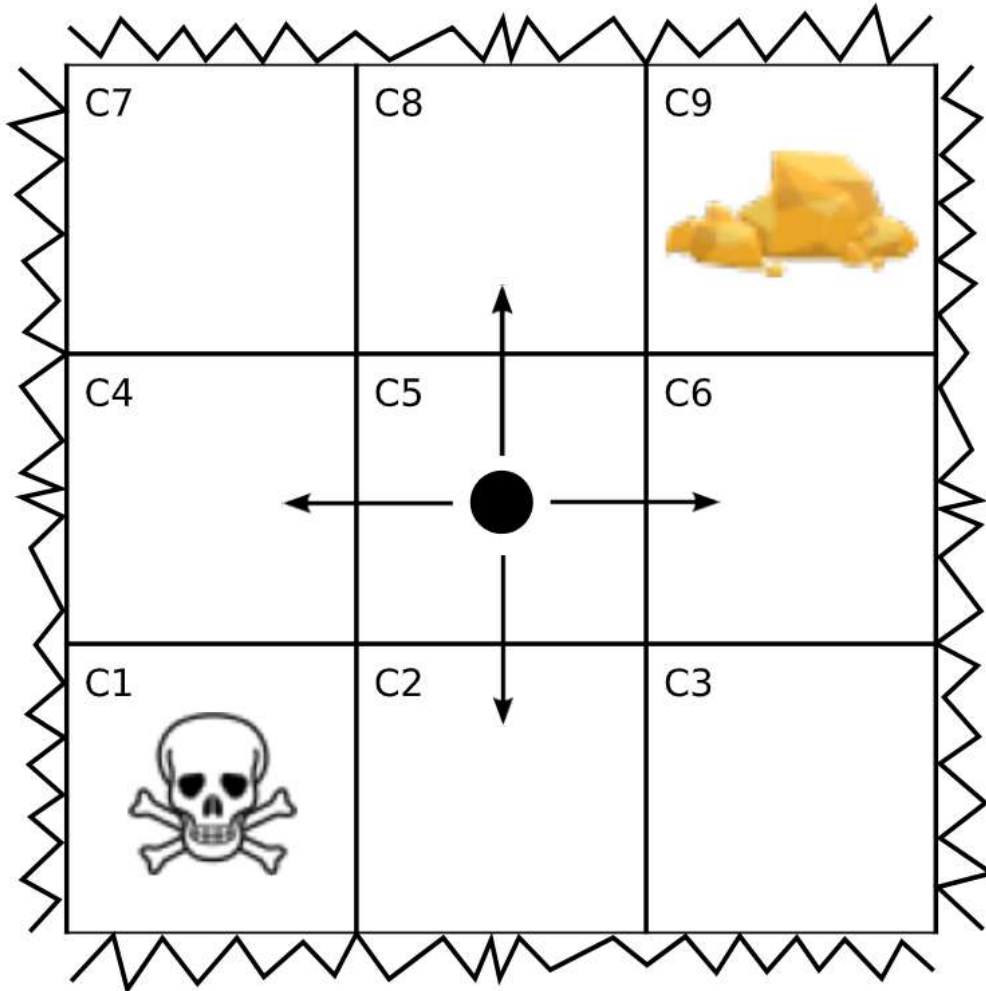
**Basic principle of RL:** Let a computer choose the actions to **maximize** the reward.

## **Part A: Main Concepts of Reinforcement Learning**



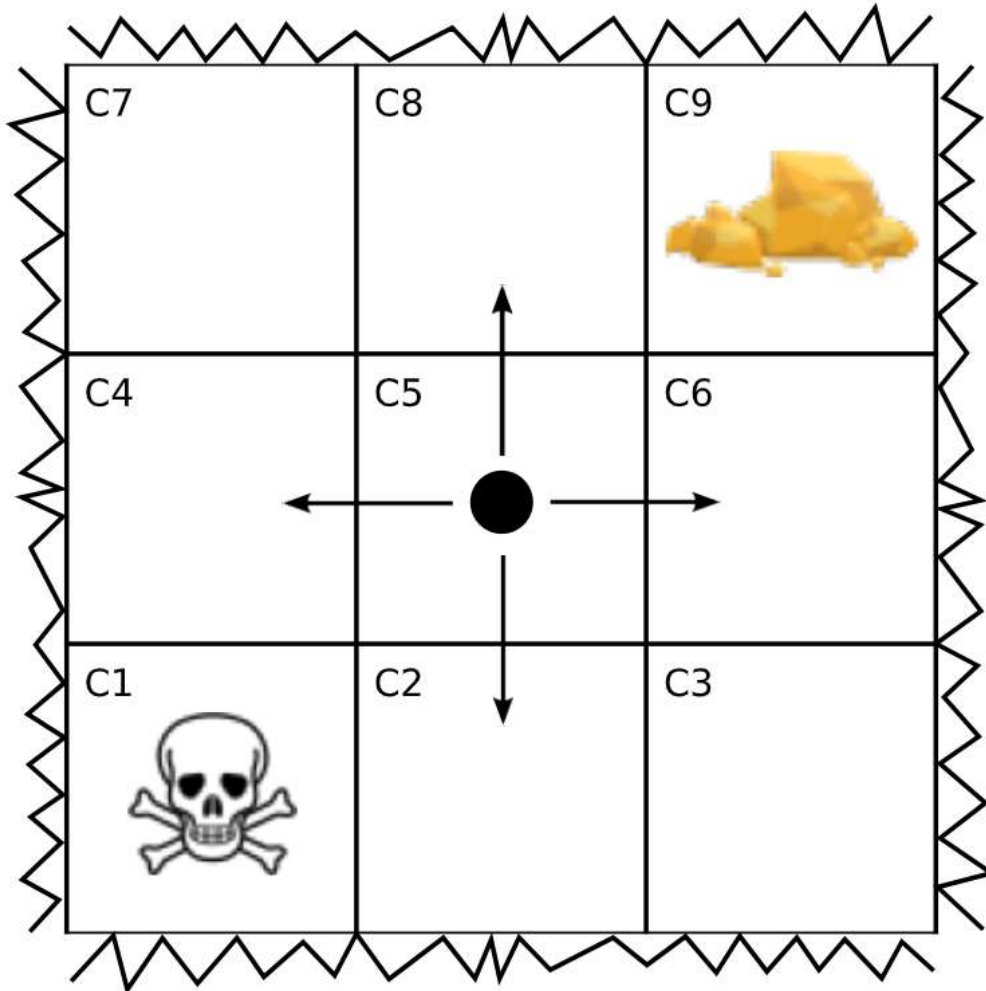
## The Miner

Once upon a time, a **miner** (black dot) was trapped in a very dark and dangerous **mine**...



## The Miner

The **miner** (black dot) moves inside the **mine** to find **gold** while avoiding the **cliff**. The miner can perform four **movements**: go right, go up, go left, and go down. The mine is constituted by 9 **cells** defined as C1, ..., C9.



## Game rules

Gold = +100 and game finishes (episode)

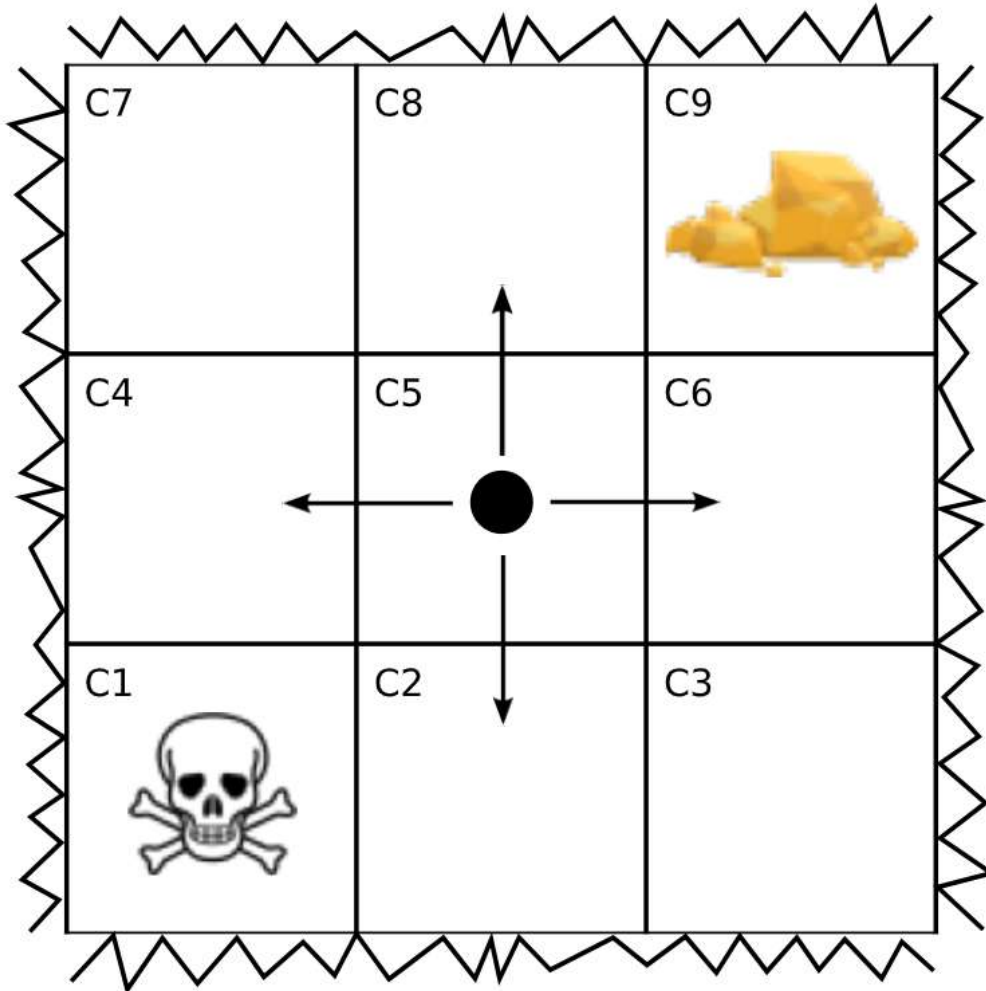
Death = -20 and game finishes

Wall = -10

Max steps = Game finishes

**Goal → Get to the gold**

Deepnote: <https://rb.gy/twfez>



## Agent (The miner)

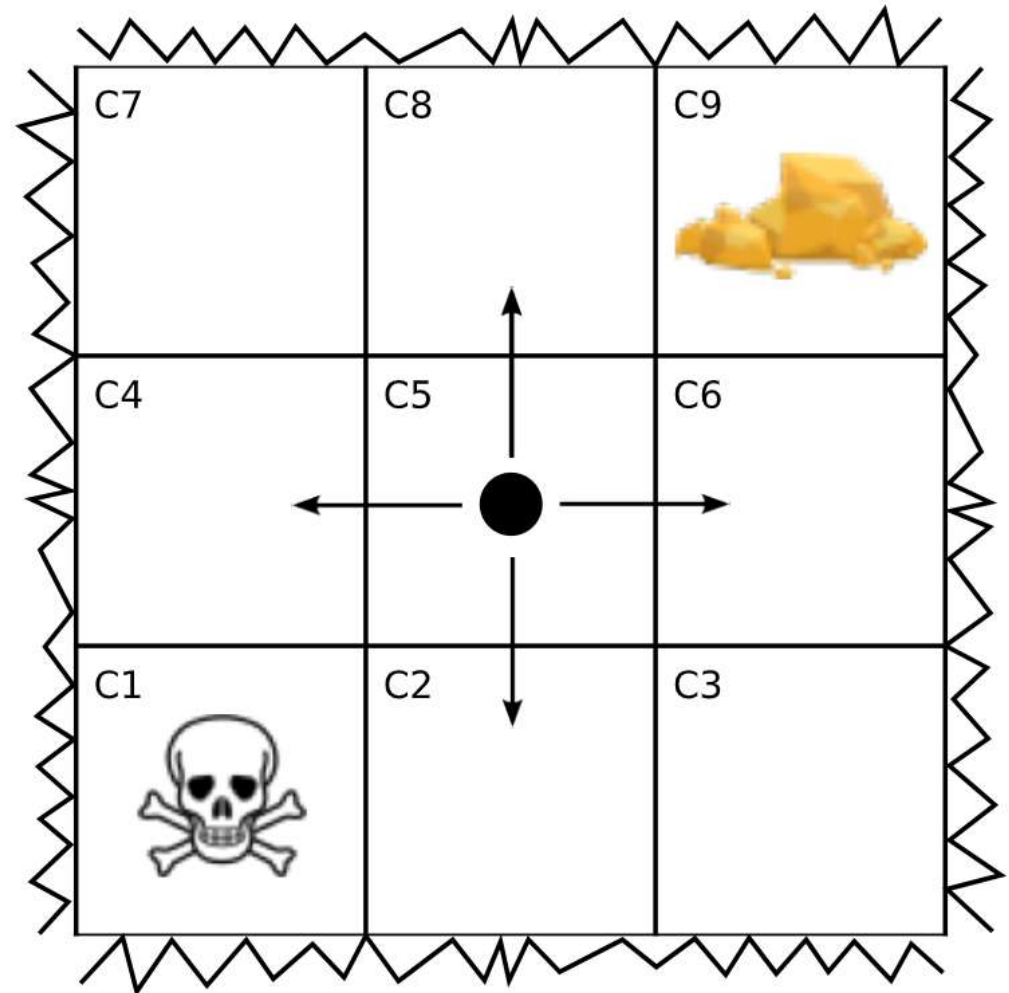
Is the element that can perceive its environment, take actions, and learn from its experiences to achieve specific goals.



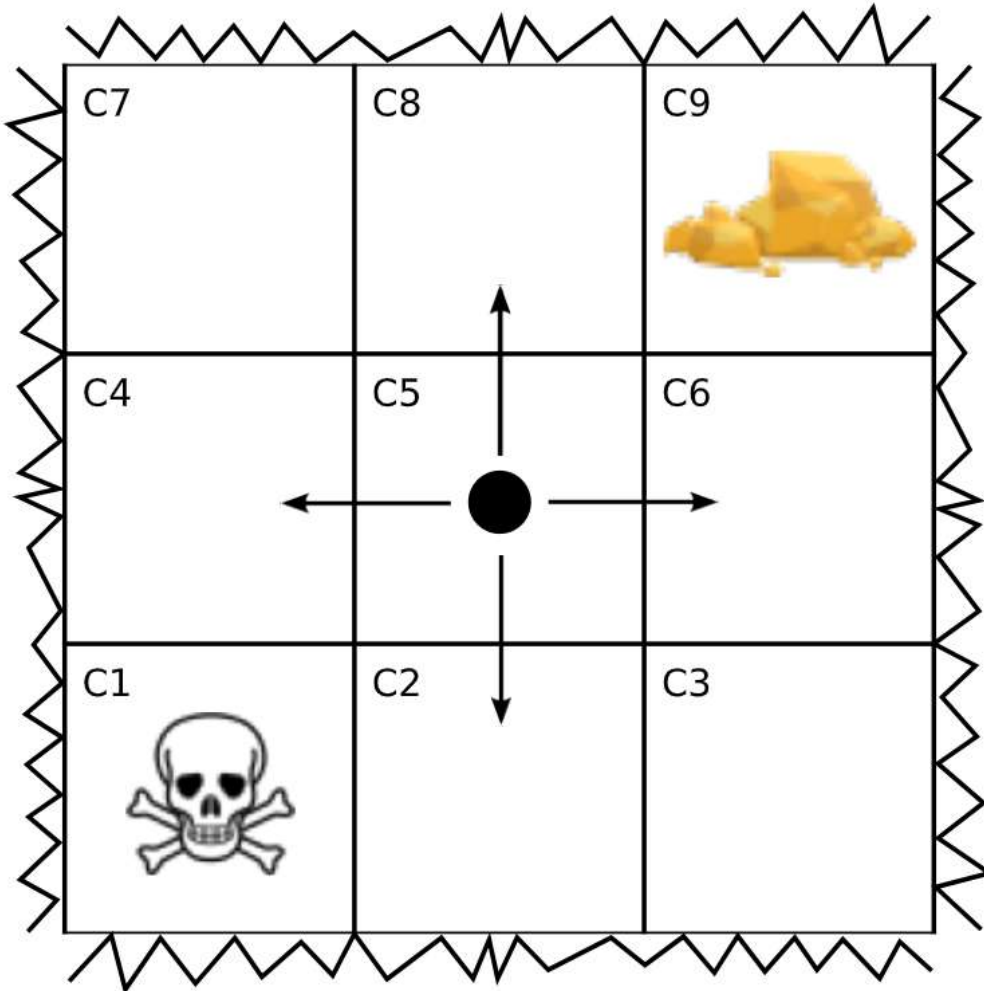
# Environment

(The mine)

Is the external context or surroundings in which an agent operates, interacts, and learns.







