

# Reinforcement learning

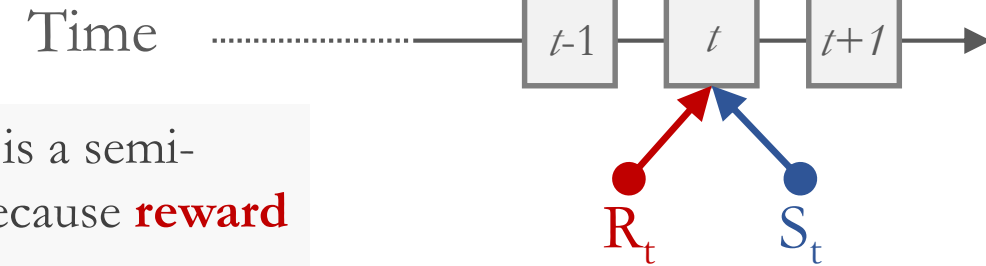
Introduction to reinforcement learning and deep reinforcement learning

Markov Decision Process (MDP)

# Reinforcement learning

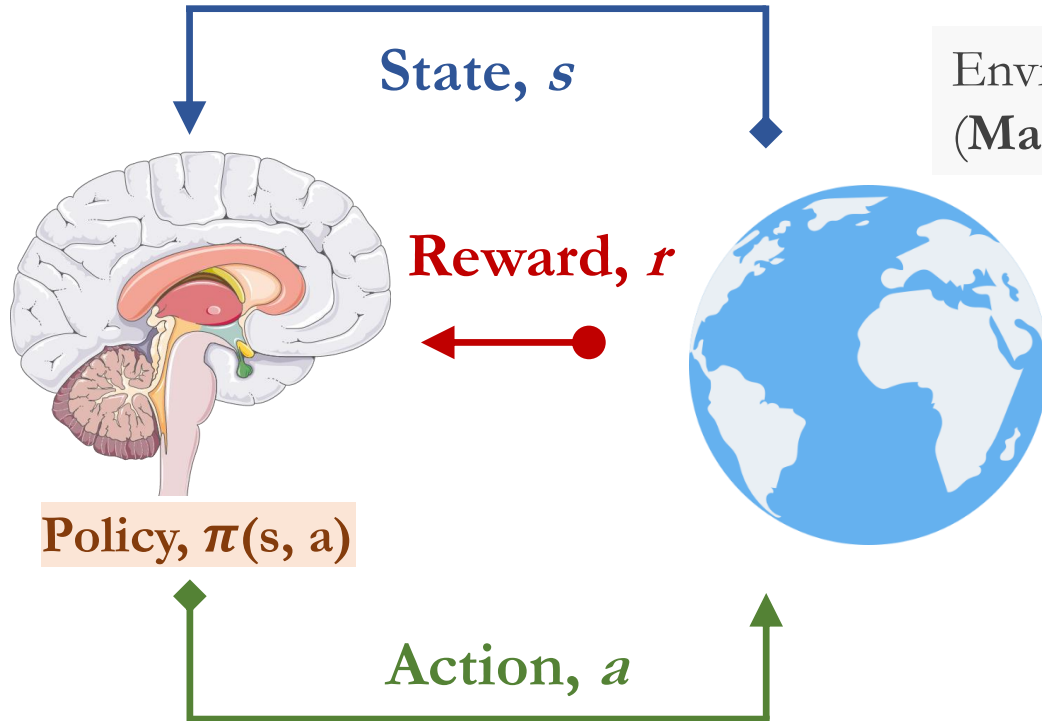
Reinforcement learning is a framework for learning how to interact with the environment from experience.

