Markov Decision Processes

Philipp Schwartenbeck

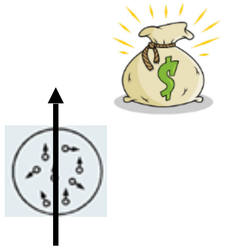
MPI for Biological Cybernetics, Tuebingen
AI Centre Tuebingen

Computational Psychiatry Course 2022, TNU, Zurich

Day 3: Models of action selection

Why MDPs?

Intelligent behaviour relies on building and employing a **model of the world**

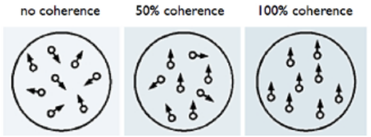


World model

Belief-state $\hat{s}$

Values Q

Agent



Reward r

Observation o

Action a

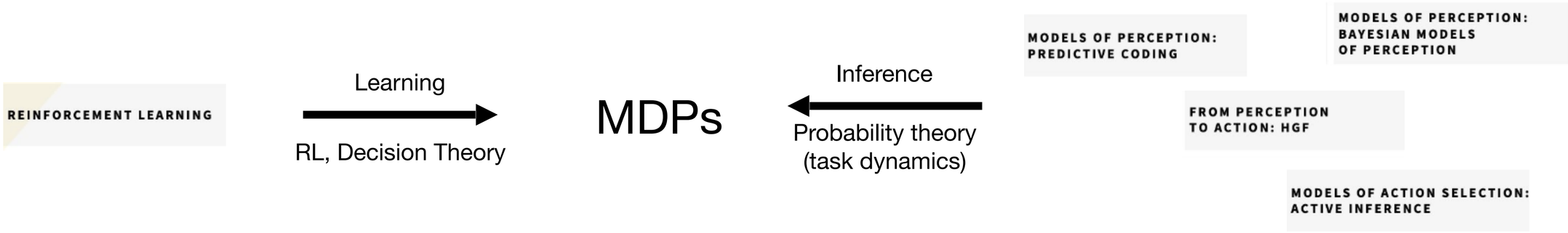
World state s

It's a good idea to play the slot machine
-> If you think $P(win)$ is high

It's a good idea to look for food
-> If you are hungry

It's a good idea to gain information
-> If you are uncertain

Structure and Outline



Outline

I. Basics: Markov Decision Processes (MDPs)
   - What's the basic structure of an MPD?

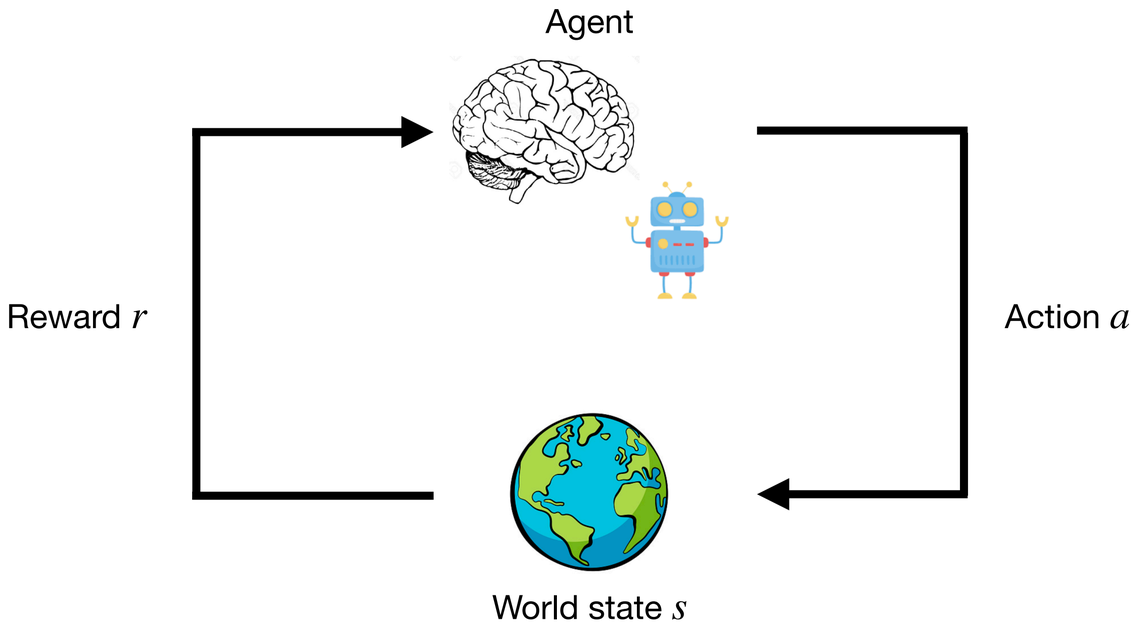
II. Solving MDPs
   - How do we use an MDP to solve problems?