## Action $\alpha_t$

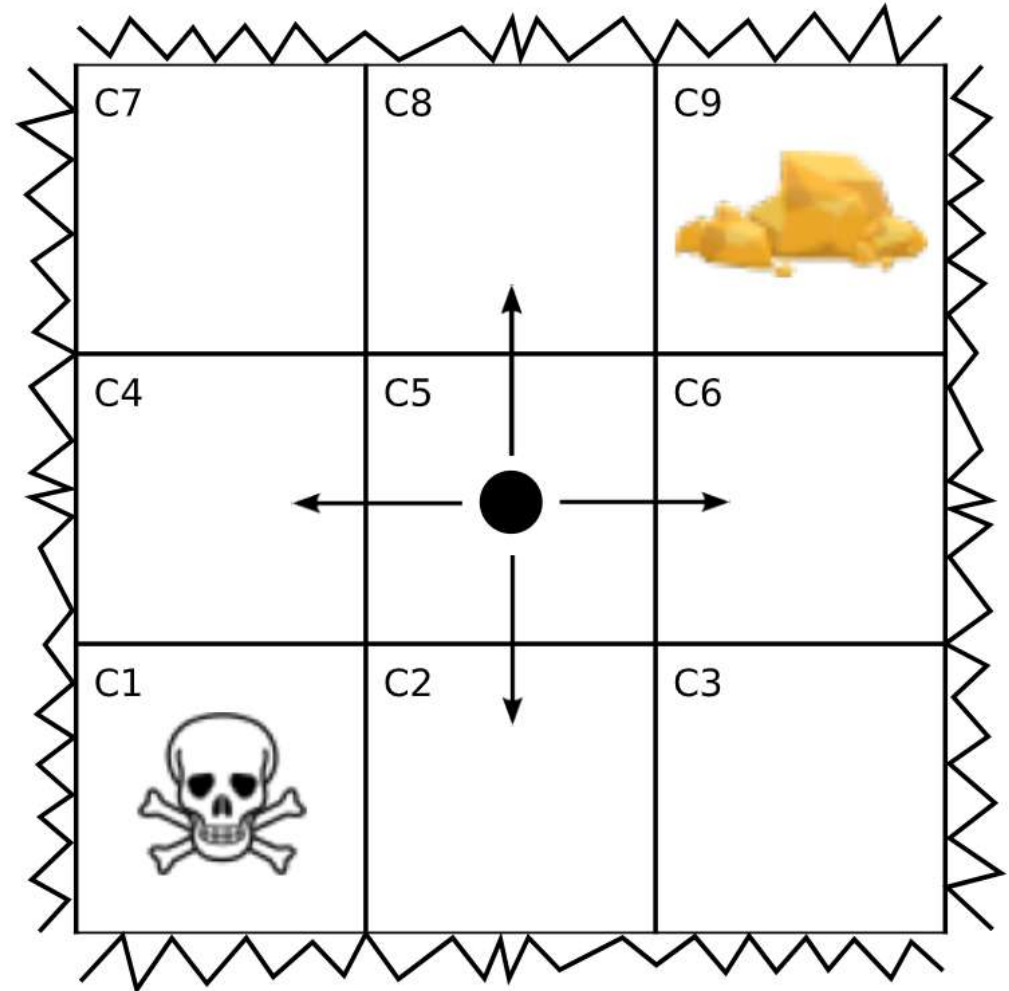
( $\rightarrow$ ,  $\uparrow$ ,  $\leftarrow$ ,  $\downarrow$ )

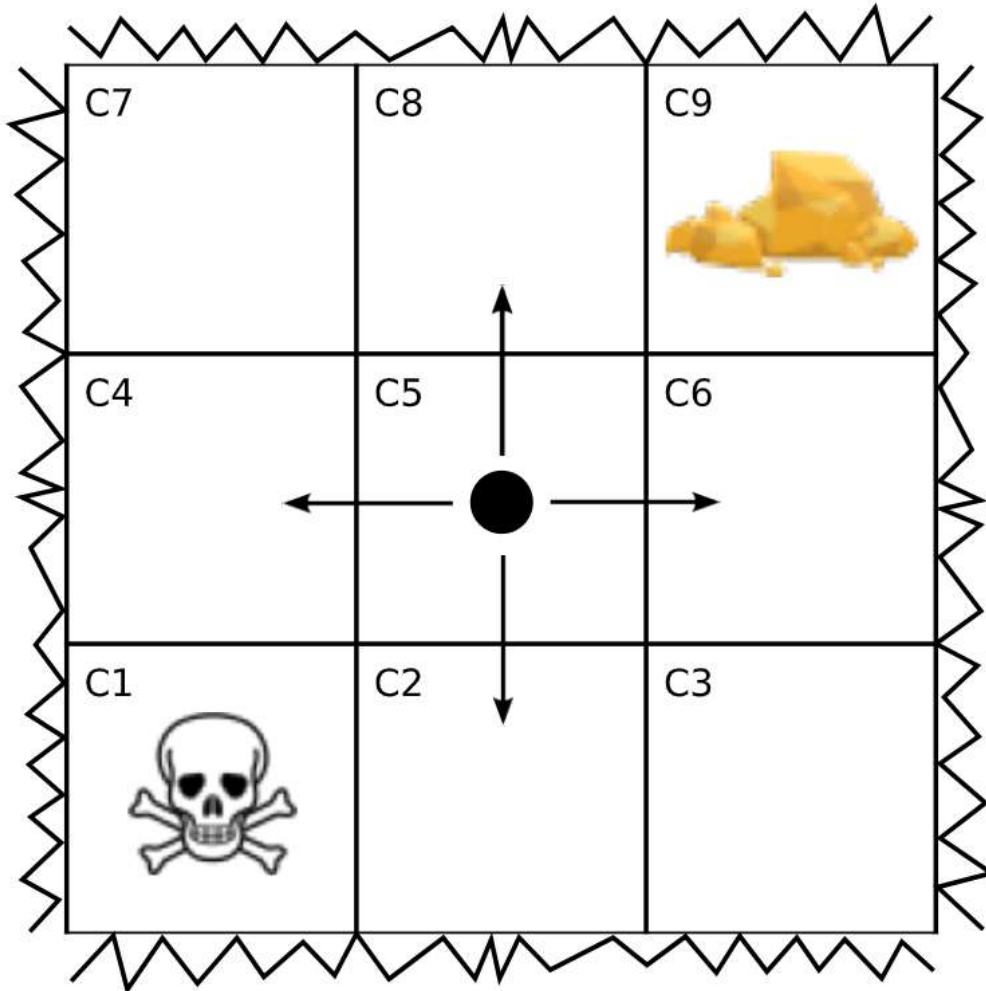
Specific move or decision that an agent takes in response to the information it receives from its environment.

## State $s_t$

(C1,...,C9)

A complete representation of the current situation or configuration of the environment, which the agent uses to make decisions and take actions.





## Reward and penalty $r_t$

(Gold, wall and death)

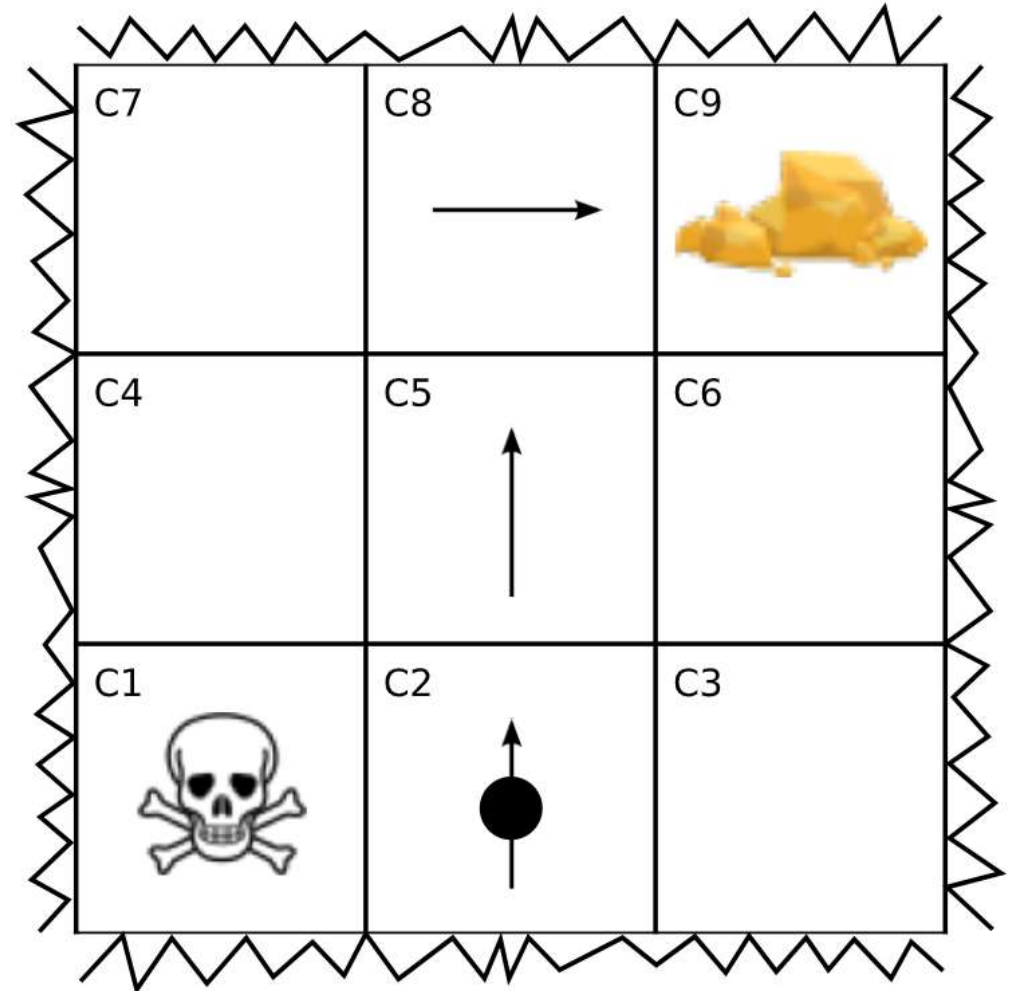
A reward is a positive feedback or signal given to an agent for taking desirable actions, while a penalty is a negative feedback or signal for undesirable actions.

## Trajectory

$$T_r = \{s_0, \alpha_0, r_1, s_1, \alpha_1, \dots, s_{T-1}, \alpha_{T-1}, r_T\}$$

For this case:

$$T_r = \{C_2, \uparrow, 0, C_5, \uparrow, 0, C_8, \rightarrow, 100\}$$



**Transition function**  $s_{t+1} = f(s_t, \alpha_t)$

Relates the next state of the agent when it performs a specific action from a current state.

**Reward function**  $r_{t+1} = R(s_{t+1}) = R(f(s_t, \alpha_t))$

Relates the next reward of the agent when it performs a specific action from a current state.

**These two are the game rules: How the miner moves and how it gets rewards.**

## Model-based RL

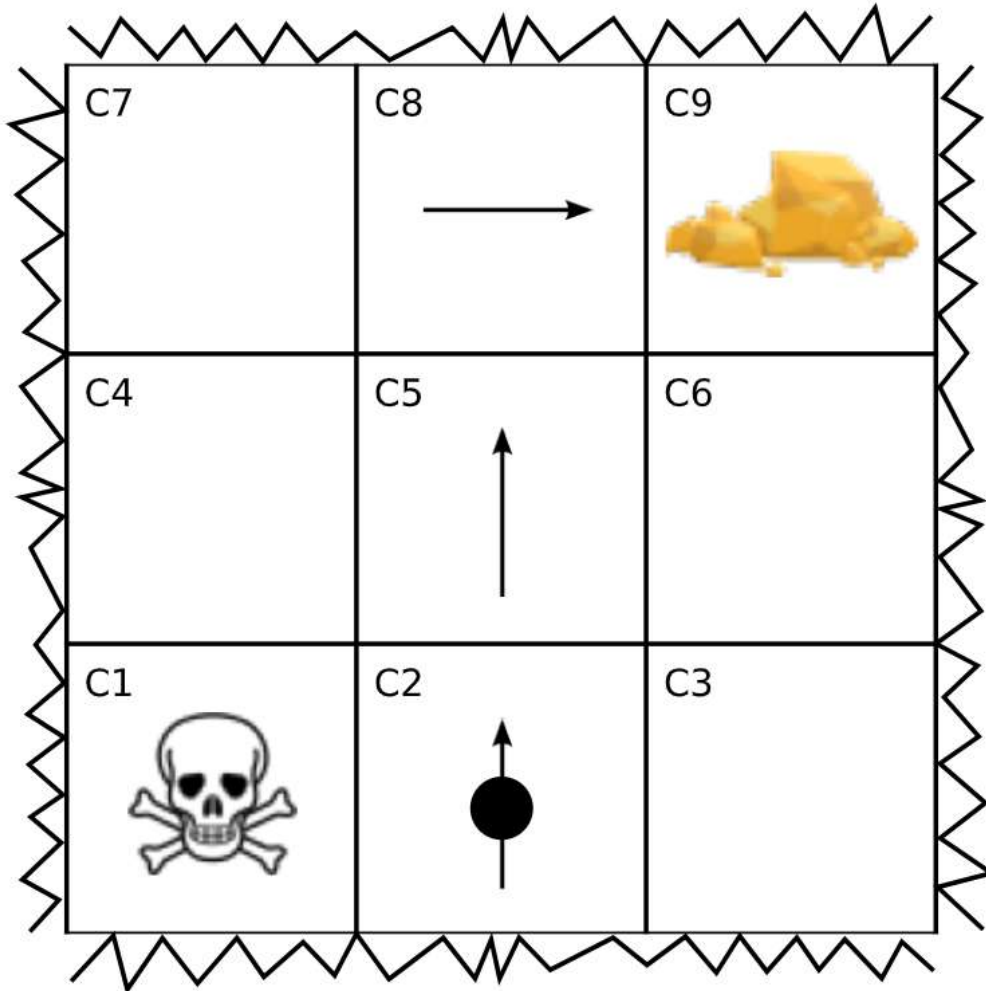
The transition function,  $f(s_t, \alpha_t)$ , is known

**Result:** Learning on every state,  $s_t$ , of the environment

## Free-model RL

The transition function,  $f(s_t, \alpha_t)$ , is unknown

**Result:** Learning on some states,  $s_t$ , of the environment



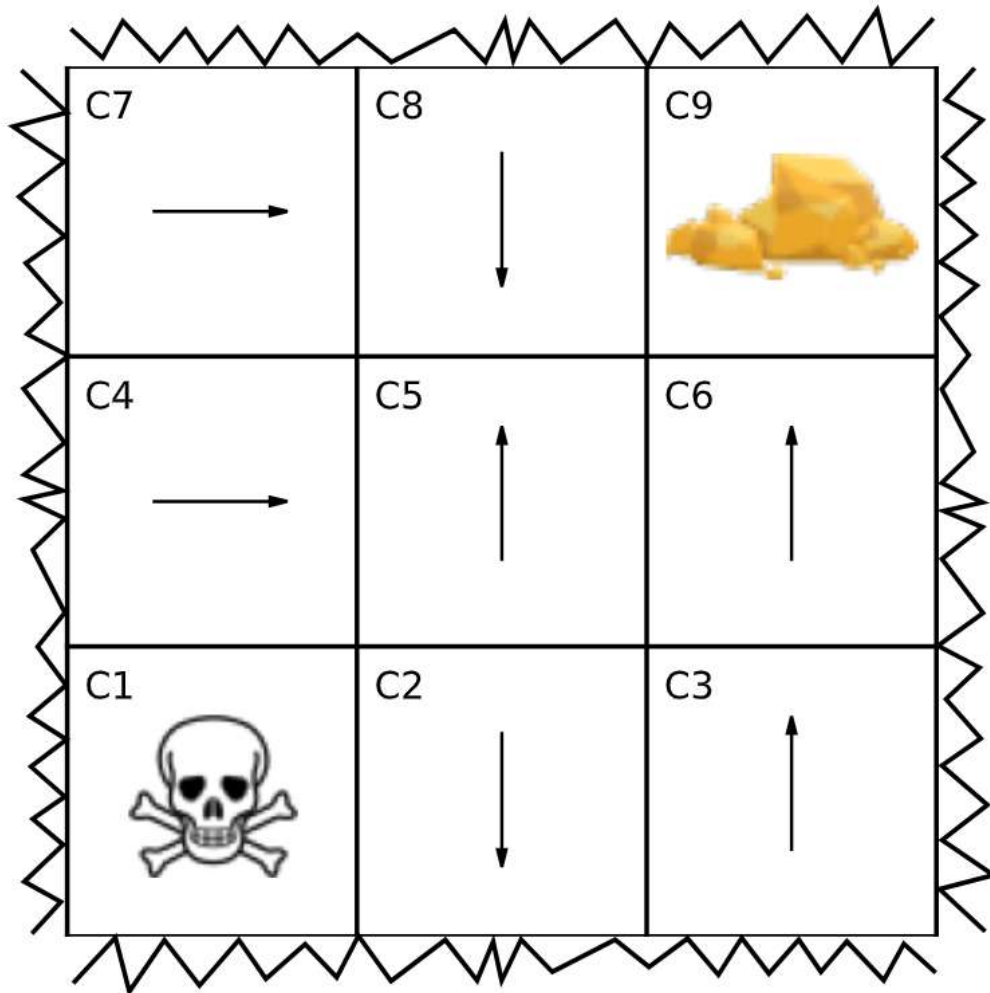
## Accumulated reward $G$

$$G = \sum_{k=0}^{T-1} r_{k+1}$$

In this trajectory:

$$G = 0 + 0 + 100 = 100$$





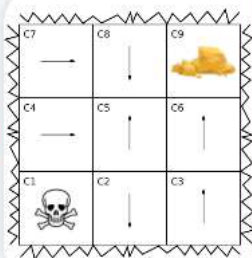
## Policy

Mapping between states and actions:

$$\alpha_t = \pi(s_t)$$

Which of the following policies do you think is the best to reach the gold?