Most of the time, RL is a semi-supervised learning because **reward** is time-delayed

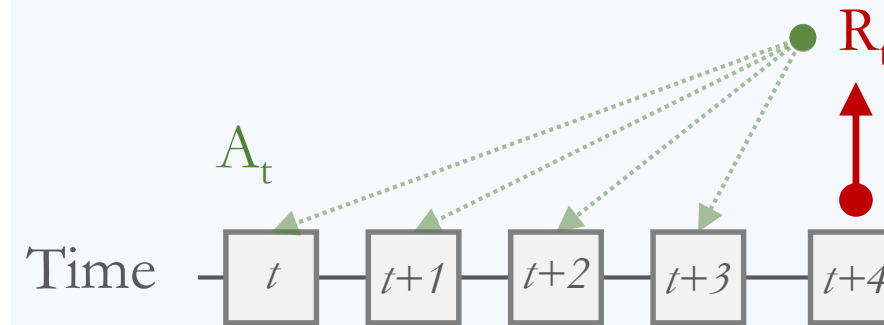


Exploration | Exploitation

Environment is modelled as probabilistic  
(**Markov Decision Process, MDP**)



Credit Assignment Problem



Source: <https://www.youtube.com/watch?v=0MNVhXEX9to>



Advanced cognitive modeling • Spring 2021

• Nicolas Legrand • Postdoctoral fellow • Embodied Computation Group



ECG  
embodied  
computation  
group

AARHUS UNIVERSITY

# Key concepts

**Model:** predict what the environment will do next.

$$p(s', r | s, a) = P(S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a)$$

**Value function:** prediction of expected rewards.

$$v_{\pi}(s) = \mathbb{E}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots | S_t = s]$$

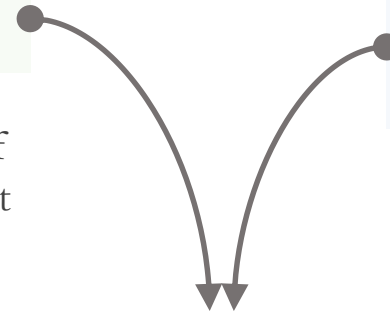
*Discount rate*

The value of a state **s** given a policy  $\pi$  is my expectation of how much reward I will get in the future if I start in that state and enact that policy.

**Policy:** how the agent pick its actions.

**Deterministic**       $\alpha = \pi(s)$

**Stochastic**       $\alpha \sim \pi(a|s)$



**Q-learning**

$Q^{\pi}(s, a)$  = quality of state/action pair

$$Q(s, a) = Q^{old}(s_t, a_t) + \alpha(r_t + \max_a Q(S_{t+1}, a) - Q^{old}(s_t, a_t))$$

Given a state **s** and an action **a**, and assuming that I will do the best thing I can in the future, what is the quality of being in that state and taking that action.

**Policy learning | Value learning**

Source: <https://www.youtube.com/watch?v=K67RJH3V7Yw&list=PLMsTLcO6ettgmyLVrcPvFLYi2Rs-R4JOE&index=4>