III. Extensions and Outlook
   - What are important additions, aspects of research and applications?

Basics: Markov Decision Processes (MDPs)

MDPs - basic setup

Lets' simplify the picture a bit:

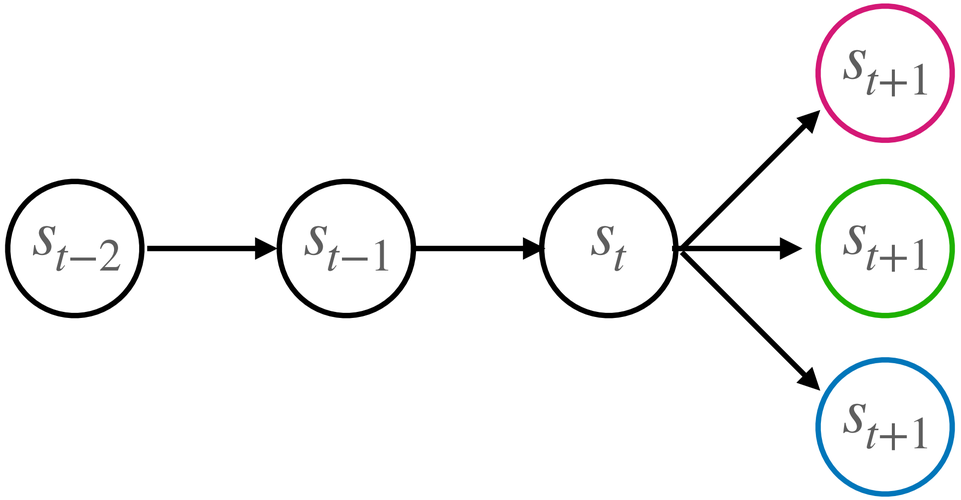


The theory of MDPs is concerned with modelling this interaction

Markov Process → Markov Reward Process → Markov Decision Process (MDP)

Markov Process

We are dealing with problems that we can model as a sequence of discrete states:



Fundamental property: **Markov property**

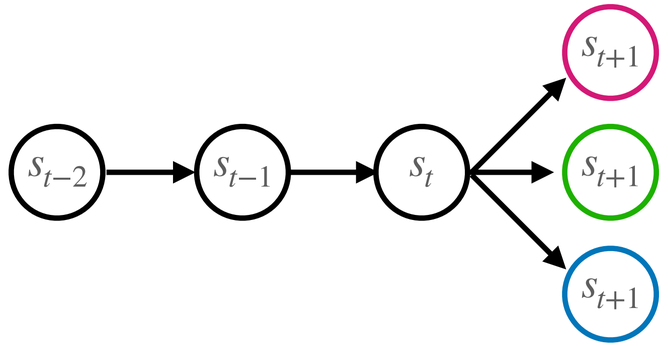
$$P(s_{t+1} = s \mid s_t, s_{t-1}, s_{t-2}, \dots) = P(s_{t+1} = s \mid s_t)$$

“The future is independent of the past given the present”

Markov Process

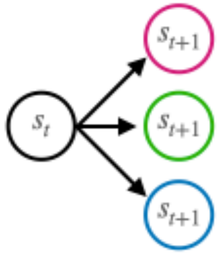
Fundamental property: **Markov property**

$$P(s_{t+1} = s \mid s_t, s_{t-1}, s_{t-2}, \dots) = P(s_{t+1} = s \mid s_t)$$



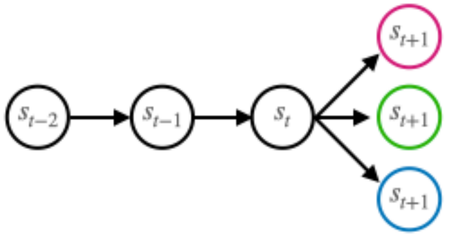
We only need the current state to predict the next state