1

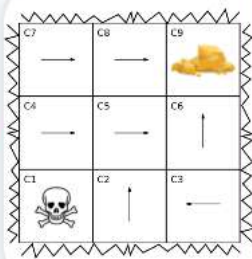


$\pi_A(s_t)$

0%

0 🧑

2

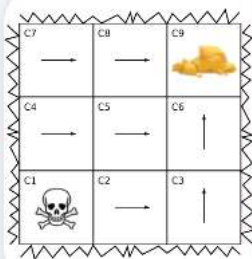


$\pi_B(s_t)$

0%

0 🧑

3



$\pi_C(s_t)$

0%

0 🧑

## Value of the state $V_{\pi}(s_t)$

$$V_{\pi}(s_t) = \sum_{k=0}^{T-t-1} r_{t+k+1}(s_{t+k}, \pi(s_{t+k}))$$

or recursively (**Bellman equation**):

$$V_{\pi}(s_t) = r_{t+1}(s_t, \pi(s_t)) + V_{\pi}(s_{t+1})$$

**A policy  $\pi'(s_t)$  is better than policy  $\pi(s_t)$  if  $V_{\pi'}(s_t) \geq V_{\pi}(s_t)$  for every  $s_t$ .**

	$V_{\pi_A}(s_0)$	$V_{\pi_B}(s_0)$	$V_{\pi_C}(s_0)$
C1	-20	-20	-20
C2	-50	100	100
C3	100	100	100
C4	0	100	100
C5	0	100	100
C6	100	100	100
C7	0	100	100
C8	0	100	100
C9	100	100	100

$V_{\pi}(s_t)$  for Max\_steps=5

**Conclusion:** We can't decide which policy is the best.

**Discount rate  $0 < \gamma < 1$**

$$V_{\pi}(s_t) = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}(s_{t+k}, \pi(s_{t+k}))$$

or recursively:

$$V_{\pi}(s_t) = r_{t+1}(s_t, \pi(s_t)) + \gamma V_{\pi}(s_{t+1})$$

**Conclusion:**  $V_{\pi_C}(s_t)$  is the best policy

$\gamma = 0.9$	$V_{\pi_B}(s_0)$	$V_{\pi_C}(s_0)$
C1	-20	-20
C2	81	81
C3	72.9	90
C4	81	81
C5	90	90
C6	100	100
C7	90	90
C8	100	100
C9	100	100

## Action value $Q_{\pi}(s_t, \alpha_t)$

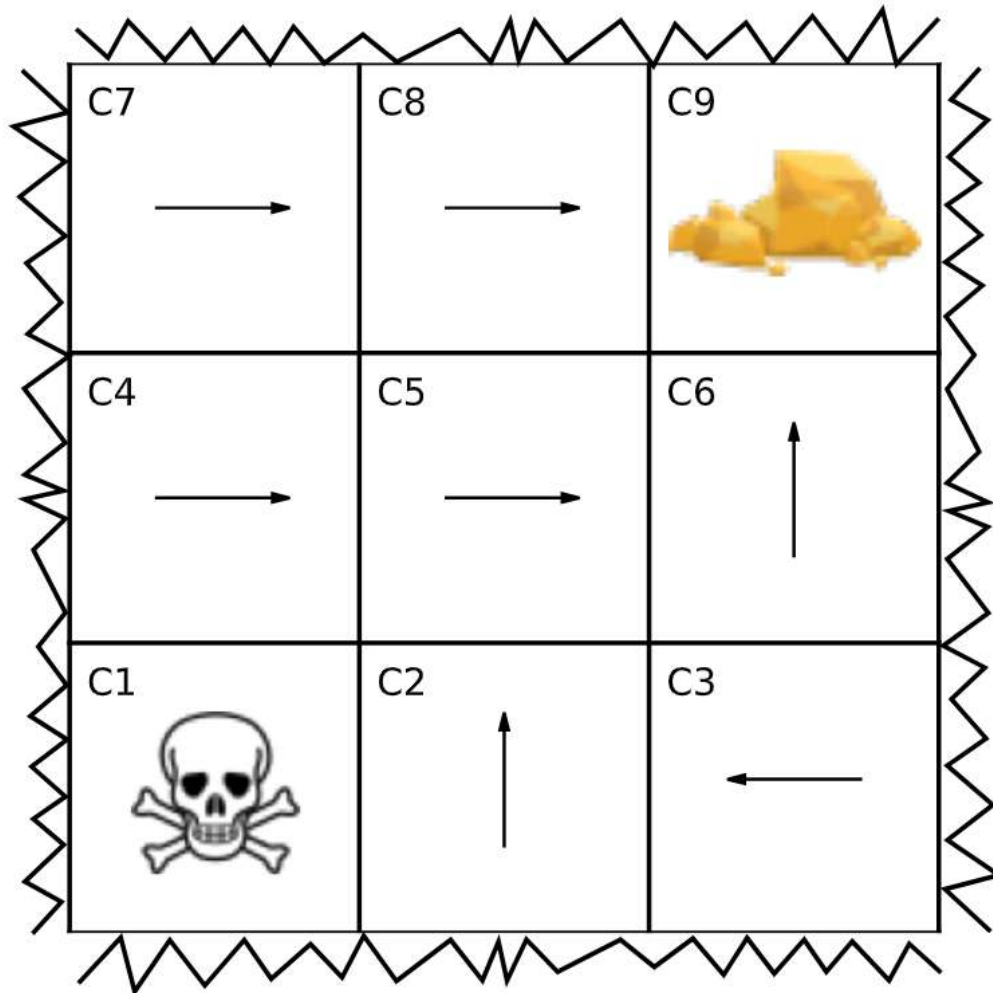
$$Q_{\pi}(s_t, \alpha_t) = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}(s_{t+k}, \alpha_{t+k})$$

or recursively:

$$Q_{\pi}(s_t, \alpha_t) = r_{t+1}(s_t, \alpha_t) + \gamma Q_{\pi}(s_{t+1}, \alpha_{t+1})$$

Under a policy  $\alpha_{t+1} = \pi(s_{t+1})$ :

$$Q_{\pi}(s_t, \alpha_t) = r_{t+1}(s_t, \alpha_t) + \gamma Q_{\pi}(s_{t+1}, \pi(s_{t+1})) = r_{t+1}(s_t, \alpha_t) + \gamma V_{\pi}(s_{t+1})$$



## $Q_{\pi}(s_t, \alpha_t)$ for state C5

$$Q_{\pi_B}(C5, \rightarrow) = r_{t+1}(C5, \rightarrow) + 0.9V_{\pi_B}(C6) = 0 + 0.9(100) = 90$$

$$Q_{\pi_B}(C5, \uparrow) = r_{t+1}(C5, \uparrow) + 0.9V_{\pi_B}(C8) = 0 + 0.9(100) = 90$$

$$Q_{\pi_B}(C5, \leftarrow) = r_{t+1}(C5, \leftarrow) + 0.9V_{\pi_B}(C4) = 0 + 0.9(81) = 72$$

$$Q_{\pi_B}(C5, \downarrow) = r_{t+1}(C5, \downarrow) + 0.9V_{\pi_B}(C2) = 0 + 0.9(81) = 72$$



$G$ : total reward from initial state  $s_0$

$V_\pi(s_t)$ : accumulated reward from intermediate state  $s_t$

$Q_\pi(s_t, \alpha_t)$ : accumulated reward from intermediate state  $s_t$  performing action  $\alpha_t$  and following policy  $\pi(s_{t+1})$  afterwards

Can I find an **optimal policy**  $\pi^*$  for maximizing  $G, V_\pi(s_t), Q_\pi(s_t, \alpha_t)$ ?

## Maximization of $V_{\pi}(s_t)$ with a policy $\alpha_t = \pi(s_t)$

Value of the state:

$$V_{\pi}(s_t) = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}(s_{t+k}, \pi(s_{t+k}))$$

Using the reward function:

$$V_{\pi}(s_t) = \sum_{k=0}^{T-t-1} \gamma^k R(s_{t+k+1})$$

## Maximization of $V_\pi(s_t)$ with a policy $\alpha_t = \pi(s_t)$

Using the transition function recursively:

$$V_\pi(s_t) = \sum_{k=0}^{T-t-1} \gamma^k R(f(s_{t+k}, \pi(s_{t+k})))$$

$$V_\pi(s_t) = R(f(s_t, \pi(s_t))) + \gamma R(f(s_{t+1}, \pi(s_{t+1}))) + \dots + \gamma^{T-t-1} R(f(s_{T-1}, \pi(s_{T-1})))$$

$$V_\pi(s_t) = R(f(s_t, \pi(s_t))) + \gamma R(f(f(s_t, \pi(s_t))), \pi(f(s_t, \pi(s_t)))) + \dots = H(s_t, \pi)$$

Main goal in RL:  $\arg \max_{\pi} H(s_t, \pi)$  for every state  $s_t$

## Part B: Methods of RL

**Optimization of the accumulated reward starting from every state (Model based methods)**



main goal:  $\operatorname{argmax}_{\pi} H(s_t, \pi)$  for every  $s_t$

Generalized Policy iteration (GPI)

# Generalized Policy iteration (GPI)

We start from a random policy that maps every state,  $s_t$ , to an action,  $\alpha_t$

We calculate the  $Q(s_t, \alpha_t)$  values according to the policy:



C13 →	C14 →	C15 →	C16 
C9 →	C10 →	C11 →	C12 ↑
C5 ↑	C6 ←	C7 ↑	C8 ↑
C1 	C2 →	C3 ↑	C4 ↑

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C1	-10.00	-10.00	65.61	65.61
C2	-20.00	-10.00	72.90	59.05
C3	65.61	-10.00	81.00	81.00
C4	72.90	-10.00	-10.00	90.00
C5	-10.00	-20.00	59.05	72.90
C6	65.61	65.61	81.00	81.00
C7	59.05	72.90	90.00	90.00
C8	81.00	81.00	-10.00	100.00
C9	-10.00	65.61	81.00	81.00
C10	72.90	59.05	90.00	90.00
C11	81.00	81.00	100.00	100.00
C12	90.00	90.00	-10.00	100.00
C13	-10.00	72.90	90.00	-10.00
C14	81.00	81.00	100.00	-10.00
C15	90.00	90.00	100.00	-10.00
C16	100.00	100.00	-10.00	-10.00

# Generalized Policy iteration (GPI)

For every state we apply the policy,  $\pi$ , repeatedly and calculate  $Q(s_t, \alpha_t)$  for state, action  $(s_t, \alpha_t)$  pair:

$$Q(s_t, \alpha_t) = r_{t+1}(s_t, \alpha_t) + \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}(s_{t+k+1}, \pi(s_{t+k+1}))$$



C13 →	C14 →	C15 →	C16 
C9 →	C10 →	C11 →	C12 ↑
C5 ↑	C6 ←	C7 ↑	C8 ↑
C1 	C2 →	C3 ↑	C4 ↑

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C1	-10.00	-10.00	65.61	65.61
C2	-20.00	-10.00	72.90	59.05
C3	65.61	-10.00	81.00	81.00
C4	72.90	-10.00	-10.00	90.00
C5	-10.00	-20.00	59.05	72.90
C6	65.61	65.61	81.00	81.00
C7	59.05	72.90	90.00	90.00
C8	81.00	81.00	-10.00	100.00
C9	-10.00	65.61	81.00	81.00
C10	72.90	59.05	90.00	90.00
C11	81.00	81.00	100.00	100.00
C12	90.00	90.00	-10.00	100.00
C13	-10.00	72.90	90.00	-10.00
C14	81.00	81.00	100.00	-10.00
C15	90.00	90.00	100.00	-10.00
C16	100.00	100.00	-10.00	-10.00

# Generalized Policy iteration (GPI)

We go through all states and for every action  $\alpha'_t$  we compare between the  $Q(s_t, \alpha'_t)$  and  $Q(s_t, \pi(s_t))$ .

If  $Q(s_t, \alpha'_t) > Q(s_t, \pi(s_t))$ : we set  $\alpha'_t = Q(s_t, \pi(s_t))$  else nothing changes.



C13 →	C14 →	C15 →	C16 
C9 →	C10 →	C11 →	C12 ↑
C5 ↑	C6 ←	C7 ↑	C8 ↑
C1 	C2 →	C3 ↑	C4 ↑

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C1	-10.00	-10.00	65.61	65.61
C2	-20.00	-10.00	72.90	72.90
C3	65.61	-10.00	81.00	81.00
C4	72.90	-10.00	-10.00	90.00
C5	-10.00	-20.00	72.90	72.90
C6	65.61	65.61	81.00	81.00
C7	72.90	72.90	90.00	90.00
C8	81.00	81.00	-10.00	100.00
C9	-10.00	65.61	81.00	81.00
C10	72.90	72.90	90.00	90.00
C11	81.00	81.00	100.00	100.00
C12	90.00	90.00	-10.00	100.00
C13	-10.00	72.90	90.00	-10.00
C14	81.00	81.00	100.00	-10.00
C15	90.00	90.00	100.00	-10.00
C16	100.00	100.00	-10.00	-10.00


























# Generalized Policy iteration (GPI)

We repeat until  $Q(s_t, a_t)$  stop changing, then the optimal policy has been found

C13 →	C14 →	C15 →	C16 
C9 →	C10 →	C11 →	C12 ↑
C5 →	C6 →	C7 →	C8 ↑
C1 	C2 →	C3 →	C4 ↑

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C1	-10.00	-10.00	65.61	65.61
C2	-20.00	-10.00	72.90	72.90
C3	65.61	-10.00	81.00	81.00
C4	72.90	-10.00	-10.00	90.00
C5	-10.00	-20.00	72.90	72.90
C6	65.61	65.61	81.00	81.00
C7	72.90	72.90	90.00	90.00
C8	81.00	81.00	-10.00	100.00
C9	-10.00	65.61	81.00	81.00
C10	72.90	72.90	90.00	90.00
C11	81.00	81.00	100.00	100.00
C12	90.00	90.00	-10.00	100.00
C13	-10.00	72.90	90.00	-10.00
C14	81.00	81.00	100.00	-10.00
C15	90.00	90.00	100.00	-10.00
C16	100.00	100.00	-10.00	-10.00




# Generalized Policy iteration (GPI)

C57	C58 	C59 	C60 	C61 	C62 	C63 	C64 
C49	C50	C51	C52	C53	C54	C55	C56
C41	C42 	C43	C44 	C45 	C46	C47 	C48
C33	C34	C35 	C36	C37	C38 	C39 	C40
C25	C26	C27	C28 	C29 	C30	C31 	C32
C17 	C18	C19	C20	C21	C22	C23	C24 
C9	C10	C11	C12	C13	C14	C15 	C16
C1 	C2	C3	C4	C5 	C6	C7 	C8

What happens in a large  $N \times N$  grid?  
I cannot iterate over all states and change incrementally the policy,  
I need to update the policy faster!

# Optimization of the accumulated reward starting from a specific state (Model free methods)

main goal:  $\operatorname{argmax}_{\pi} H(s_t, \pi)$  for specific  $s_t = s_0$

C13	C14	C15	C16 
C9	C10	C11	C12
C5	C6	C7	C8
C1 	C2  START	C3	C4

# Optimization of the accumulated reward starting from a specific state (Model free methods)

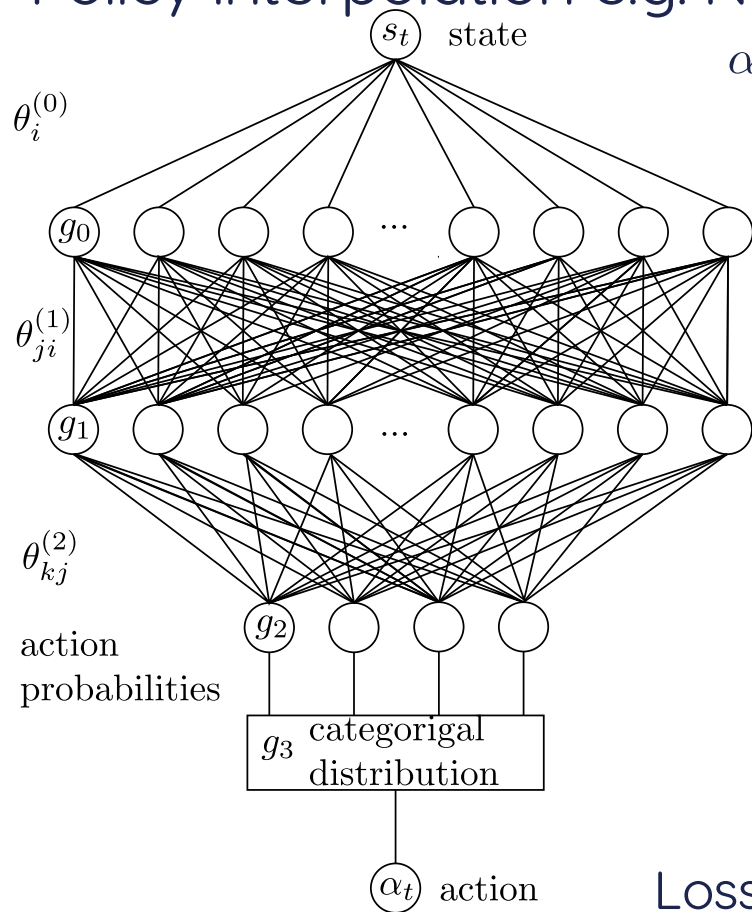
main goal:  $\operatorname{argmax}_{\pi} H(s_t, \pi)$  for specific  $s_t = s_0$

- Policy Gradients (Actor method)
- Q-Learning (Critic Method)
  - ✓ Monte Carlo method
  - ✓ Estimates of the accumulated reward
  - ✓ Exploitation of the state information (SARSA, Q-Learning)
- Deep Q-learning
- Actor-Critic method

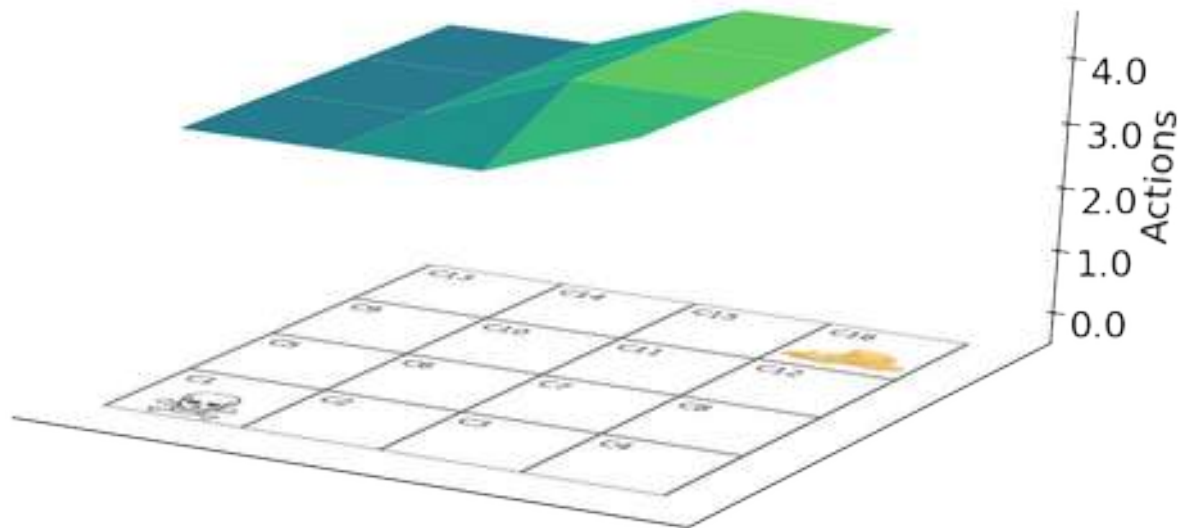
# Policy Gradients

Policy Interpolation e.g. Neural Network (Actor):

$$\alpha_t = \pi_{\theta}(s_t) = g_3 \left( g_2 \left( \sum_{j=1}^N \theta_{kj}^{(2)} g_1 \left( \sum_{i=1}^N \theta_{ji}^{(1)} g_0 \left( \theta_i^{(0)} s_t \right) \right) \right) \right)$$