**Advanced cognitive modeling • Spring 2021**

• Nicolas Legrand • Postdoctoral fellow • Embodied Computation Group



ECG  
embodied  
computation  
group

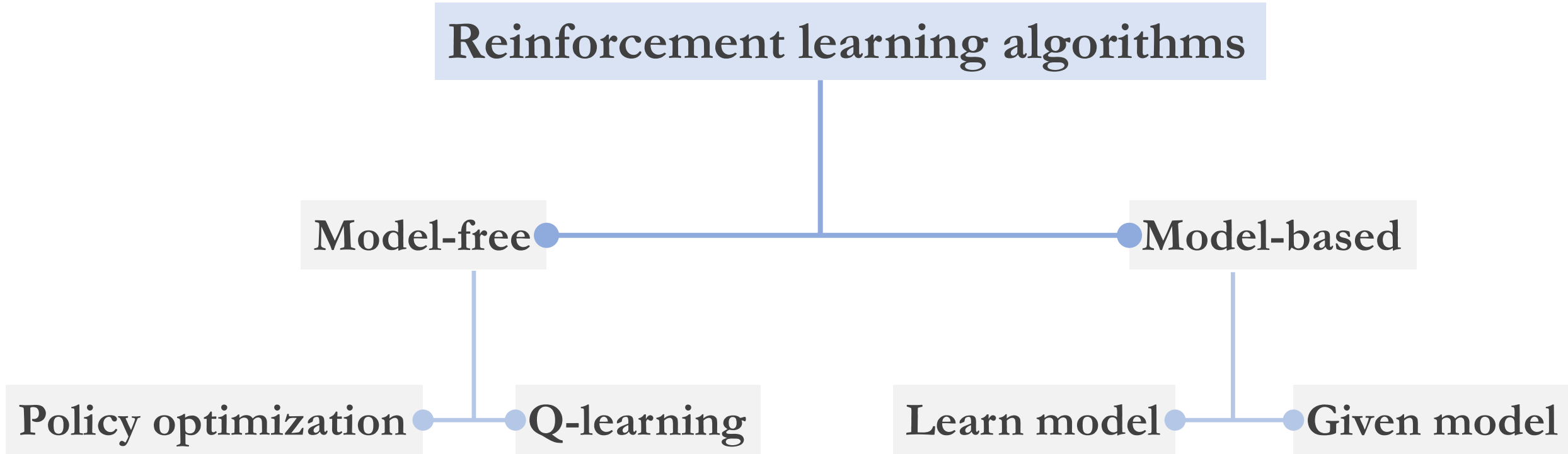
AARHUS UNIVERSITY

# RL Algorithms

Hindsight Experience Replay

Save all behaviors and code reward for different goal.

[https://www.youtube.com/watch?v=0Ey02HT\\_1Ho](https://www.youtube.com/watch?v=0Ey02HT_1Ho)