Predict next state based on current state



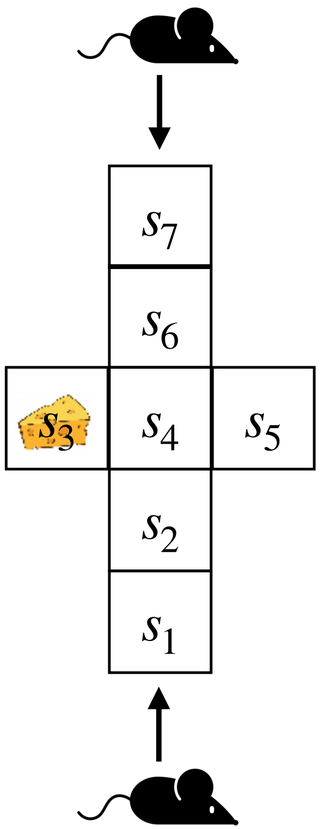
=

Predict next state based on current state and all previous states



Q: Can you think of situations where this doesn't work?

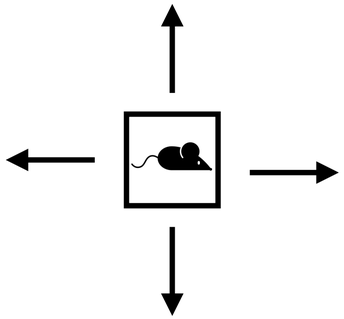
Markov Process



Let's assume a super simple problem

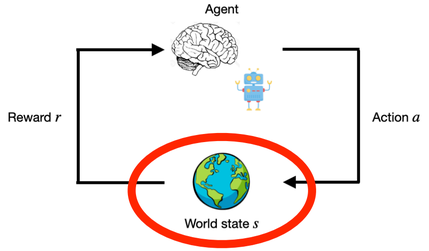
- A mouse is running through a cross-shaped maze
- Starting position either at the top or bottom of the maze
- There is a reward in one state in that maze (s_3)

In any state, the mouse can move **left**, **right**, **up** or **down**



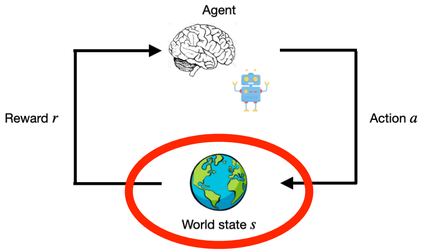
However, it turns out our mouse isn't very clever

- It has no concept of rewards
- It randomly moves around – *no active control*

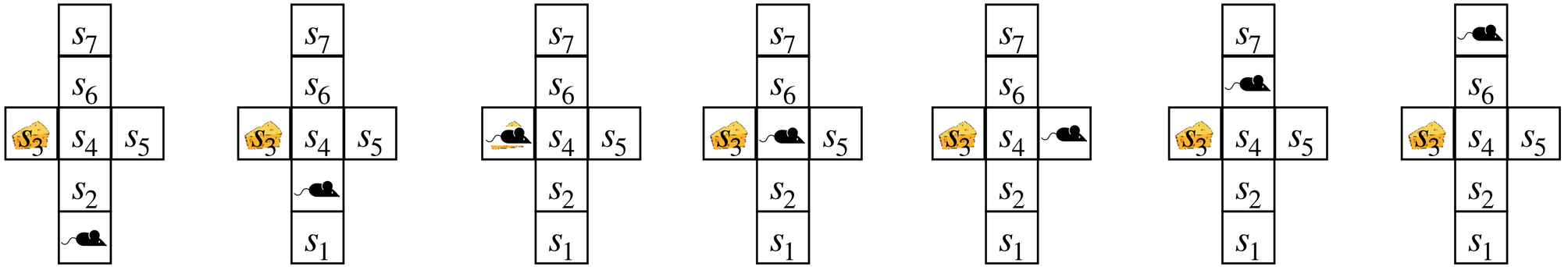


To define a **Markov Process**, we need to define a **state space** and **transition probabilities**

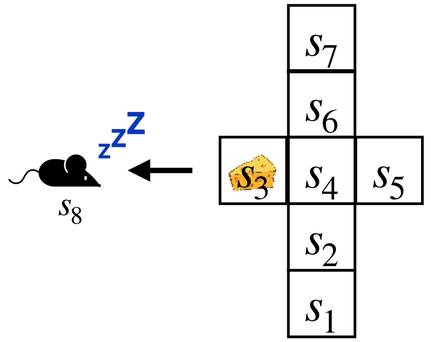
Markov Process



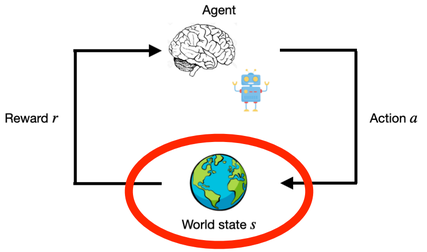
We can now define a **state space** S:



Often, we also define **terminal (absorbing) states**, such as reaching a goal

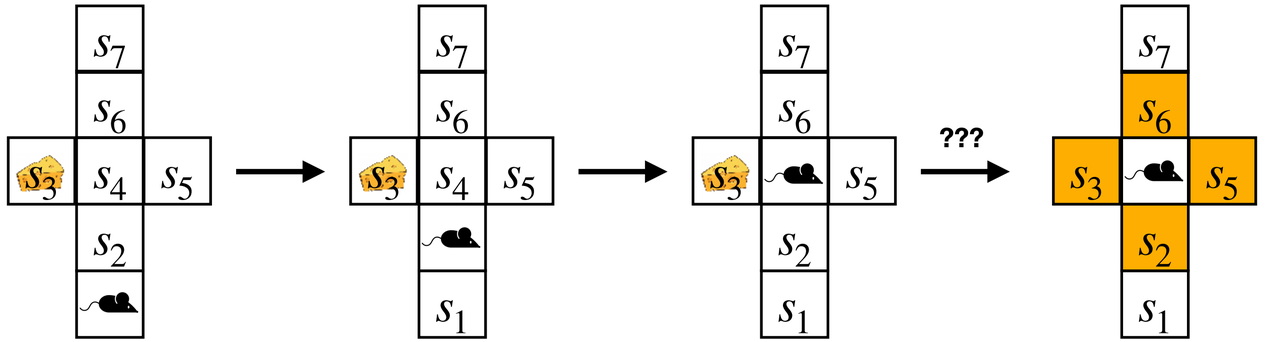


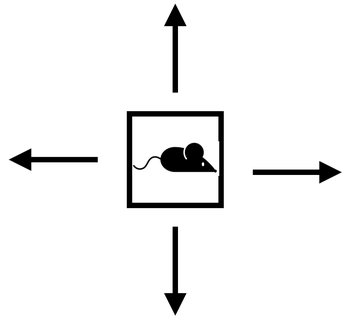
Markov Process



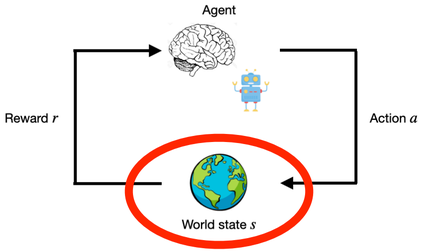
$$P(s_{t+1} = s \mid s_t, s_{t-1}, s_{t-2}, \dots) \overset{???}{=} P(s_{t+1} = s \mid s_t)$$

Does this problem have the **Markov Property**?



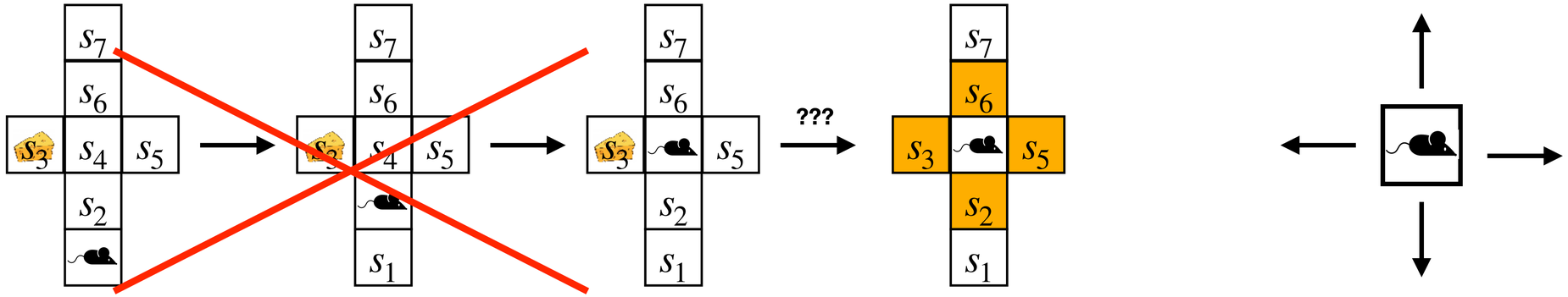


Markov Process



$$P(s_{t+1} = s \mid s_t, s_{t-1}, s_{t-2}, \dots) \overset{!}{=} P(s_{t+1} = s \mid s_t)$$

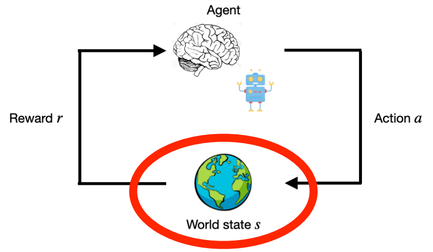
Does this problem have the **Markov Property**?