Interpolation result:

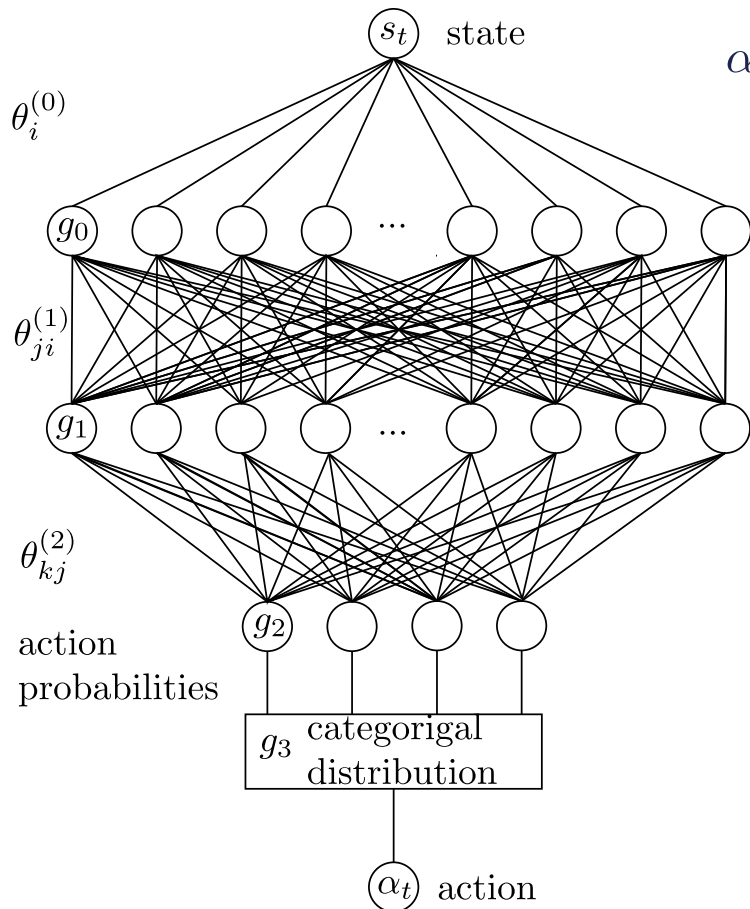


Loss function:  $-H(s_t, \pi_{\theta}(s_t))$

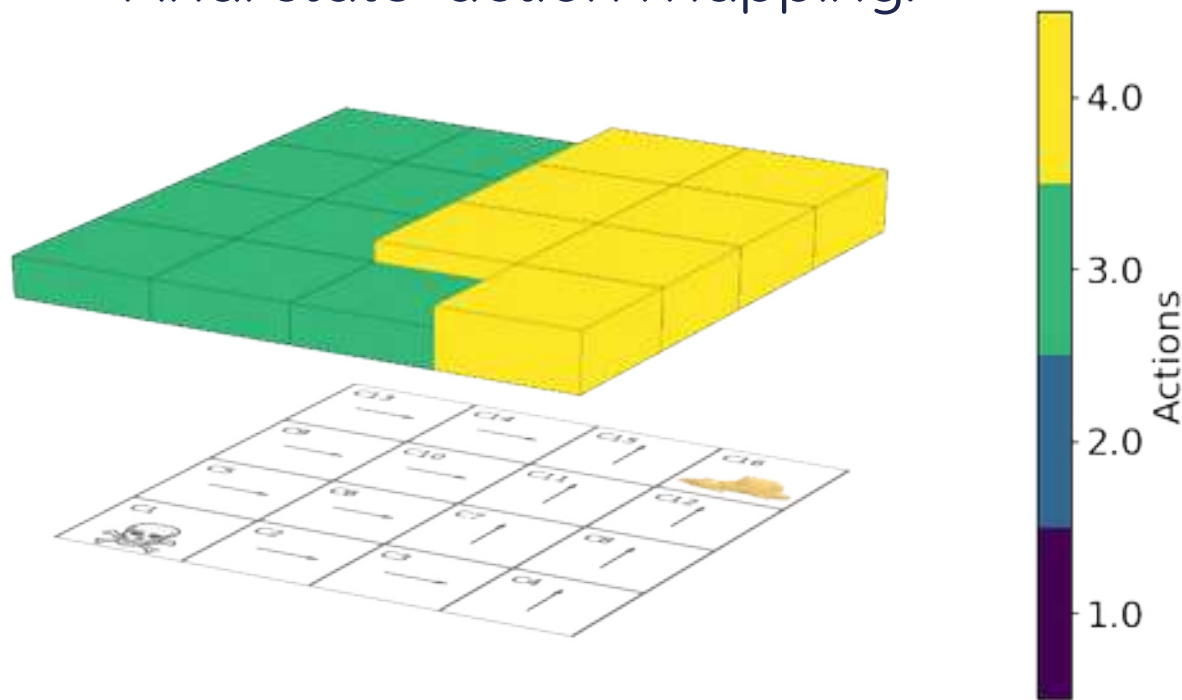
# Policy Gradients

Policy Interpolation e.g. Neural Network (Actor):

$$\alpha_t = \pi_{\theta}(s_t) = g_3 \left( g_2 \left( \sum_{j=1}^N \theta_{kj}^{(2)} g_1 \left( \sum_{i=1}^N \theta_{ji}^{(1)} g_0 \left( \theta_i^{(0)} s_t \right) \right) \right) \right)$$

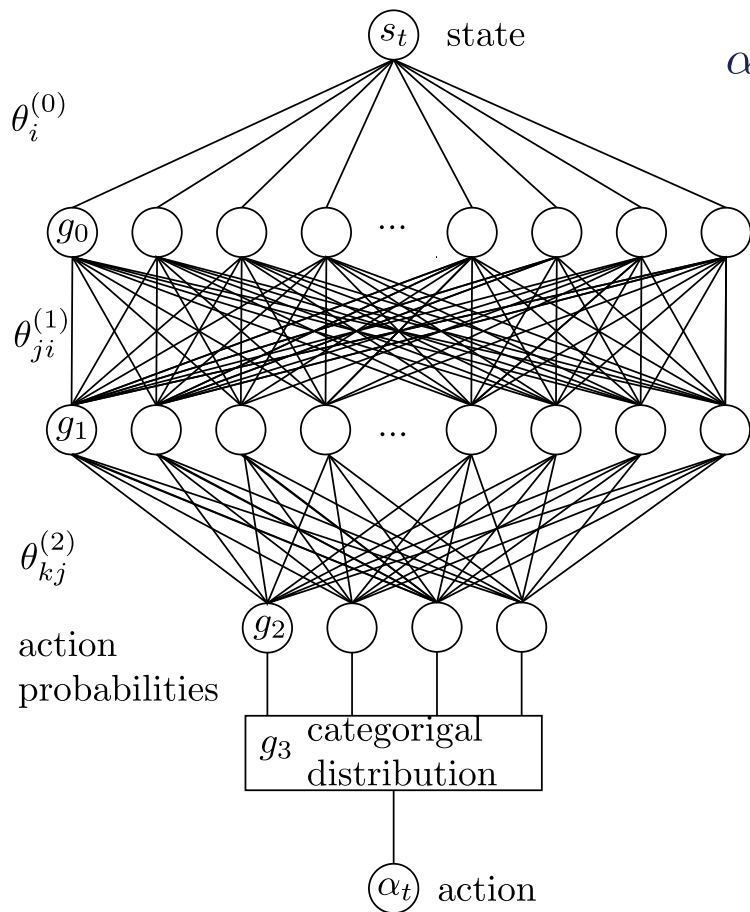


Final state-action mapping:



# Policy Gradients

## Policy Interpolation e.g. Neural Network



$$\alpha_t = \pi_{\theta}(s_t) = g_3 \left( g_2 \left( \sum_{j=1}^N \theta_{kj}^{(2)} g_1 \left( \sum_{i=1}^N \theta_{ji}^{(1)} g_0 \left( \theta_i^{(0)} s_t \right) \right) \right) \right)$$

Inconvenience:

The algorithm needs to estimate the accumulated reward  $H(s_t, \pi)$  in almost all the states and actions pairs before it finally settles in a policy.

That is 4 actions in 16 states:  $4^{16}$  combinations!!

What would happen in a bigger grid (e.g. 8x8)?

If only we could speed up this process...



## Critic methods

We will use the quality of the state  $Q(s_t, \alpha_t)$ , (from part A) to assign a value to each possible action at each state.

The agent moves inside the grid for:

a set number of steps (nsteps,  $k$ )

a set number of episodes (nepisodes,  $n$ )

and accumulates the reward after the completion of the episode.

# Critic methods

We will use the  $Q(s_t, \alpha_t)$  values as estimates of the reward in order to speed training.

Instead of a complicated interpolator for the policy (e.g. a neural network) we will use a random policy and we will accumulate the reward for each state and action taken i.e.  $Q(s_t, \alpha_t)$ .

After a set number of episodes (nepisodes) we will start using the estimates of the table for the reward from the previous runs of the agent.




We will look at the different ways to evaluate the estimates of the reward (SARSA, Q-learning, interpolation)

# Critic Methods

The agent monitors in a table the accumulated reward it receives for moving inside the mine, at the end of the episode.

Initially: Set  $Q(s_t, \alpha_t) = 0$  everywhere and random policy in every episode:

Episode 1 Step 0






				$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C13	C14	C15	C16	C1	0.00	0.00	0.00	0.00
				C2	0.00	0.00	0.00	0.00
				C3	0.00	0.00	0.00	0.00
				C4	0.00	0.00	0.00	0.00
C9	C10	C11	C12	C5	0.00	0.00	0.00	0.00
				C6	0.00	0.00	0.00	0.00
				C7	0.00	0.00	0.00	0.00
				C8	0.00	0.00	0.00	0.00
C5	C6	C7	C8	C9	0.00	0.00	0.00	0.00
				C10	0.00	0.00	0.00	0.00
				C11	0.00	0.00	0.00	0.00
				C12	0.00	0.00	0.00	0.00
C1	C2	C3	C4	C13	0.00	0.00	0.00	0.00
	 START			C14	0.00	0.00	0.00	0.00
				C15	0.00	0.00	0.00	0.00
				C16	0.00	0.00	0.00	0.00

## Critic Method - $Q(s_t, \alpha_t)$ values update (1/4)

Use of the definition for the  $Q(s_t, \alpha_t)$  values, i.e. accumulated reward per episode:

$$Q^{n+1}(s_t, \alpha_t) = (1 - \beta)Q^n(s_t, \alpha_t) + \beta \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \quad \text{Running average}$$

Episode 1

C13	C14	C15	C16 
C9	C10	C11	C12
C5 	C6 	C7	C8
C1 	C2 	C3	C4

$\beta$ : Learning rate hyperparameter

Monte Carlo method:

$$\beta = \frac{1}{C((s_t, \alpha_t))}$$

$C((s_t, \alpha_t))$ : Number of times the specific state action pair  $((s_t, \alpha_t))$  was visited






To understand how the updates are made in the  $Q(s_t, \alpha_t)$  table we will apply the whole magnitude of the update to the  $Q(s_t, \alpha_t)$  table,  $\beta = 1$ .

## Critic Method - $Q(s_t, \alpha_t)$ values update (1/4)

Use of the definition for the  $Q(s_t, \alpha_t)$  values, i.e. accumulated reward per episode:

$$Q^{n+1}(s_t, \alpha_t) = (1 - \beta)Q^n(s_t, \alpha_t) + \beta \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \quad \text{Running average}$$

Episode 1

C13	C14	C15	C16 
C9	C10	C11	C12
C5 	C6 	C7	C8
C1 	C2 	C3	C4










$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	-19.60
C5	0.00	-20.00	0.00	0.00
C6	-19.80	0.00	0.00	0.00

## Critic Method - $Q(s_t, \alpha_t)$ values update (2/4)

Use of the definition for the  $Q(s_t, \alpha_t)$  values, i.e. accumulated reward per episode

$$Q^{n+1}(s_t, \alpha_t) = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$$

Episode 2

C13 	C14 	C15 	C16 
C9 	C10	C11	C12
C5 	C6 	C7	C8
C1 	C2 	C3	C4








$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	94.15
C5	0.00	-20.00	0.00	96.06
C6	95.10	0.00	0.00	0.00
C9	0.00	0.00	0.00	97.03
C13	0.00	0.00	98.01	0.00
C14	0.00	0.00	99.00	0.00
C15	0.00	0.00	100.00	0.00

# Critic Method - $Q(s_t, \alpha_t)$ values update (3/4)

Use of the definition for the Q values, i.e. accumulated reward per episode

$$Q^{n+1}(s_t, \alpha_t) = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$$

Episode 3

C13	C14	C15	C16 
C9	C10	C11 	C12 
C5	C6 	C7 	C8
C1 	C2 	C3	C4








$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	-9.61
C6	95.10	0.00	-9.70	0.00
C7	0.00	0.00	0.00	-9.80
C11	0.00	0.00	-9.90	0.00
C12	0.00	0.00	-10.00	0.00

# Critic Method - $Q(s_t, \alpha_t)$ values update (4/4)

Initially: Random policy in every episode: (Let's run 4 episodes)

$$Q^{n+1}(s_t, \alpha_t) = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$$

Episode 4

C13	C14	C15	C16 
C9	C10	C11 	C12 
C5	C6 	C7 	C8
C1 	C2 	C3	C4

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	96.06
C6	95.10	0.00	97.03	0.00
C7	0.00	0.00	0.00	98.01
C11	0.00	0.00	99.00	0.00
C12	0.00	0.00	-10.00	100.00










# Critic Method - Estimates 1/6 (SARSA, TD learning )

Use of Q estimate for the action value function before the end of the episode:

$$Q^{n+1}(s_t, \alpha_t) = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$$

Episode 5

Accumulated reward:

C13	C14 	C15 	C16 
C9	C10 	C11	C12
C5	C6 	C7	C8
C1 	C2 	C3	C4








$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	96.06
C6	95.10	0.00	97.03	97.03
C10	0.00	0.00	0.00	98.01
C14	0.00	0.00	99.00	0.00
C15	0.00	0.00	100.00	0.00

## Critic Method - Estimates 1/6 (SARSA, TD learning )

Use of Q estimate for the action value function before the end of the episode:

$$Q^{n+1}(s_t, \alpha_t) = Q^n(s_t, \alpha_t) + \beta (r_{t+1}(s_t, \alpha_t) + \gamma Q^n(s_{t+1}, \pi(s_{t+1})) - Q^n(s_t, \alpha_t))$$

Episode 5

C13	C14 	C15 	C16 
C9	C10 	C11	C12
C5	C6 	C7	C8
C1 	C2 	C3	C4








## Critic Method - Estimates 1/6 (SARSA, TD learning )

Use of Q estimate for the action value function before the end of the episode:

$$Q^{n+1}(s_t, \alpha_t) = Q^n(s_t, \alpha_t) + \beta (r_{t+1}(s_t, \alpha_t) + \gamma Q^n(s_{t+1}, \pi(s_{t+1})) - Q^n(s_t, \alpha_t))$$

Episode 5

Advantage








C13	C14 	C15 	C16 
C9	C10 	C11	C12
C5	C6 	C7	C8
C1 	C2 	C3	C4

# Critic Method - Estimates 2/6 (SARSA, TD learning )

Use of Q estimate for the action value function before the end of the episode:

$$Q^{n+1}(s_t, \alpha_t) = r_{t+1}(s_t, \alpha_t) + \gamma Q^n(s_{t+1}, \pi(s_{t+1}))$$

Episode 5

C13	C14 	C15 	C16 
C9	C10 	C11	C12
C5	C6 	C7	C8
C1 	C2 	C3	C4

Accumulated reward:

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	96.06
C6	95.10	0.00	97.03	97.03
C10	0.00	0.00	0.00	98.01
C14	0.00	0.00	99.00	0.00
C15	0.00	0.00	100.00	0.00

Q estimate from table:








$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	0.00
C6	95.10	0.00	97.03	0.00
C10	0.00	0.00	0.00	98.01
C14	0.00	0.00	99.00	0.00
C15	0.00	0.00	100.00	0.00

# Critic Method - Estimates 3/6 (SARSA, TD learning)

Use of Q estimate for the action value function before the end of the episode:

$$Q^{n+1}(s_t, \alpha_t) = r_{t+1}(s_t, \alpha_t) + \gamma Q^n(s_{t+1}, \pi(s_{t+1}))$$

Episode 5

C13 	C14 	C15	C16 
C9	C10 	C11	C12
C5	C6 	C7	C8
C1 	C2 	C3	C4

Accumulated reward:

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	-19.12
C6	95.10	0.00	97.03	-19.31
C10	0.00	0.00	0.00	-19.50
C13	-19.90	0.00	98.01	0.00
C14	-19.70	0.00	99.00	0.00

Q estimate from table:








$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	0.00
C6	95.10	0.00	97.03	0.00
C10	0.00	0.00	0.00	0.00
C13	-10.00	0.00	98.01	0.00
C14	0.00	0.00	99.00	0.00

# Critic Method - Estimates 4/6 (SARSA, TD learning)

Use of Q estimate for the action value function before the end of the episode:

$$Q^{n+1}(s_t, \alpha_t) = r_{t+1}(s_t, \alpha_t) + \gamma Q^n(s_{t+1}, \pi(s_{t+1}))$$

Episode 5

C13 	C14 	C15	C16 
C9	C10 	C11	C12
C5	C6 	C7	C8
C1 	C2 	C3	C4

The updated Q- values differ when we exploit the updated Q-table!