Advanced cognitive modeling • Spring 2021

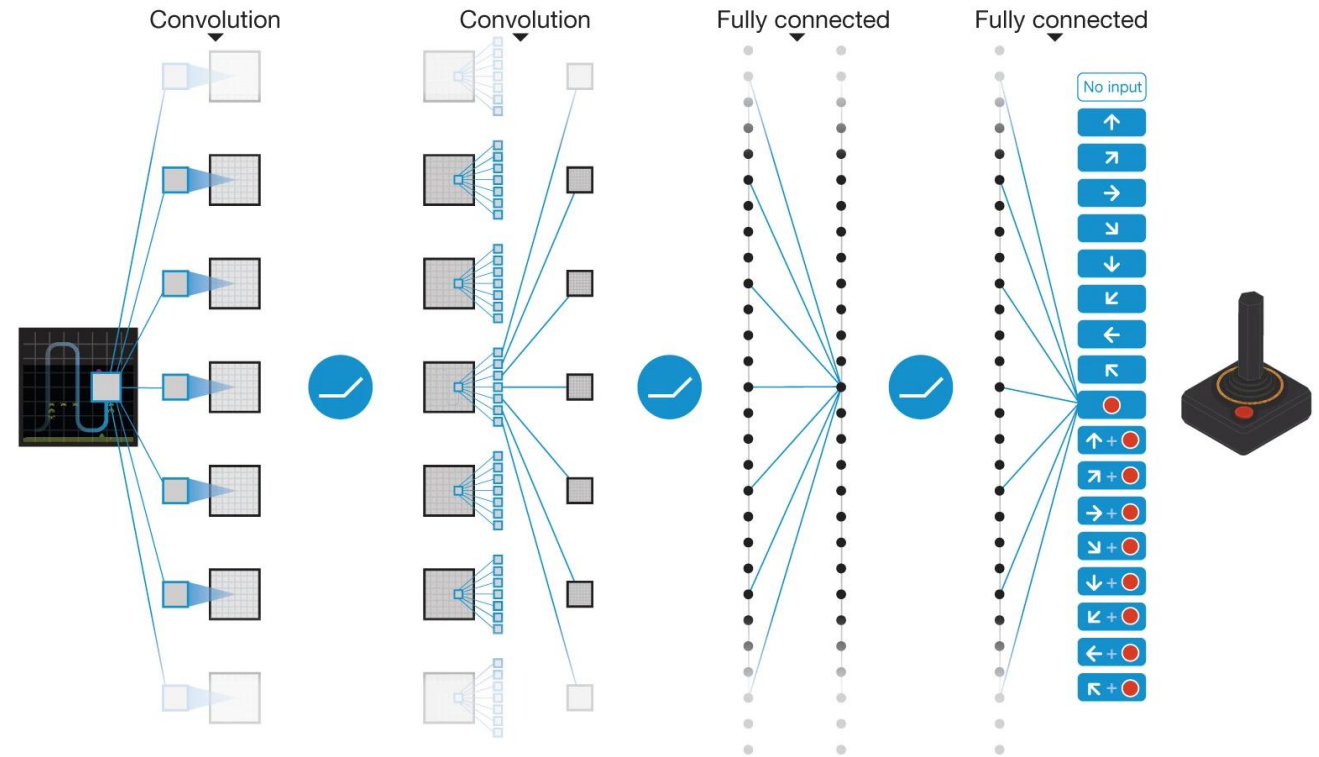
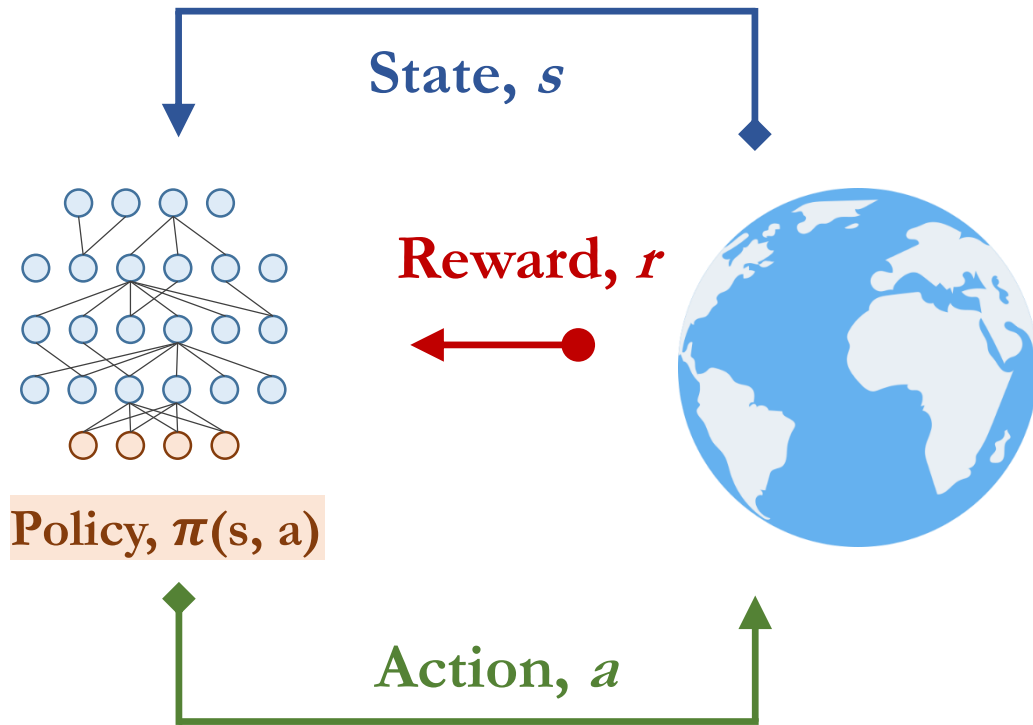
• Nicolas Legrand • Postdoctoral fellow • Embodied Computation Group



ECG  
embodied  
computation  
group

AARHUS UNIVERSITY

# Deep reinforcement learning



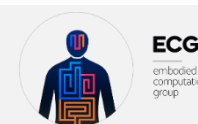
Mnih et al. (2015)