We don’t need to know the past states to predict possible next states in the maze

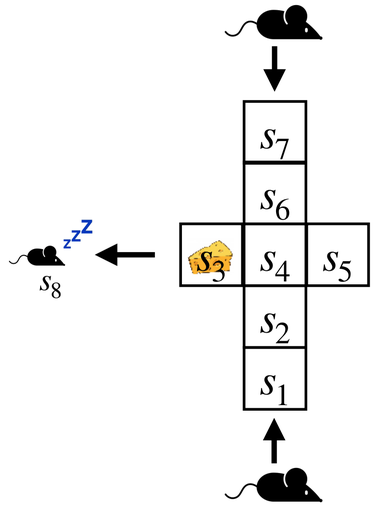
- We only need the current state!

Markov Process



This allows us to define **transition probabilities** P: $P(s_{t+1} = s' \mid s_t = s)$

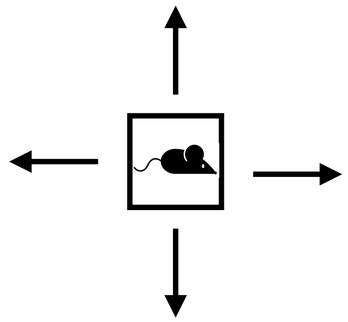
These specify the **dynamics** between the environment states



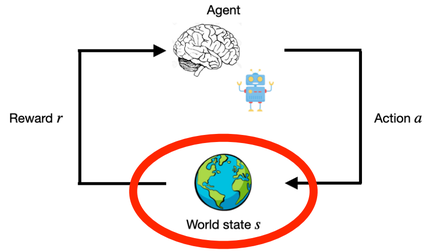
To

From	s_1	s_2	s_3	s_4	s_5	s_6	s_7	zzz
s_1	3/4	1/4	0	0	0	0	0	0
s_2	1/4	2/4	0	1/4	0	0	0	0
s_3	0	0	0	0	0	0	0	1
s_4	0	1/4	1/4	1/4	1/4	0	0	0
s_5	0	0	0	1/4	3/4	0	0	0
s_6	0	0	0	0	1/4	2/4	1/4	0
s_7	0	0	0	0	0	1/4	3/4	0
zzz	0	0	0	0	0	0	0	1