Q estimate from table:








$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	0.00
C6	95.10	0.00	97.03	0.00
C10	0.00	0.00	0.00	0.00
C13	-10.00	0.00	98.01	0.00
C14	0.00	0.00	99.00	0.00

# Critic Method- Estimates 5/6 (Q-learning )

Use of Q estimate for the action value function before the end of the episode:

$$Q^{n+1}(s_t, \alpha_t) = r_{t+1}(s_t, \alpha_t) + \gamma \max_{\alpha_{t+1}} Q^n(s_{t+1}, \alpha_{t+1})$$

Episode 5

C13 	C14 	C15	C16 
C9	C10 	C11	C12
C5	C6 	C7	C8
C1 	C2 	C3	C4

Q estimate from table:

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	0.00
C6	95.10	0.00	97.03	0.00
C10	0.00	0.00	0.00	0.00
C13	-10.00	0.00	98.01	0.00
C14	0.00	0.00	99.00	0.00

max Q estimate from table:

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	96.06
C6	95.10	0.00	97.03	0.00
C10	0.00	0.00	0.00	98.01
C13	87.03	0.00	98.01	0.00
C14	97.03	0.00	99.00	0.00








## Critic Method- Estimates 5/6 (Q-learning )

Use of Q estimate for the action value function before the end of the episode:

$$Q^{n+1}(s_t, \alpha_t) = r_{t+1}(s_t, \alpha_t) + \gamma \max_{\alpha_{t+1}} Q^n(s_{t+1}, \alpha_{t+1})$$

Episode 5

max Q estimate from table:

C13 	C14 	C15	C16 
C9	C10 	C11	C12
C5	C6 	C7	C8
C1 	C2 	C3	C4

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C2	0.00	0.00	0.00	96.06
C6	95.10	0.00	97.03	0.00
C10	0.00	0.00	0.00	98.01
C13	87.03	0.00	98.01	0.00
C14	97.03	0.00	99.00	0.00

The algorithm exploits the fact that in episode #2 in state C13 the miner was able to find the gold!



## Critic Method - Estimates 6/6 (Q-learning)

Q-table after episode #5 using  
accumulated reward

Q-table after episode #5 using the  
maximum estimate

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C1	0.00	0.00	0.00	0.00
C2	0.00	0.00	0.00	0.00
C3	0.00	0.00	0.00	0.00
C4	0.00	0.00	0.00	0.00
C5	0.00	-20.00	0.00	96.06
C6	95.10	0.00	97.03	0.00
C7	0.00	0.00	0.00	98.01
C8	0.00	0.00	0.00	0.00
C9	0.00	0.00	0.00	97.03
C10	0.00	0.00	0.00	0.00
C11	0.00	0.00	99.00	0.00
C12	0.00	0.00	-10.00	100.00
C13	-10.00	0.00	98.01	0.00
C14	0.00	0.00	99.00	0.00
C15	0.00	0.00	100.00	0.00
C16	0.00	0.00	0.00	0.00

$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C1	0.00	0.00	0.00	0.00
C2	0.00	0.00	0.00	96.06
C3	0.00	0.00	0.00	0.00
C4	0.00	0.00	0.00	0.00
C5	0.00	-20.00	0.00	96.06
C6	95.10	0.00	97.03	0.00
C7	0.00	0.00	0.00	98.01
C8	0.00	0.00	0.00	0.00
C9	0.00	0.00	0.00	97.03
C10	0.00	0.00	0.00	98.01
C11	0.00	0.00	99.00	0.00
C12	0.00	0.00	-10.00	100.00
C13	0.00	0.00	98.01	0.00
C14	0.00	0.00	99.00	0.00
C15	0.00	0.00	100.00	0.00
C16	0.00	0.00	0.00	0.00

## Critic Method - Estimates 4/4 (Q-learning)

The estimates allow us to not start learning at an newly visited state from zero, and speed up training.

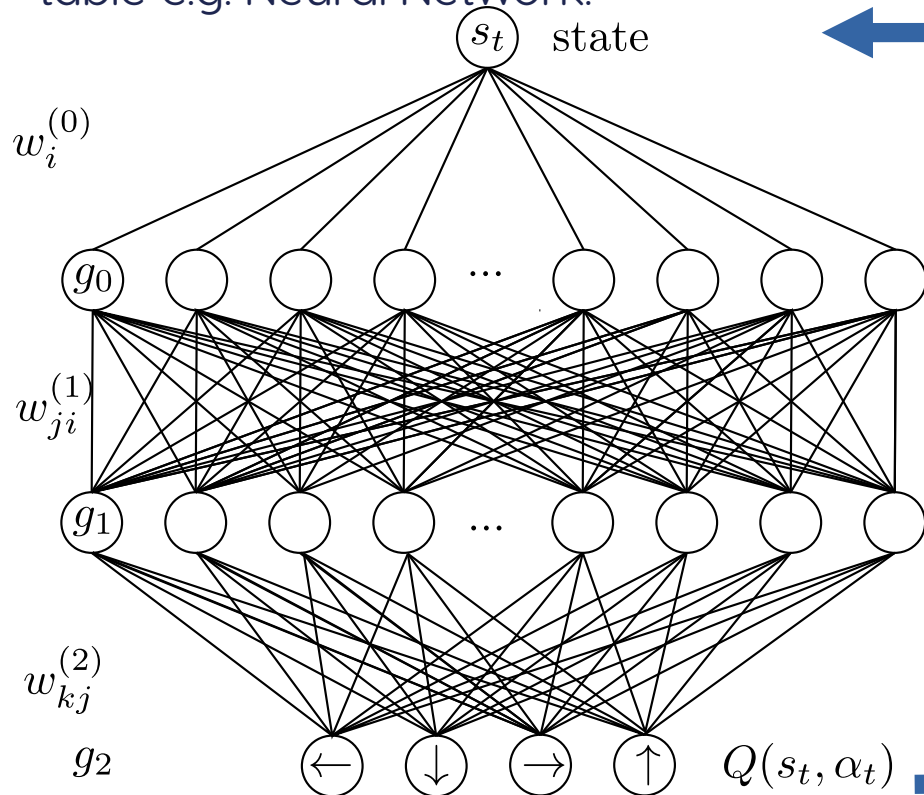
But the values we don't know, i.e "0" in the table, are still too many, we need to explore the state action space more, before we exploit totally the values of the Q table

# Deep Q-Learning (Critic Network)

Interpolation of the Q-values in the Q table e.g. Neural Network.

$$(Q(s_t, \alpha_t))_k = g_2 \left( \sum_{j=1}^N w_{kj}^2 g_1 \left( \sum_{i=1}^N w_{ji}^{(1)} g_0 \left( w_i^{(0)} s_t \right) \right) \right)$$

New state:  $s_{t+1}$



C13	C14	C15	C16
C9	C10	C11	C12
C5	C6	C7	C8
C1	C2	C3	C4

Choose action: 1) Random  
2)  $\max Q(s_t, \alpha_t)$

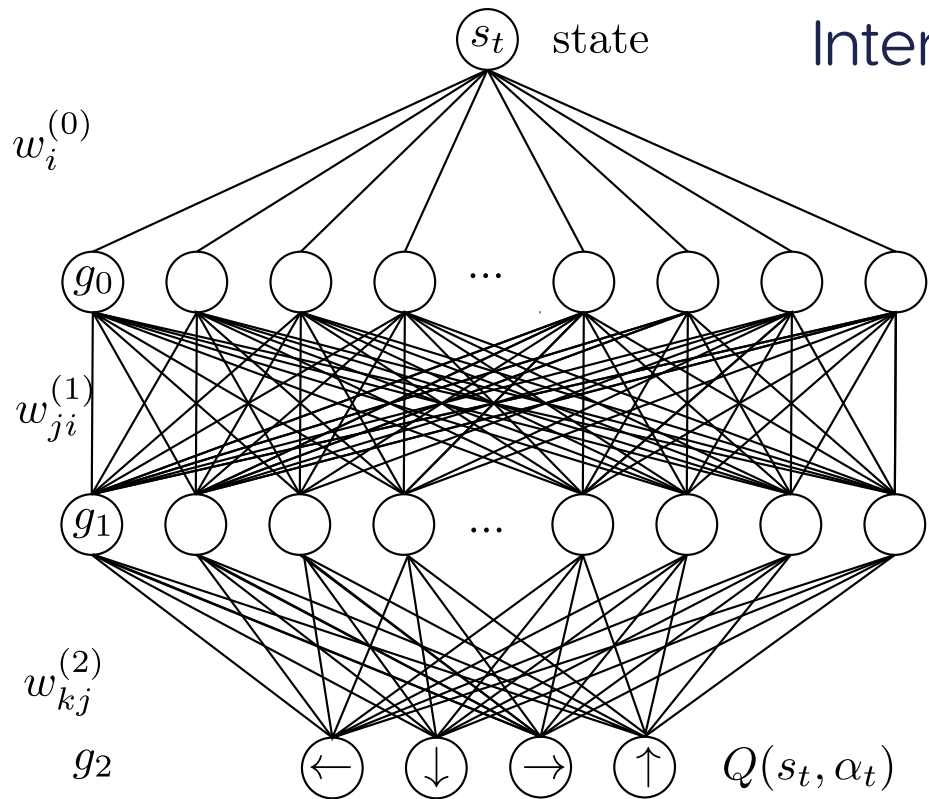
Loss function:  $L = (Q^{(target)}(s_t, \alpha_t) - Q^{(prediction)}(s_t, \alpha_t))$

# Deep Q-Learning (Critic Network)

Interpolation of the Q-values in the Q table e.g. Neural Network.

$$(Q(s_t, \alpha_t))_k = g_2 \left( \sum_{j=1}^N w_{kj}^2 g_1 \left( \sum_{i=1}^N w_{ji}^{(1)} g_0 \left( w_i^{(0)} s_t \right) \right) \right)$$

Interpolation result:

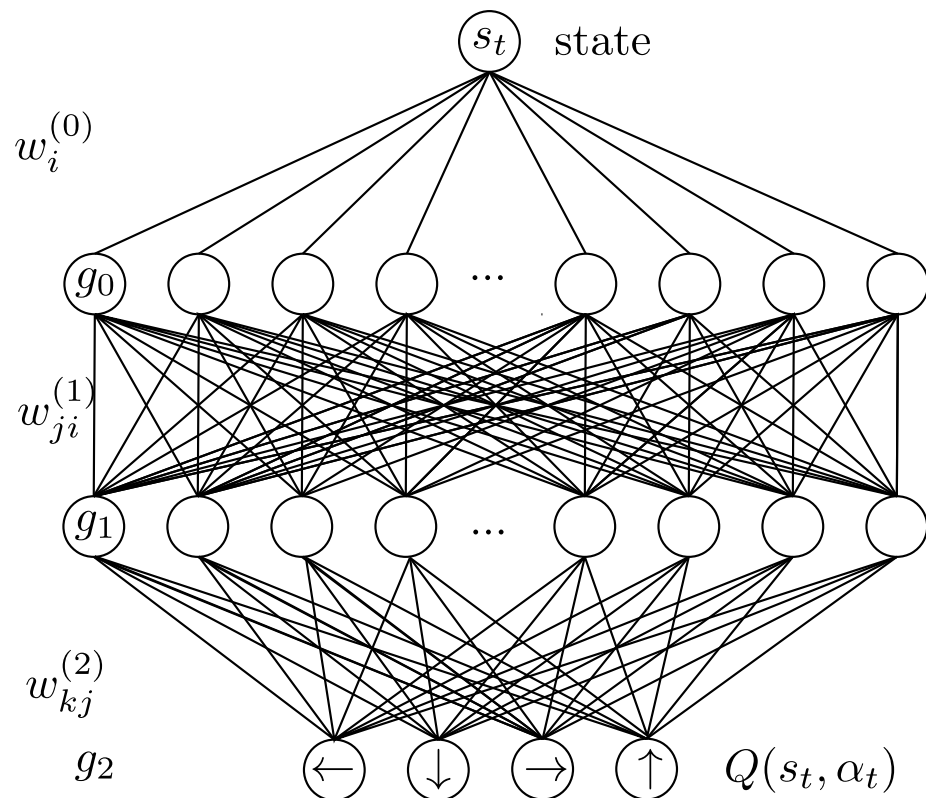


$s^*$	$Q(s_t = s^*, \leftarrow)$	$Q(s_t = s^*, \downarrow)$	$Q(s_t = s^*, \rightarrow)$	$Q(s_t = s^*, \uparrow)$
C1	-9.10	-4.03	2.13	11.85
C2	-15.27	-8.27	2.31	15.79
C3	-11.87	-7.19	1.40	15.55
C4	-11.50	-7.60	0.55	16.15
C5	-9.77	-7.16	-0.15	14.07
C6	-6.32	0.83	5.97	2.28
C7	-7.23	1.57	7.87	-0.24
C8	-8.74	-8.34	-3.07	18.77
C9	-8.19	-2.81	2.87	7.90
C10	-7.55	-1.67	4.62	5.19
C11	-9.19	-4.29	1.87	8.49
C12	-19.98	-13.00	-2.63	39.00
C13	-10.69	-3.73	3.25	9.05
C14	-10.40	-4.15	2.89	10.70
C15	-9.03	-0.28	7.52	-0.08
C16	-10.79	-5.42	1.69	15.70



# Deep Q-Learning (Critic Network)

Interpolation of the Q-values in the Q table e.g. Neural Network.

$$(Q(s_t, \alpha_t))_k = g_2 \left( \sum_{j=1}^N w_{kj}^2 g_1 \left( \sum_{i=1}^N w_{ji}^{(1)} g_0 \left( w_i^{(0)} s_t \right) \right) \right)$$



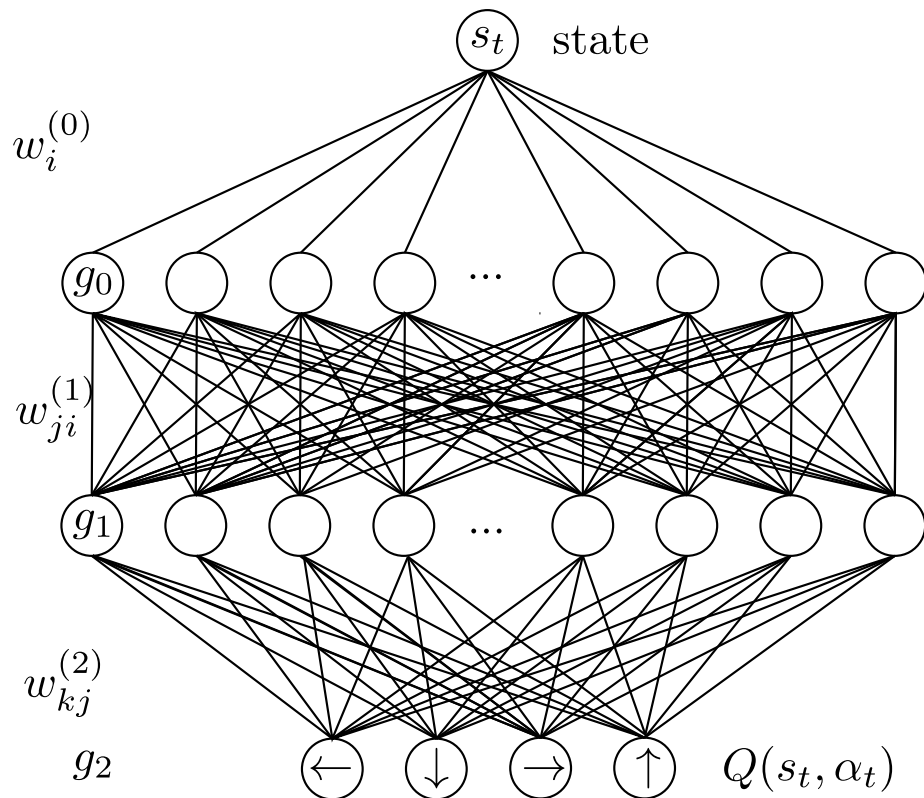
Learned policy:

C13 →	C14 →	C15 →	C16 
C9 ↑	C10 ↑	C11 ↑	C12 ↑
C5 →	C6 →	C7 ↑	C8 ↑
C1 	C2 ↑	C3 ↑	C4 ↑

# Deep Q-Learning (Critic Network)

Interpolation of the Q-values in the Q table e.g. Neural Network.

$$(Q(s_t, \alpha_t))_k = g_2 \left( \sum_{j=1}^N w_{kj}^2 g_1 \left( \sum_{i=1}^N w_{ji}^{(1)} g_0 \left( w_i^{(0)} s_t \right) \right) \right)$$



Inconvenience:

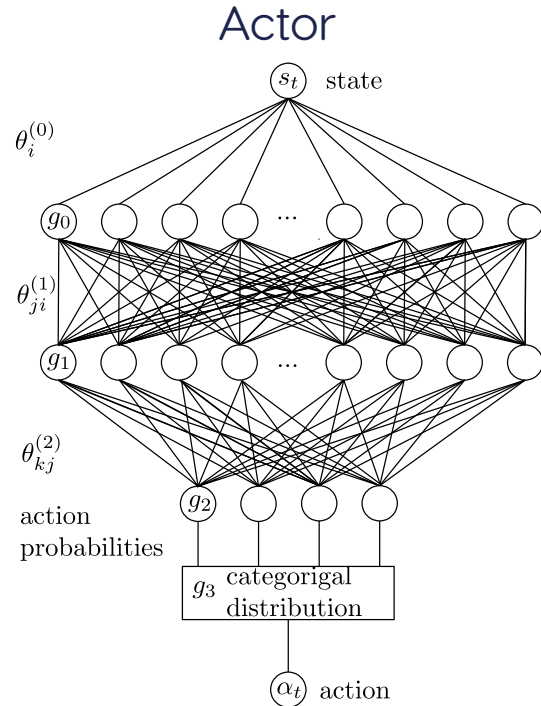
In this algorithm the Q values are interpolated in all parts of the table. These are not exact estimates of the reward based on a completed path.