Source: <https://www.youtube.com/watch?v=IUiKAD6cuTA>



Advanced cognitive modeling • Spring 2021

• Nicolas Legrand • Postdoctoral fellow • Embodied Computation Group



ECG  
embodied  
computation  
group

AARHUS UNIVERSITY

# Examples

## Hide and seek

<https://www.youtube.com/watch?v=Lu56xVlZ40M>

## Flexible muscle-based locomotion for bipedal creatures

<https://vimeo.com/79098420>

## Atari video games

<https://www.youtube.com/watch?v=TmPfTpjtdgg&t=43s>

## AlphaGo Move 37

<https://www.youtube.com/watch?v=JNrXgpSEEIE>

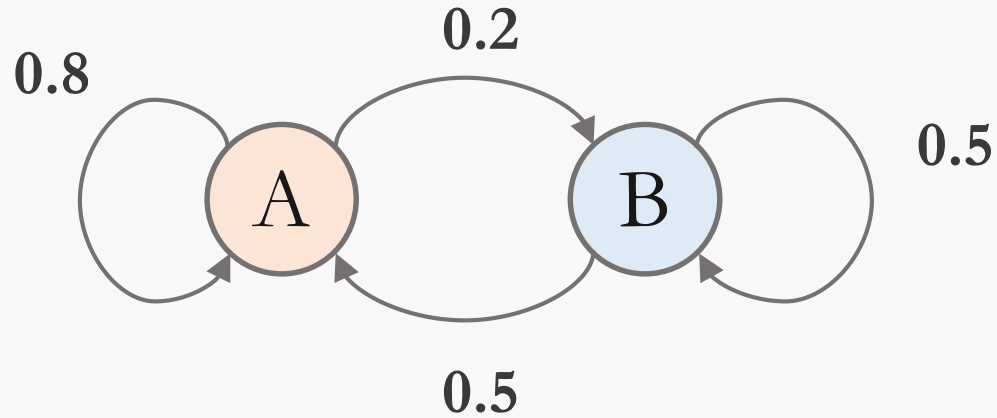
## Cart-Pole

<https://www.youtube.com/watch?v=XiigTGKZfks>



# Markov Decision Process

## Markov chains

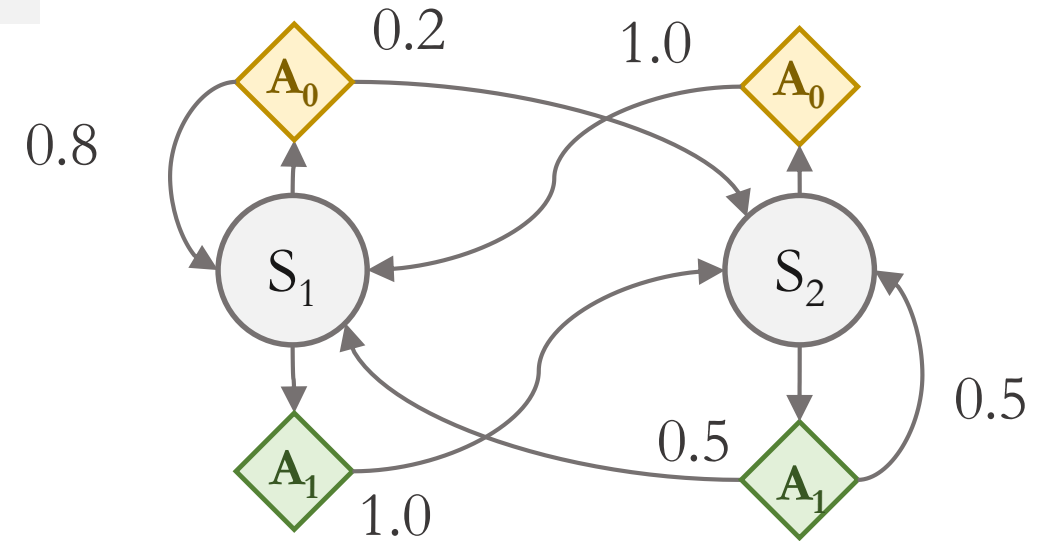


## Markov property

$$P(R_{n+1} | R_1, R_2 \dots R_n) = P(R_{n+1} | R_n)$$

*Almost all reinforcement learning problems can be modeled as MDP.*

## Markov decision process



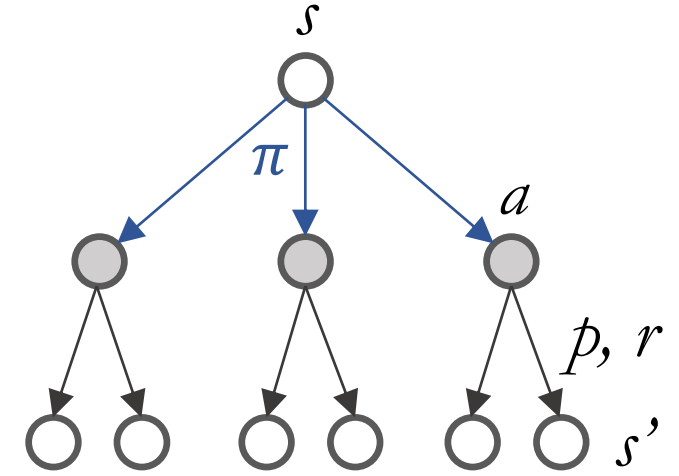
s	a	s'	p(s'   s, a)	r(s, a, s')
S1	A0	S1	0.8	1
S1	A0	S2	0.2	1
S1	A1	S2	1.0	1
S2	A0	S1	1.0	1
S2	A1	S2	0.5	1
S2	A0	S1	1.0	1