Terminal (absorbing) state →

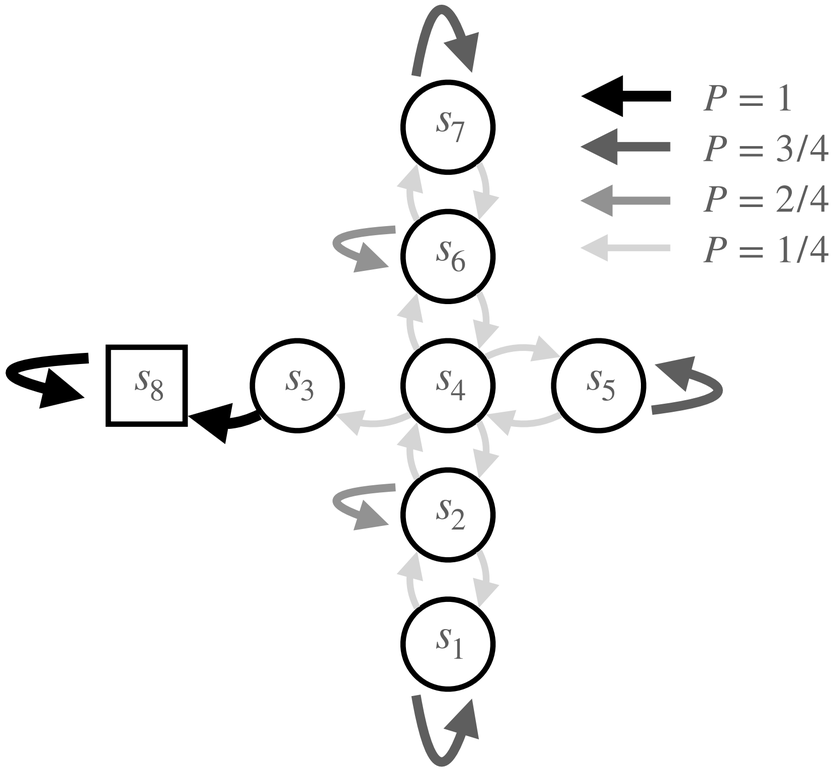


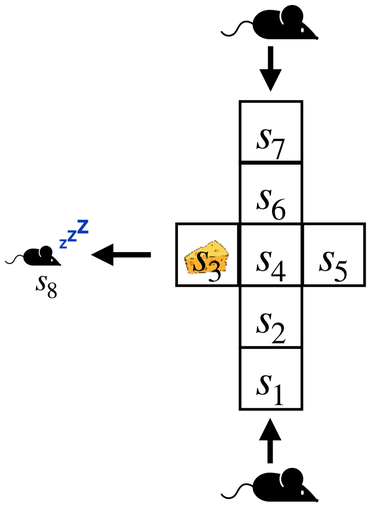
Markov Process



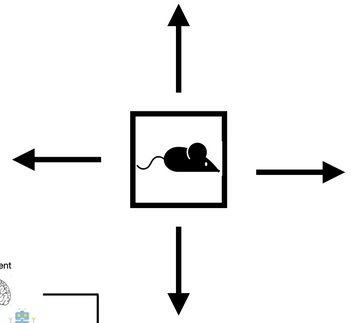
This allows us to define a **Markov Process** or **Markov Chain** (S, P) based on:

- A (finite) state space S
- Transition probabilities $P(s_{t+1} = s' \mid s_t = s)$

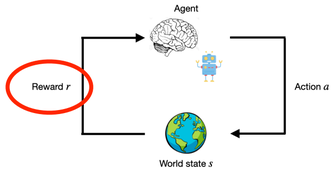




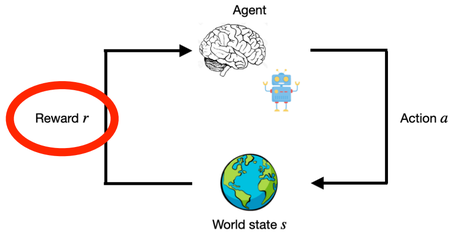
We can also illustrate our problem via a **transition graph**:



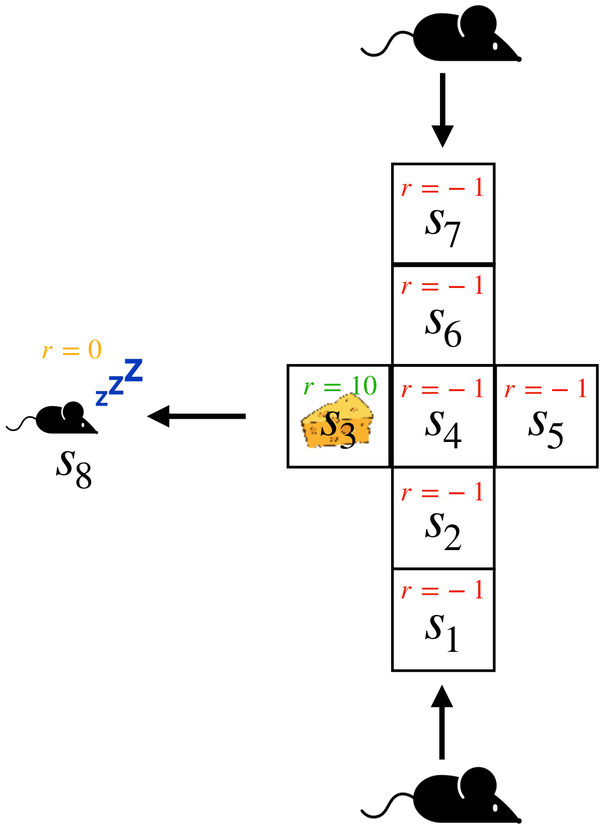
To define a **Markov Reward Process**, we need to add **rewards**