Because the algorithm sets the interpolation values for Q and then selects actions based on these Q at later stages of training, the algorithm is not guaranteed to converge.

It is also very sensitive to hyperparameter selection.

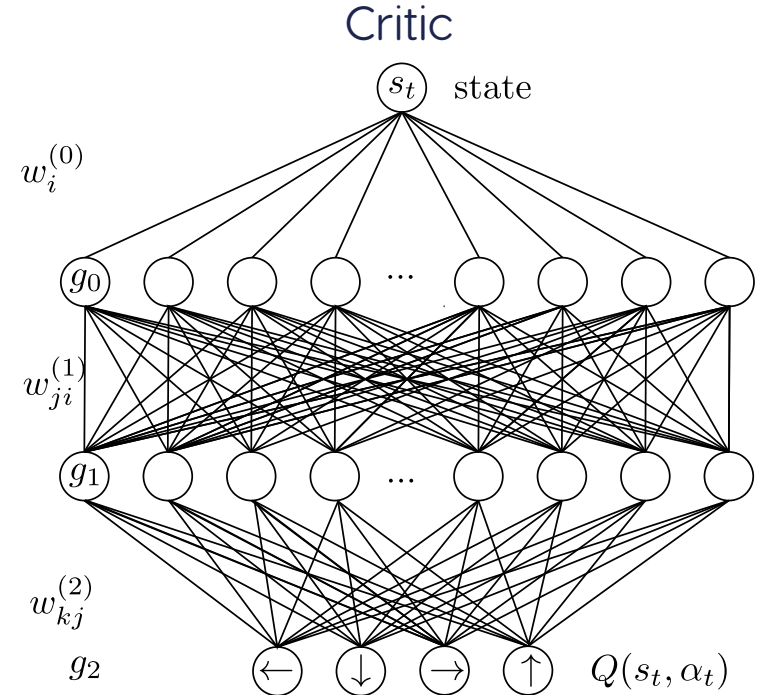
Training is slow, the algorithm may oscillate or diverge.

# Summarizing inconveniences



Outputs a policy

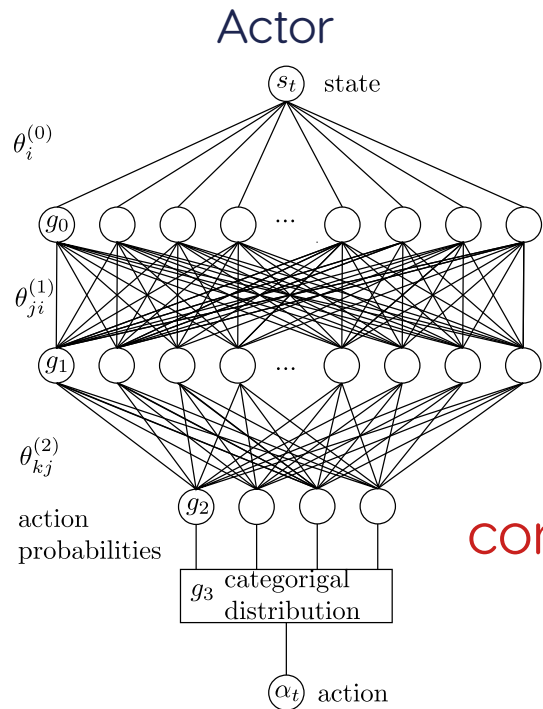
Needs to finish each episode  
batch to calculate the reward  
Needs a lot of training...



Outputs action values

Uses its own estimates to select  
The optimal policy.  
(Chases its own tail)

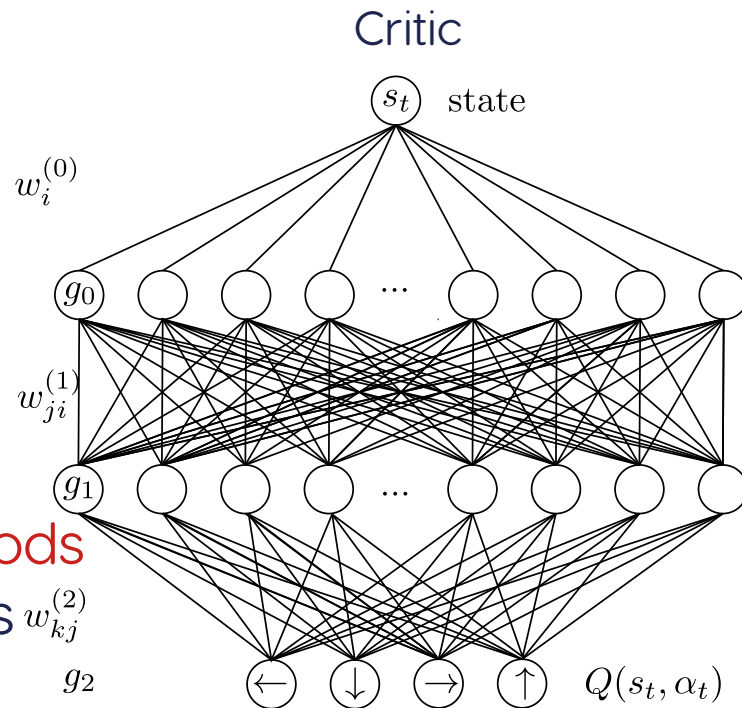
# Summarizing inconveniences



Outputs a policy

Needs to finish each episode  
batch to calculate the reward  
Needs a lot of training...

We will  
combine the two methods  
to cover each others  
deficiencies!

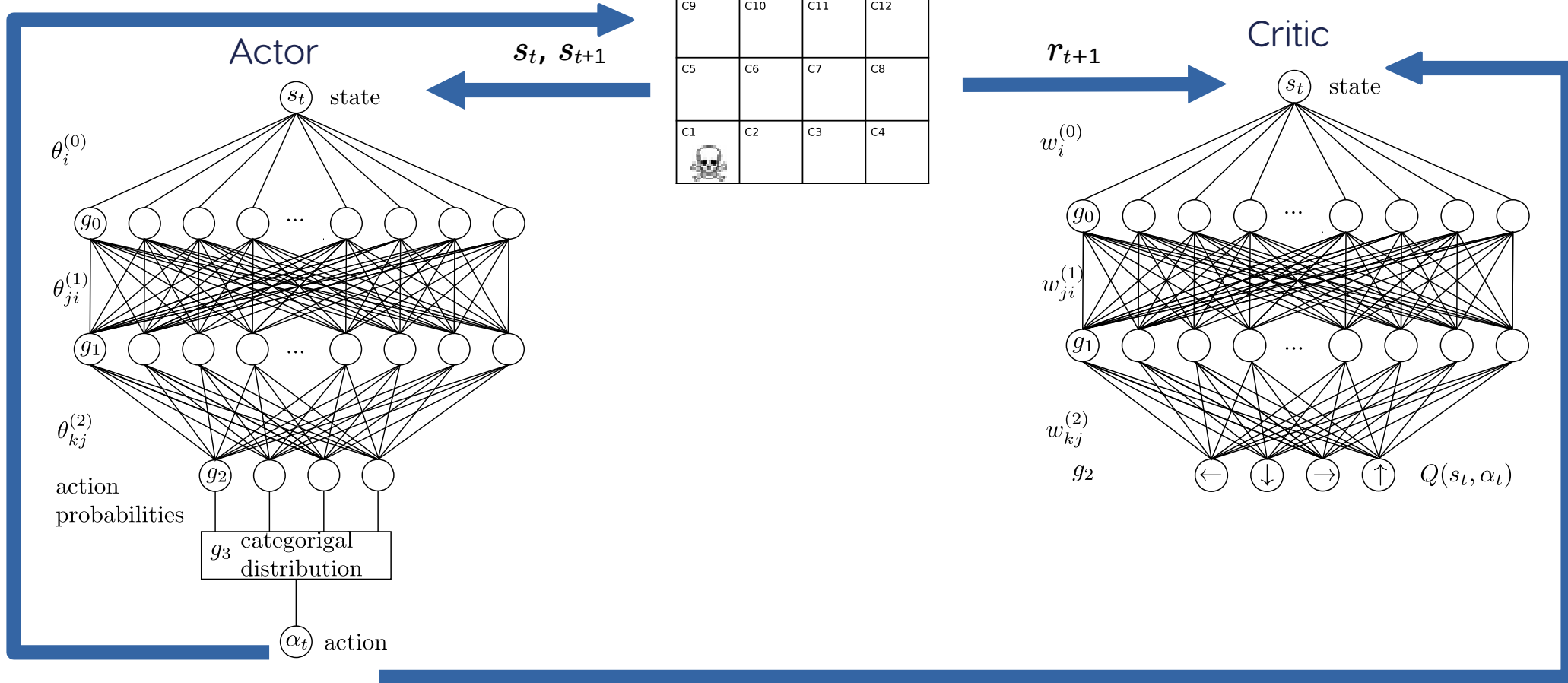


Outputs action values

Uses its own estimates to select  
The optimal policy.  
(Chases its own tail)

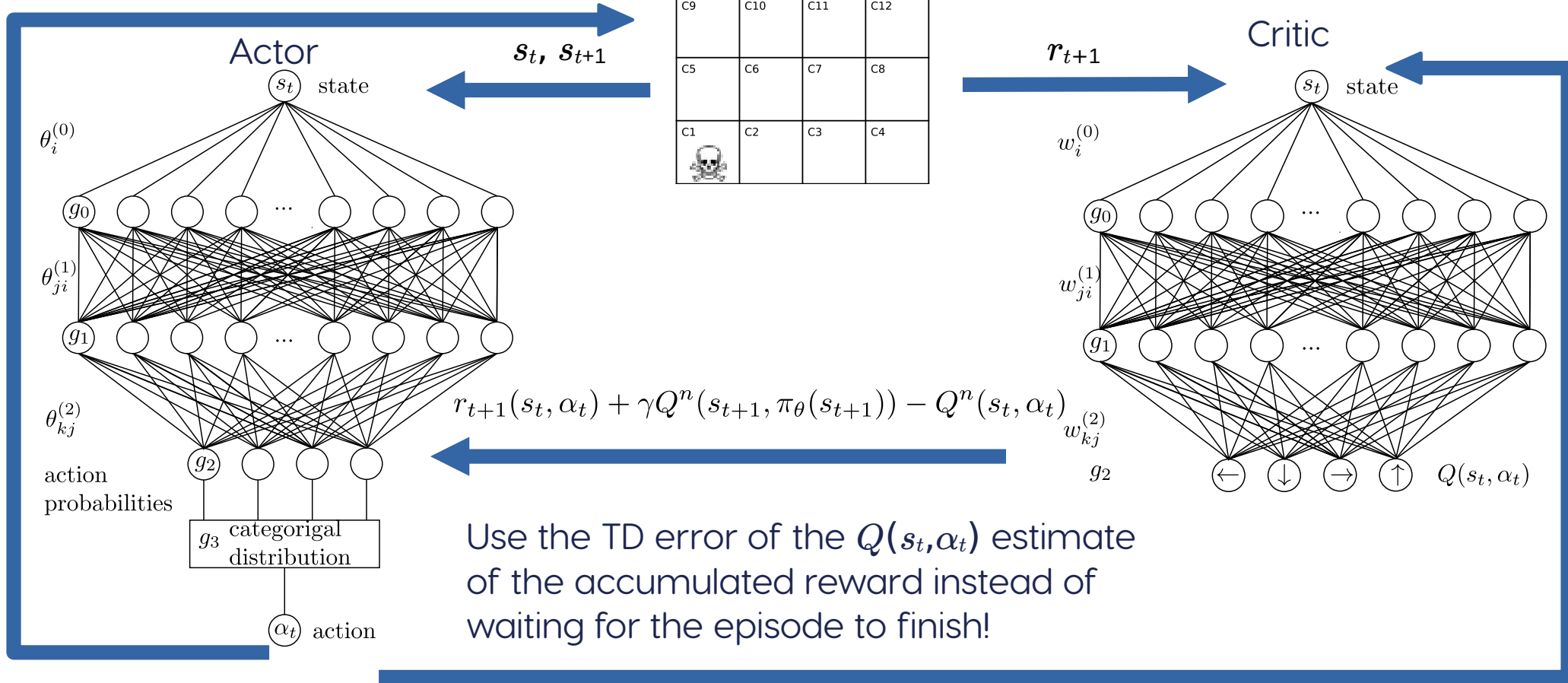


# Actor-Critic method



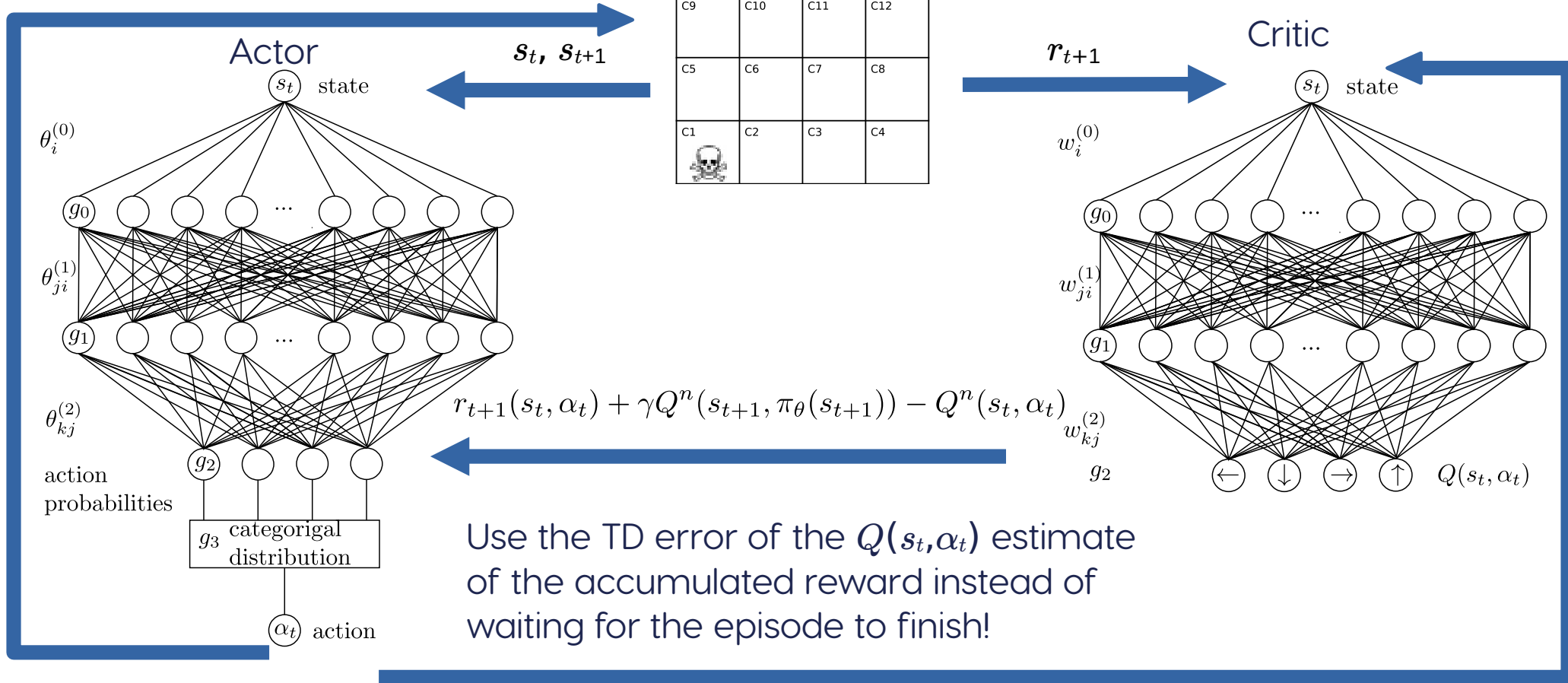
Use the current and next state action pairs  $((s_t, \alpha_t), (s_{t+1}, \alpha_{t+1}))$ , given by the actor policy!

# Actor-Critic method



Use the current and next state action pairs  $((s_t, \alpha_t), (s_{t+1}, \alpha_{t+1}))$ , given by the actor policy!

# Actor-Critic method



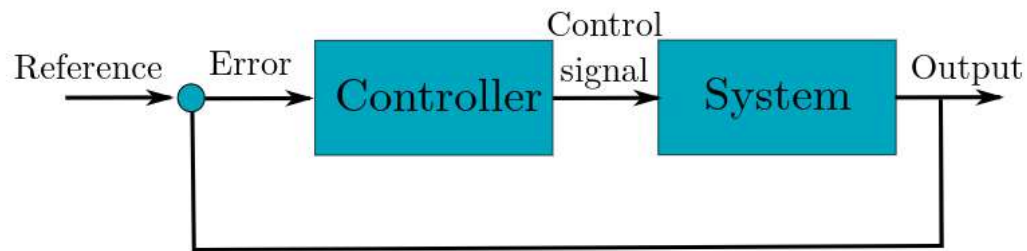
Use the current and next state action pairs  $((s_t, \alpha_t), (s_{t+1}, \alpha_{t+1}))$ , given by the actor policy!

## **Part C: Applications in Geomechanics**



## Filling a glass of water 101

- 1.- **Open** tap.
- 2.- **Observe** the level of water until it reaches the desired amount.
- 3.- **Close** tap.