# The dynamic function

$$p(s', r | s, a) = P(S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a)$$

The probability of going from state  $s$  to state  $s'$ , getting the reward  $r$ , only depend on state  $s$  and the action  $a$  initiated by the agent.



$$p(s' | s, a) = P(S_t = s' | S_{t-1} = s, A_{t-1} = a) = \sum_{r \in \mathcal{R}} p(s', r | s, a)$$

A corollary of this is that the probability of going from state  $s$  to state  $s'$ , only depend on state  $s$  and the action  $a$  initiated by the agent, considering all the possible rewards  $r$ .

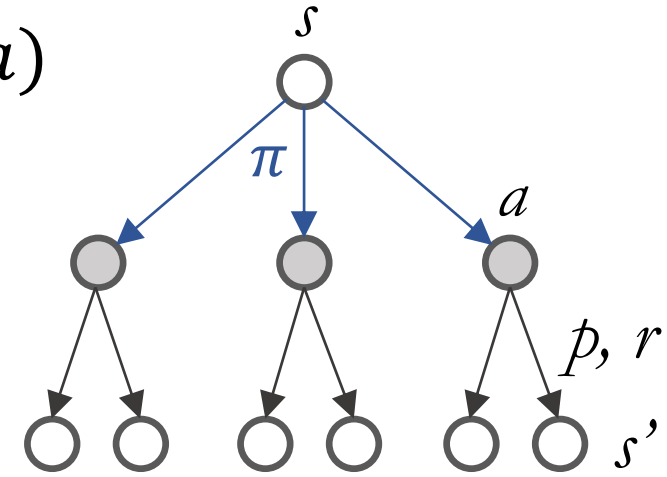




# The reward function

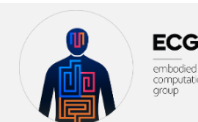
$$r(s, a) = \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

The reward value that we can expect when making action  $a$  while in state  $s$  is the weighted sum of possible rewards and their probabilities.



$$r(s, a, s') = \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

The reward value that we can expect when making action  $a$  while in state  $s$  and going to state  $s'$  is the weighted sum of possible rewards by the ratio of their probabilities.