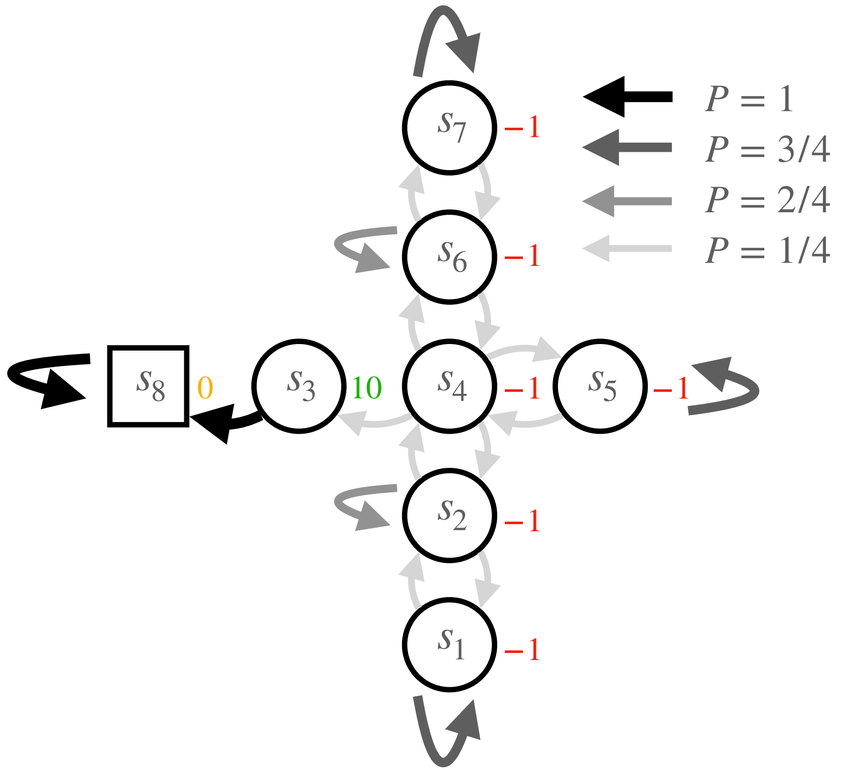


Markov Reward Process

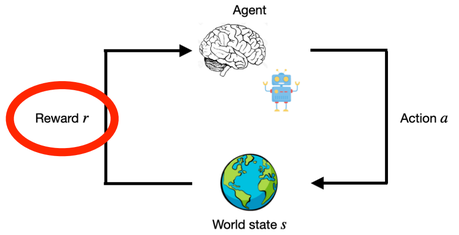


We now add a feedback signal that indicates good/bad outcomes



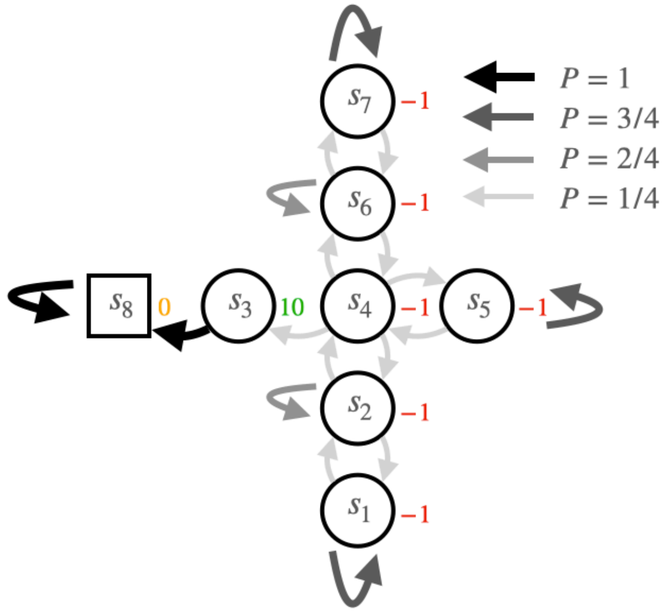


Markov Reward Process



A **Markov Reward Process** is defined based on:

- A (finite) state space S
- Transition probabilities $P(s_{t+1} = s' \mid s_t = s)$
- A **Reward Function** $R_s = \mathbb{E}[R_t \mid s_t = s]$
- A **Discount Factor** $\gamma \in [0,1]$



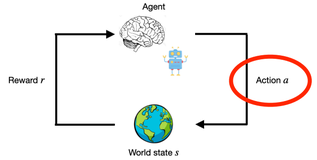
Allows us to define the '**return**':

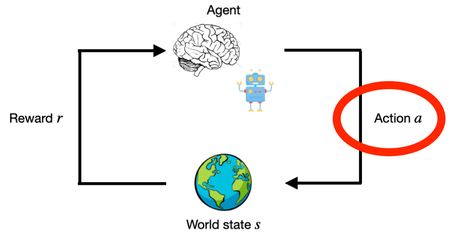
- γ determines how much we take the future into account

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

This will be useful for determining good and bad states (later..)