## Closed-loop control

**Goal:** Take the output to a desired reference

- 1.- Measurement of an output (**Sensor**)
- 2.- Calculation of an error w.r.t. a desired reference and command of a control signal (**Controller**)
- 3.- Perform actions in the system depending on the error (**Actuator**)

**How to design a controller?**

## Simple second-order system

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = u$$

$x_1, x_2$  - states of the system

$u$  - control input

**Goal:** Take the output  $x_1$  to a desired **constant** reference  $r$  ( $\dot{r} = 0$ )

## Error dynamics

Define  $e_1 = x_1 - r$ ,  $e_2 = x_2 - \dot{r} = x_2$ , obtaining:

$$\dot{e}_1 = e_2,$$

$$\dot{e}_2 = u$$

**Advantage:** We can design  $u$  to take  $e_1, e_2$  to zero. This will lead to  $x_1 \rightarrow r$  and  $x_2 \rightarrow \dot{r} = 0$ .



# Proportional-Integral-Derivative Control

$$u = -k_1 e_1 - k_2 \dot{e}_1 - k_3 \int e_1 dt$$

Closed-loop system:

$$u = -k_1 e_1 - k_2 e_2 - k_3 \zeta$$

$$\dot{e}_1 = e_2,$$

$$\dot{e}_2 = -k_1 e_1 - k_2 e_2 - k_3 \zeta,$$

$$\dot{\zeta} = e_1$$

How to choose  $k_1, k_2, k_3$ ?

# Effect of the gains on a PID control

**Proportional gain  $k_1$ :** Time response

**Derivative gain  $k_2$ :** Dampening

**Integral gain  $k_3$ :** Steady-state error and Oscillations

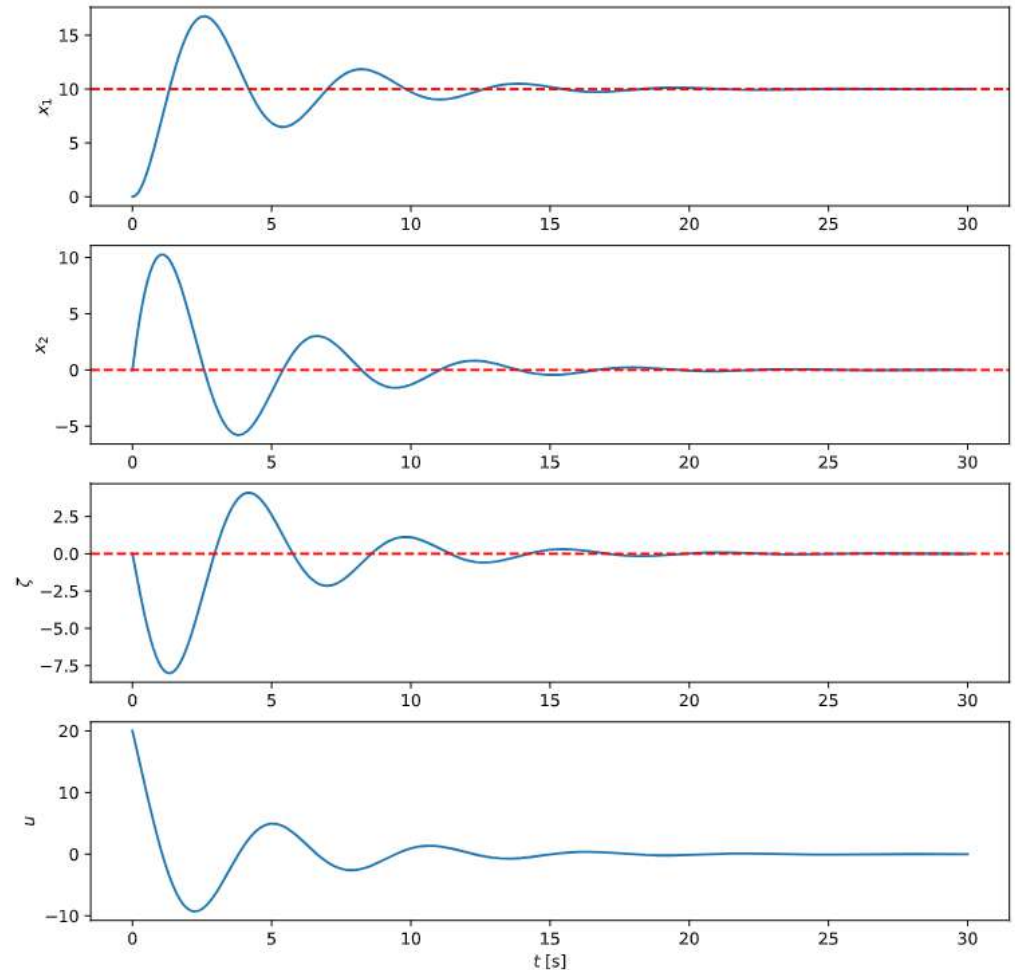
# Gain design

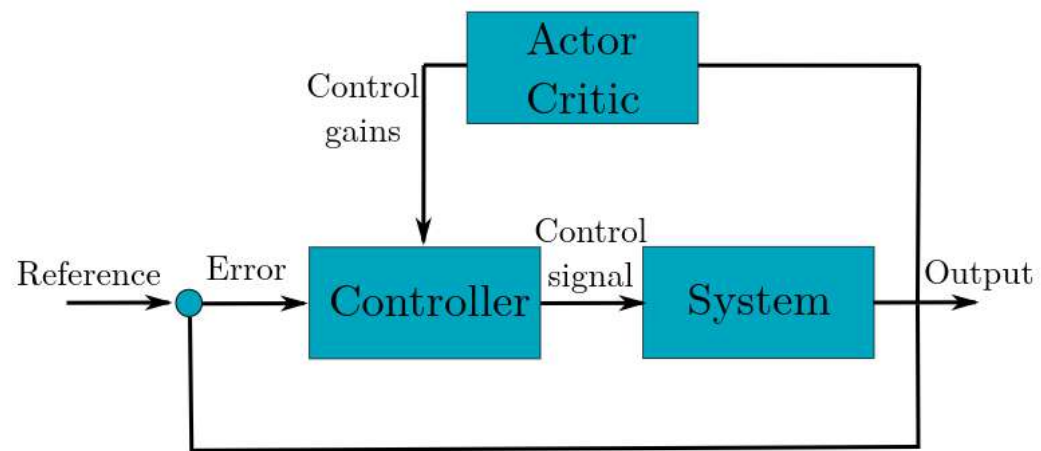
The system is linear and simple, then, it can be proven analytically that the states  $e_1$ ,  $e_2$  will tend to zero exponentially if:

$$k_1 > 0, k_2 > 0, k_3 < k_1 k_2.$$

**How can we tune the gains for nonlinear systems, uncertainties, disturbances, noise, .... ?**

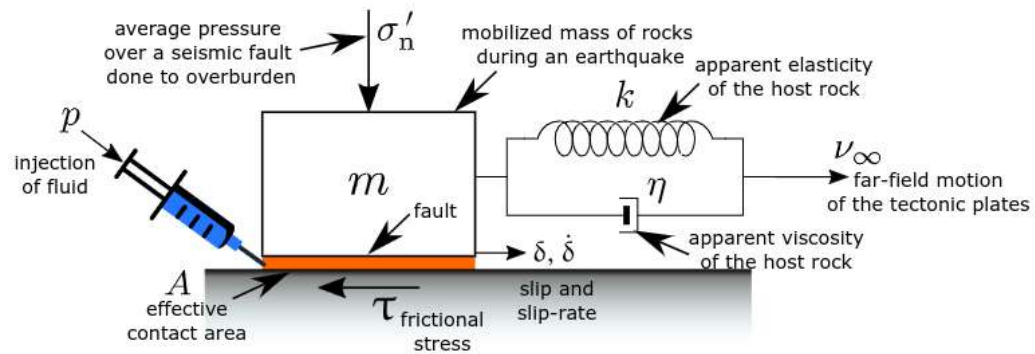
We will use RL for two applications in geomechanics.





## Control + Actor-Critic RL

$$\text{Reward} = e^{-||\text{error}||}$$



## Reduced model for earthquakes

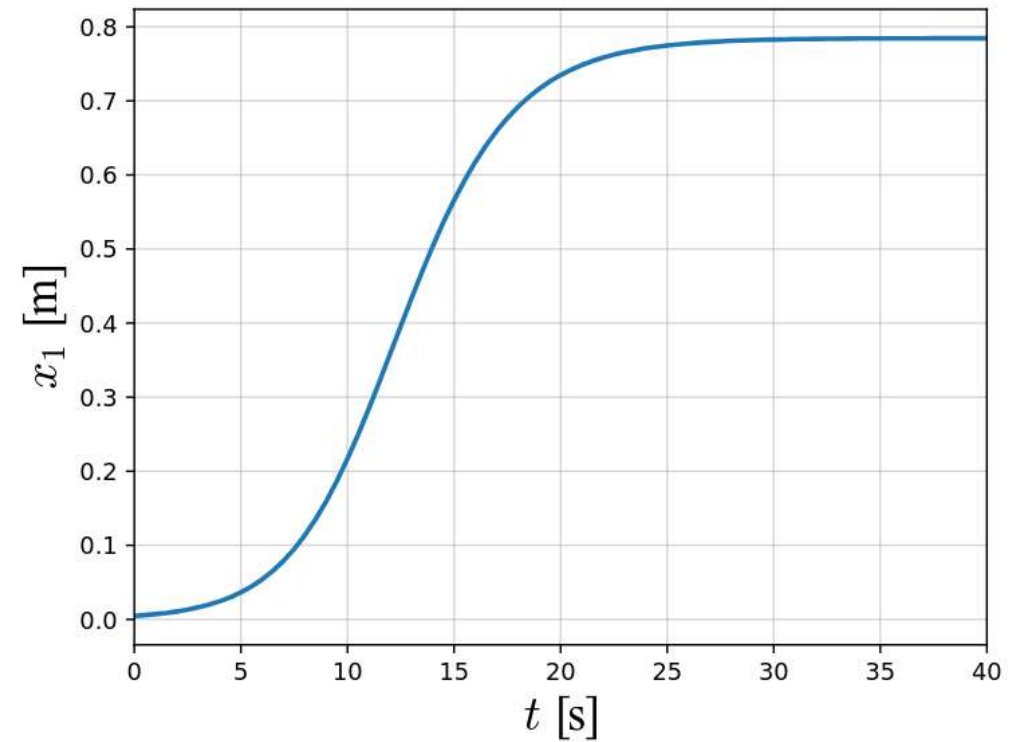
States:  $\delta$  - slip,  $\dot{\delta}$  - slip rate

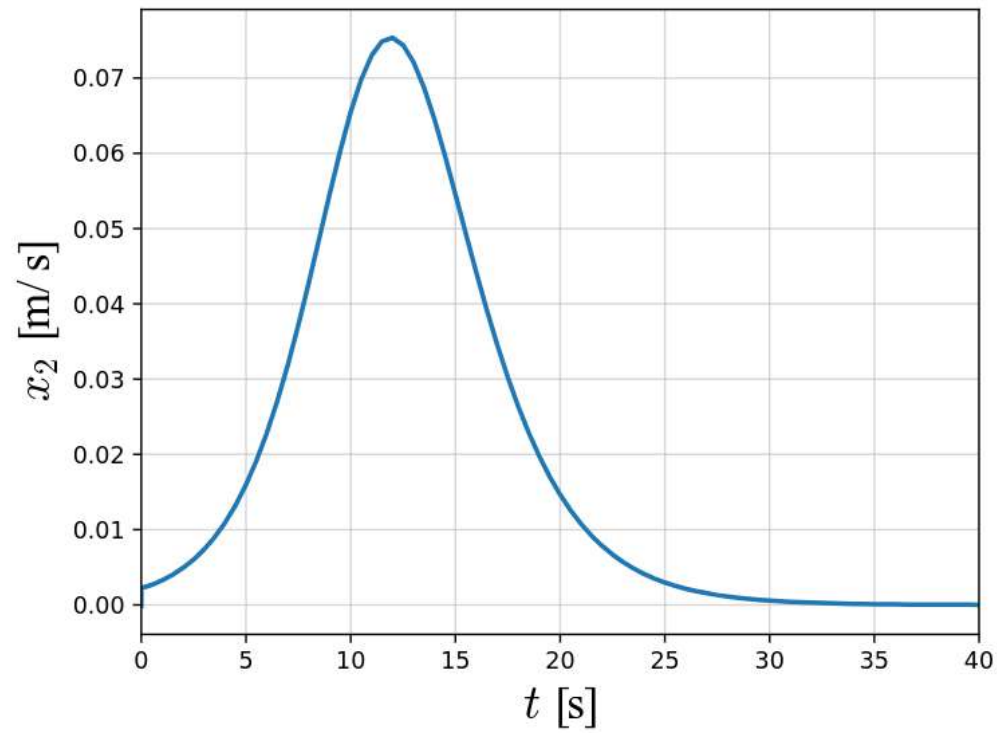
Control:  $p$  - pressure of the injected fluid

Goal: Prevent fast-slip behaviour (earthquakes) by designing  $p$ .

## No control ( $p = 0$ )

The slip goes from zero to a final value in a **short time**...



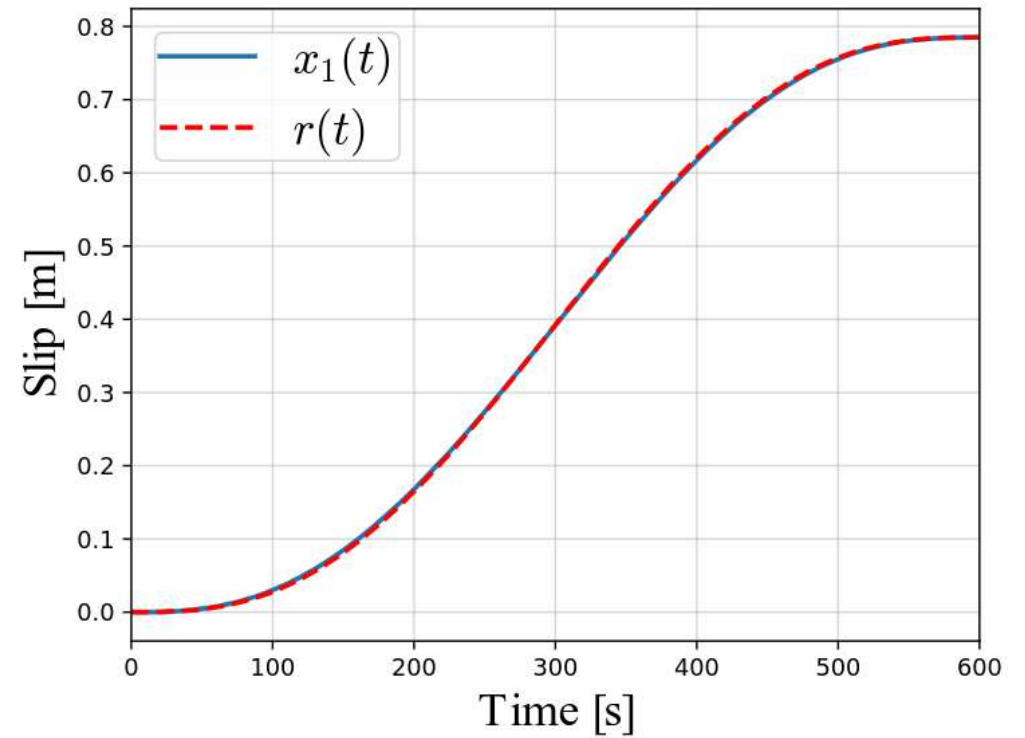


**No control ( $p = 0$ )**

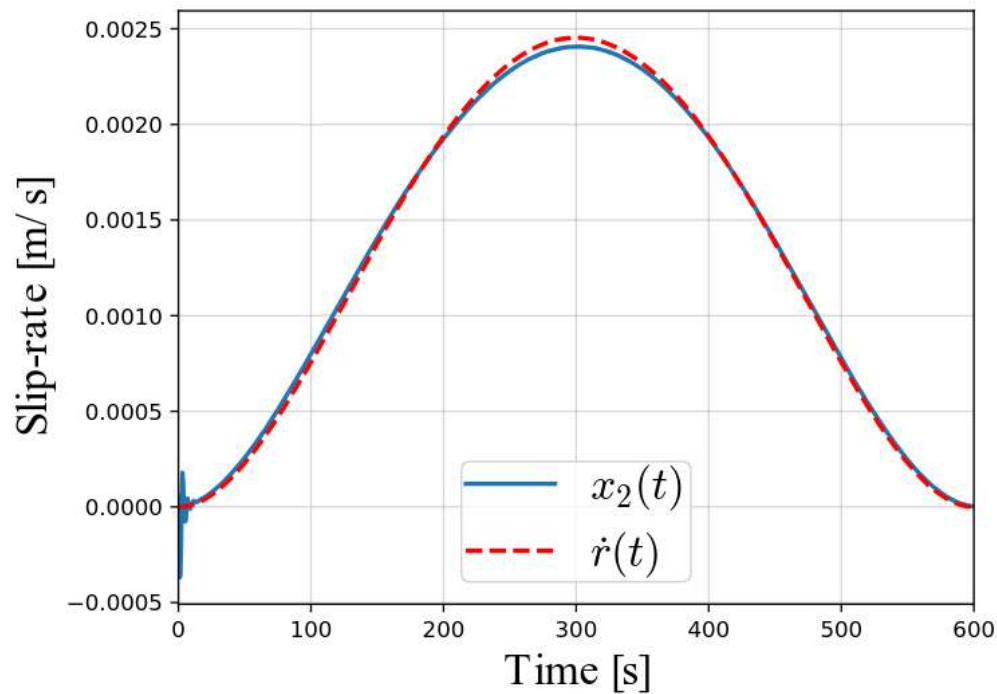
... liberating the stored energy with **high slip-rate**.

## PID control + RL ( $p \neq 0$ )

The slip follows a reference to reach the same final value over a longer time...