# Goals and return

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

The return is the sum of rewards. An agent tries to maximize the expected return.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

This works well for *episodic task* that have a finite number of states. For *continuing tasks*, we use a **discounting factor**.

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

Returns at successive time steps are related to each other in a way that is important for the theory and algorithms of reinforcement learning.

*Consistency condition*



# Value function

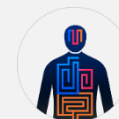
Value functions estimate how good it is for the agent to be in a given state.

Value functions are defined with respect to particular ways of acting, called policies.

$\pi(a | s)$  is the probability of performing action  $a$  in state  $s$ .

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] = \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right]$$

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



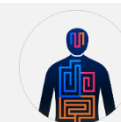
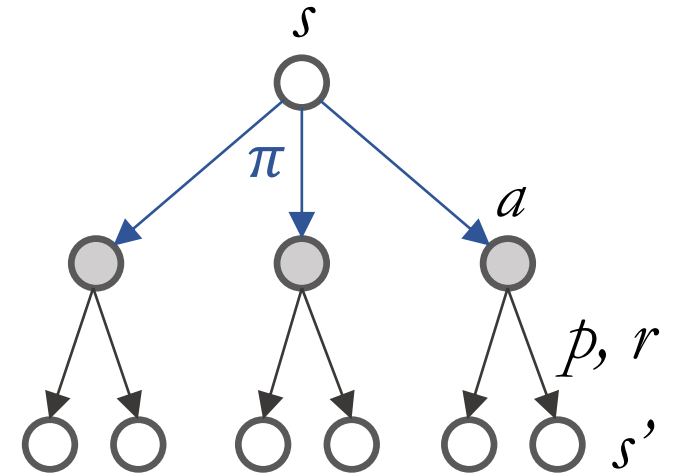
# Action-value function

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a]$$

$$= \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right]$$



The value function of taking action  $a$  in state  $s$  under a policy  $\pi$ , denoted  $q_{\pi}(s, a)$ , is the expected return when starting from  $s$ , taking action  $a$ , and following  $\pi$  thereafter.



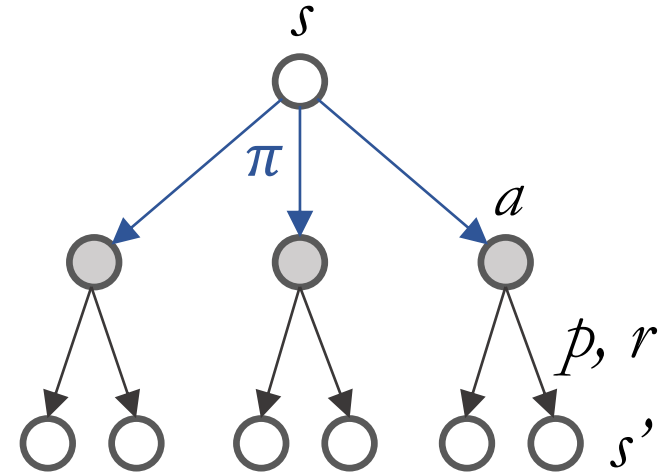
# Bellman expectation equation

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s]$$

$$= \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a) [r + \gamma \mathbb{E}_{\pi}[G_{t+1} \mid S_{t+1} = s']]$$

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



The Bellman equation averages over all the possibilities, weighting each by its probability of occurring.



# Optimal policies and optimal value functions

Optimal state-value function

$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

There is always at least one policy that is better than or equal to all other policies. This is an optimal policy, we denote by  $\pi_*$ .

Optimal policies also share the same optimal action-value function, denoted  $q_*$

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

We can write  $q_*$  in term of  $v_*$

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$



# Bellman optimality equation

$$v_*(s) = \max_{a \in A(s)} q_{\pi^*}(s, a)$$

$$= \max_{a \in A} q_{\pi^*}(s, a)$$

$$= \max_a \mathbb{E}_{\pi^*}[G_t \mid S_t = s, A_t = a]$$

$$= \max_a \mathbb{E}_{\pi^*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a]$$

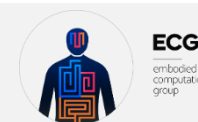
$$= \max_a \mathbb{E}_{\pi^*}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

$$= \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_*(s')]$$

The Bellman equation averages over all the possibilities, weighting each by its probability of occurring.

The value of a state under an optimal policy must equal the expected return for the best action from that state.

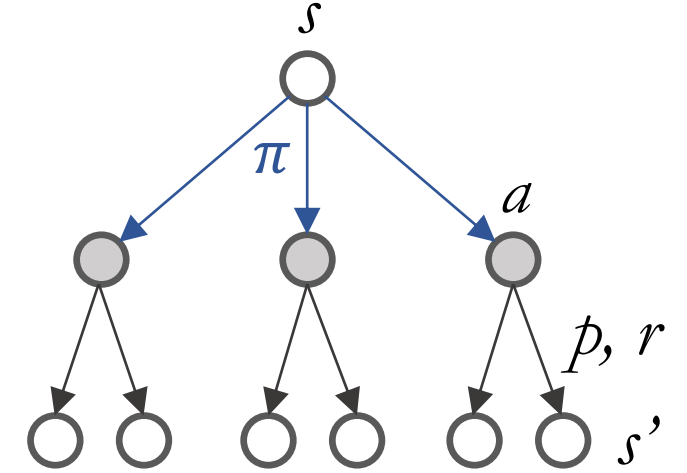
The Bellman optimality equation is actually a system of equations, one for each state, so if there are  $n$  states, then there are  $n$  equations in  $n$  unknowns.



# Bellman optimality equation

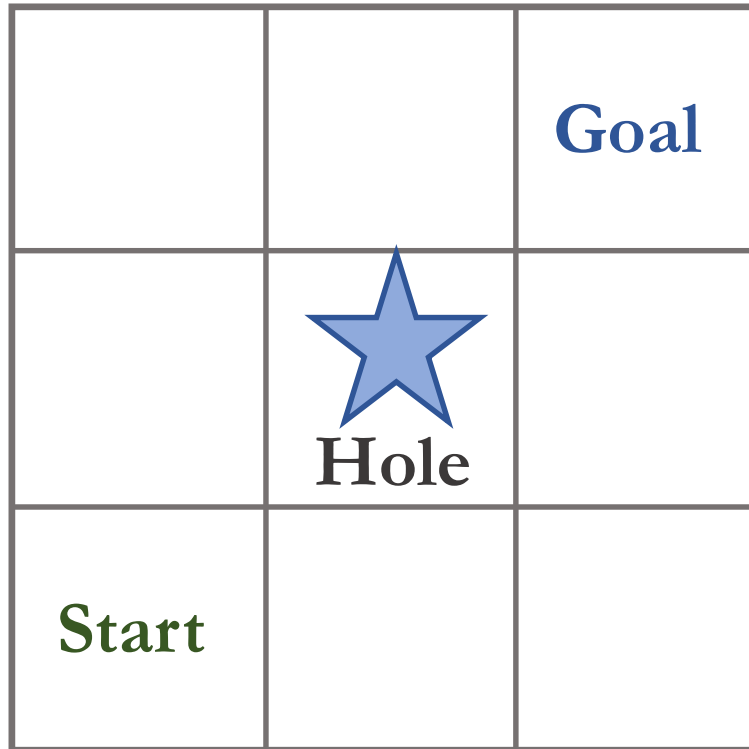
The Bellman optimality equation for  $q_*$  is

$$\begin{aligned} q_*(s, a) &= \mathbb{E}_{\pi_*}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma \max_{a'} q_*(s', a)] \end{aligned}$$





# Frozen Lake



Advanced cognitive modeling • Spring 2021

• Nicolas Legrand • Postdoctoral fellow • Embodied Computation Group



ECG  
embodied  
computation  
group

AARHUS UNIVERSITY