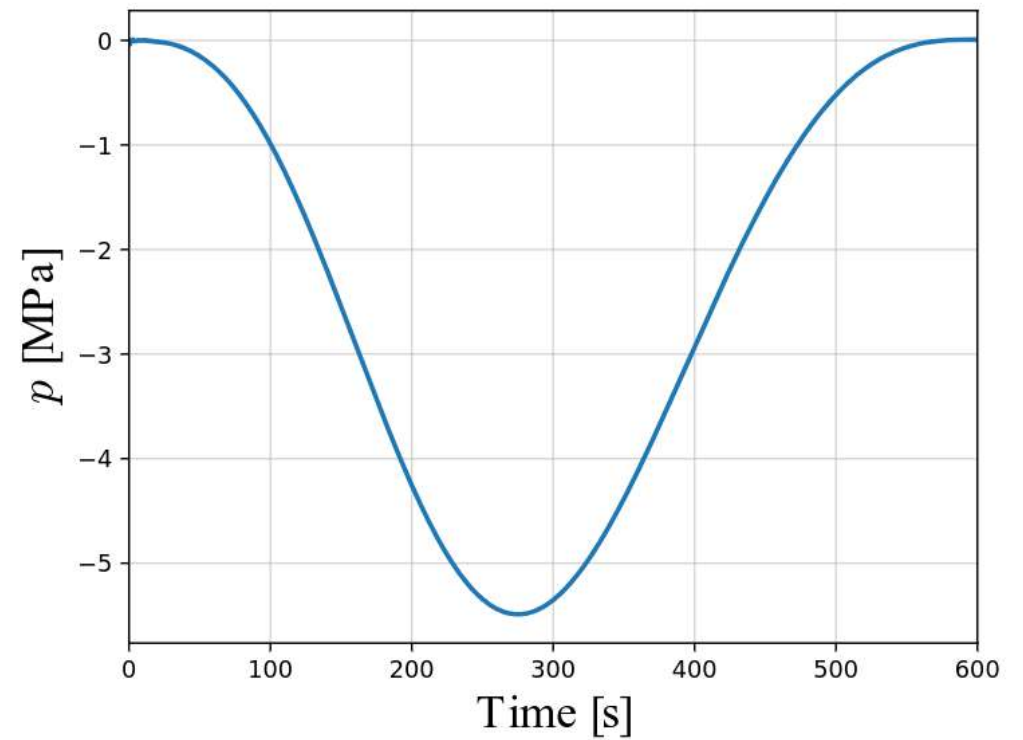


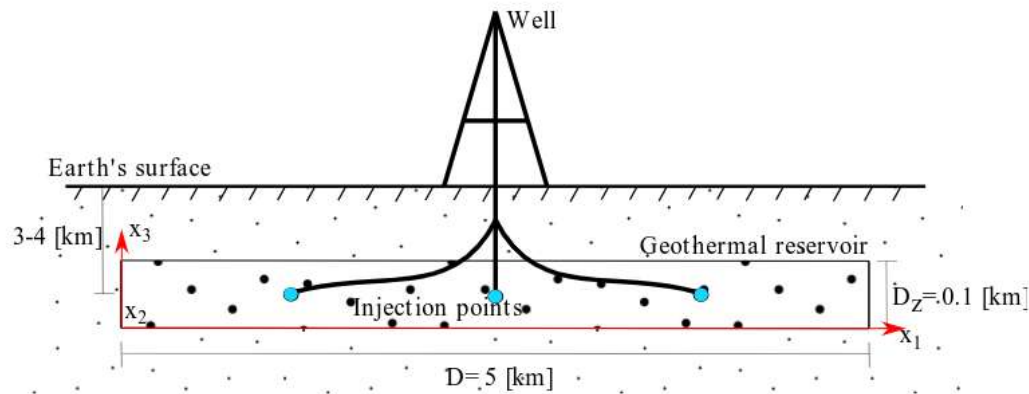
## PID control + RL ( $p \neq 0$ )

... the slip-rate follows a **slow reference** liberating the stored energy in a desired time (**600 [s]**).

## PID control + RL ( $p \neq 0$ )

The pressure was **automatically adjusted** with PID control adjusted by the RL algorithm, **without knowing anything about the system**.





## Geothermal reservoir

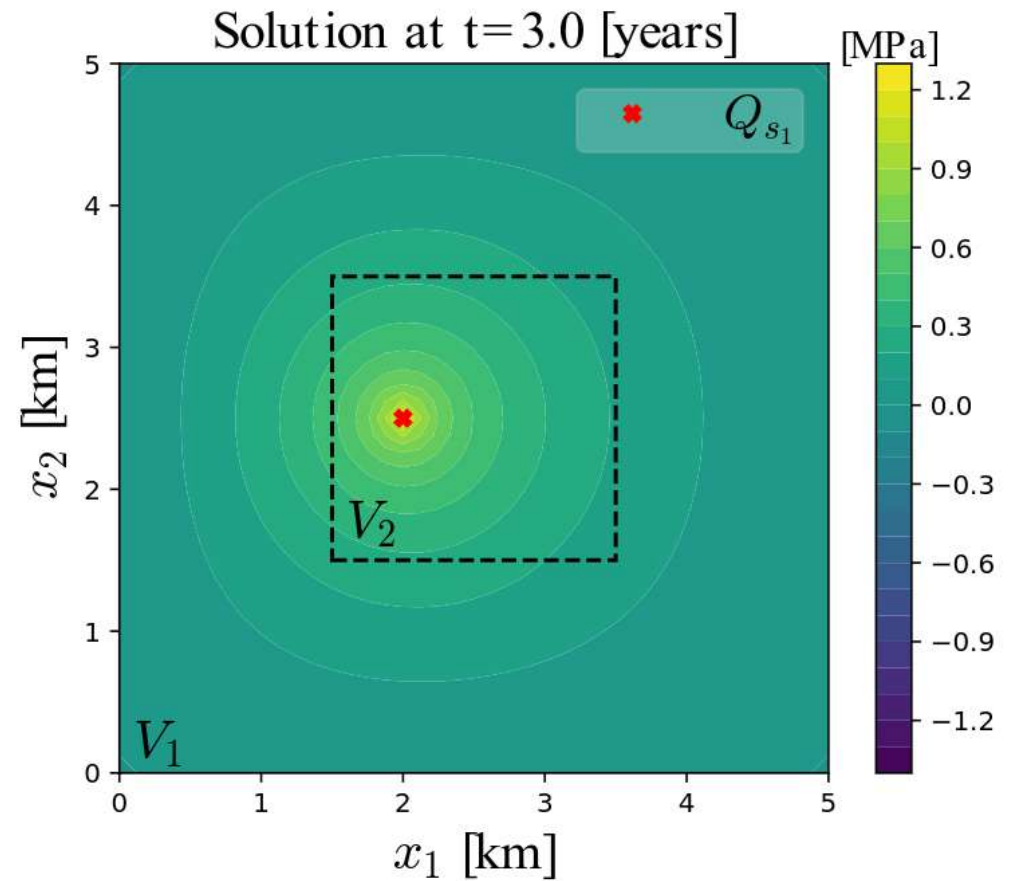
States:  $R_i$  - seismicity rate in  $i$ -regions  $V_i$

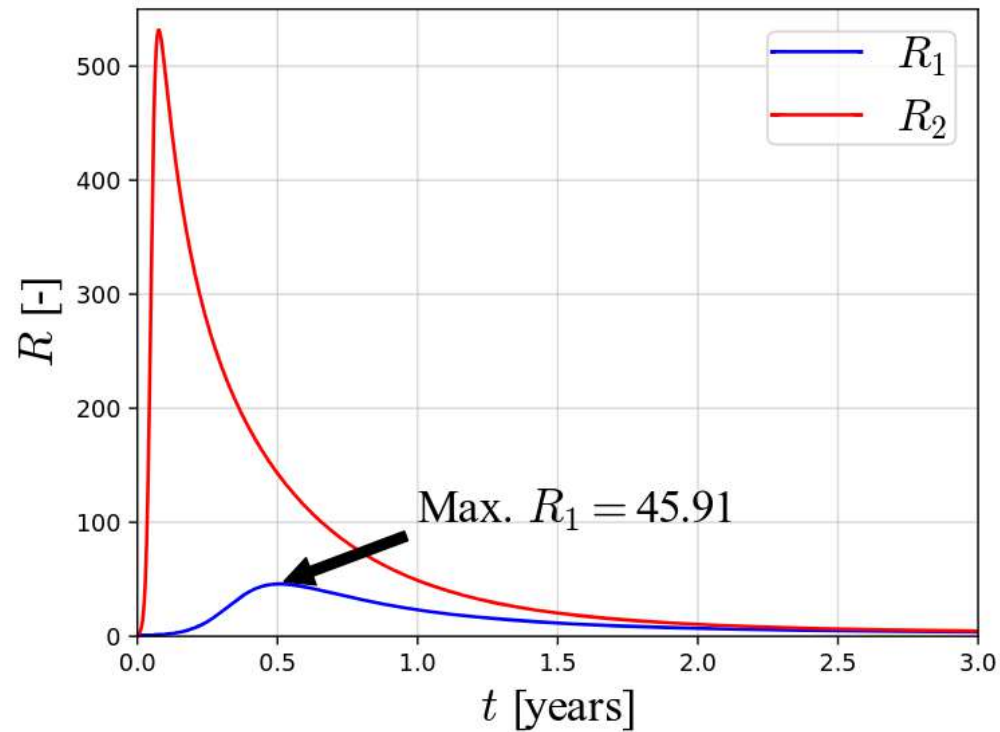
Control:  $Q_{c_i}$  - fluid fluxes of the  $i$ -injection points

Goal: Prevent the number of earthquakes (high seismicity rate) over the  $i$ -regions  $V_i$ .

## No control ( $Q_{ci} = 0$ )

The pressure over the reservoir reaches a steady state.



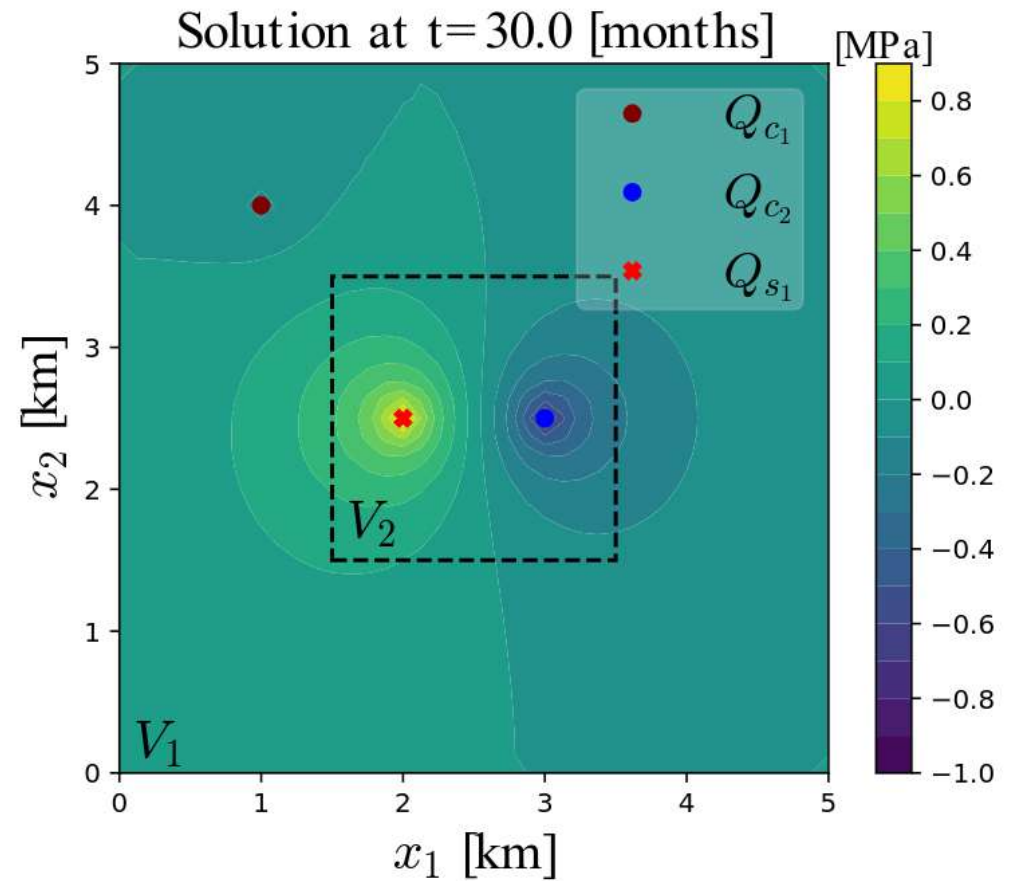


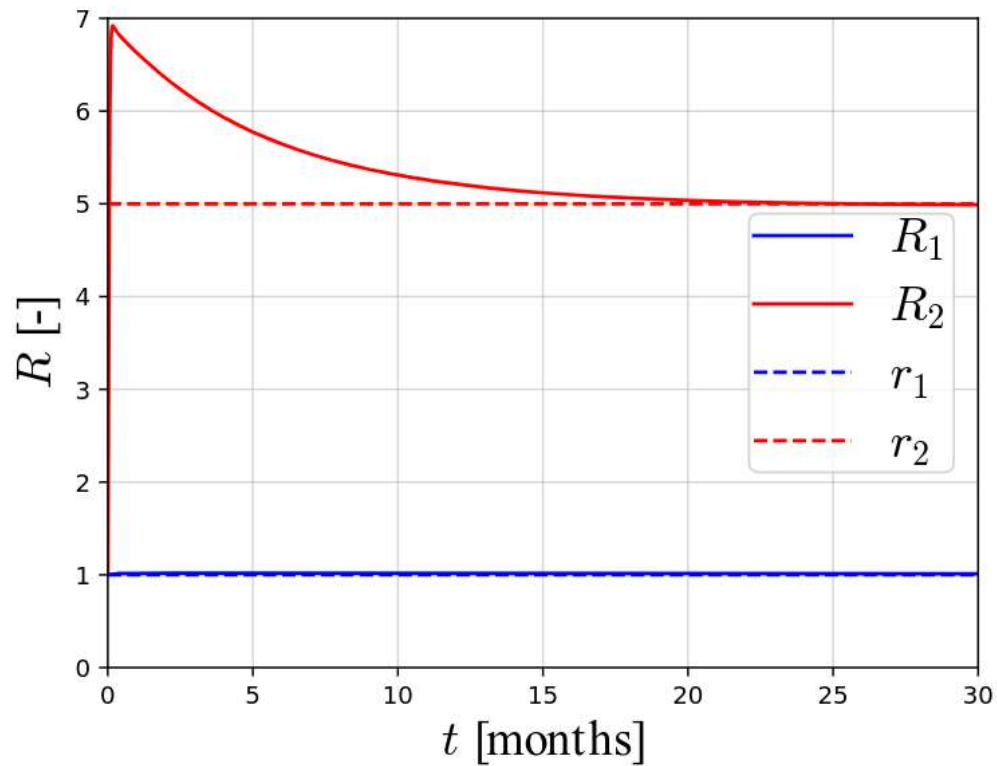
**No control ( $Q_{c_i} = 0$ )**

**45.91 more earthquakes** of a given magnitude in a determined time in the surrounding region!

## PI control + RL ( $Q_{c_i} \neq 0$ )

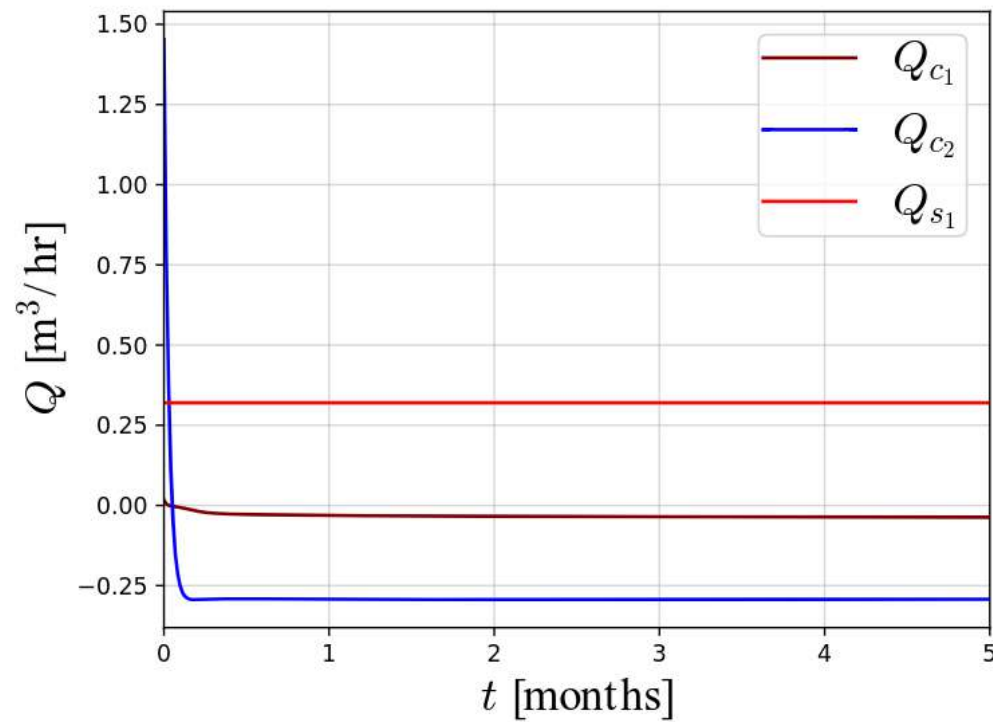
The pressure over the reservoir reaches a steady state **faster** than before.





## PI control + RL ( $Q_{c_i} \neq 0$ )

The seismicity rate in both regions reached the desired reference, **optimizing energy production while avoiding earthquakes.**



## PI control + RL ( $Q_{c_i} \neq 0$ )

The two control fluxes were **automatically adjusted** with PI control adjusted by the RL algorithm, **without knowing anything about the system**.





# Thank you for your attention!

[alexandros.stathas@ec-nantes.fr](mailto:alexandros.stathas@ec-nantes.fr)

[diego.gutierrez-oribio@ec-nantes.fr](mailto:diego.gutierrez-oribio@ec-nantes.fr)

[ioannis.stefanou@ec-nantes.fr](mailto:ioannis.stefanou@ec-nantes.fr)