To define a **Markov Decision Process**, we need to add **actions**



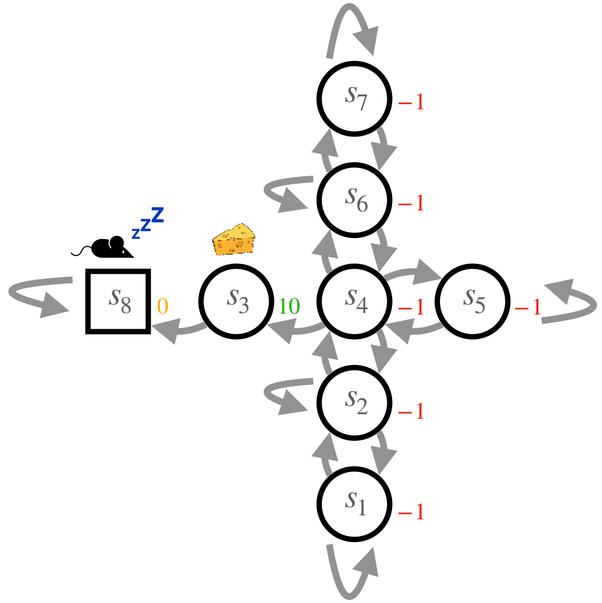


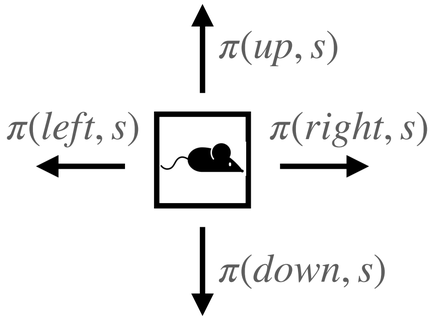
Markov Decision Process

What about agents who can actively control their environment?

A **Markov Decision Process** is defined based on:

- A (finite) state space S
- Transition probabilities $P(s_{t+1} = s' \,|\, s_t = s, \mathbf{a_t} = \mathbf{a})$
- A Reward Function $R_s = \mathbb{E}[R_t \,|\, s_t = s, \mathbf{a_t} = \mathbf{a}]$
- A Discount Factor $\gamma \in [0,1]$
- A (finite) **action space** A



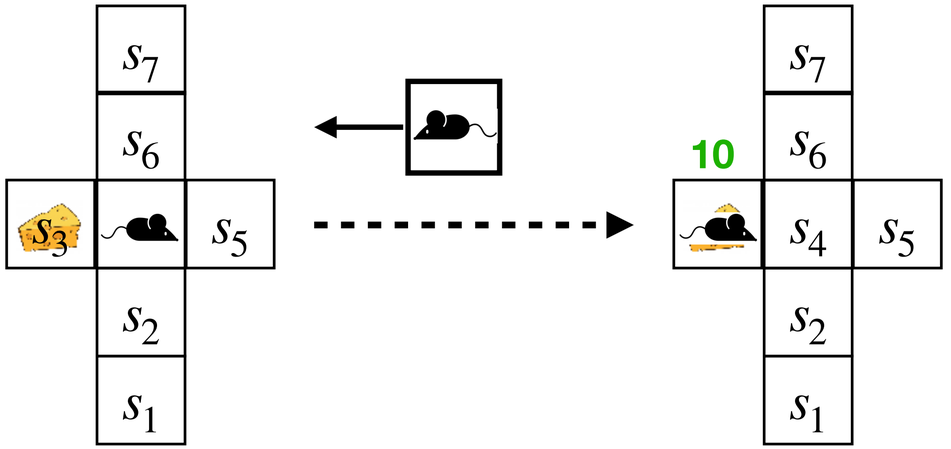


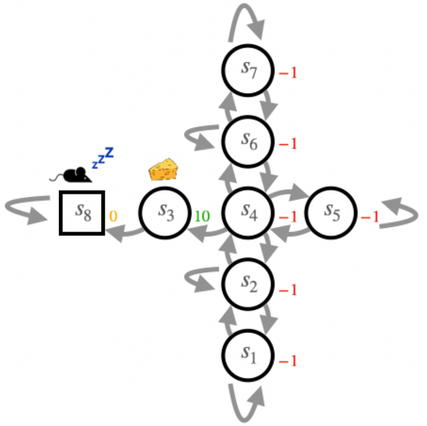
Movement not random process any more, but actively controlled via **policies**

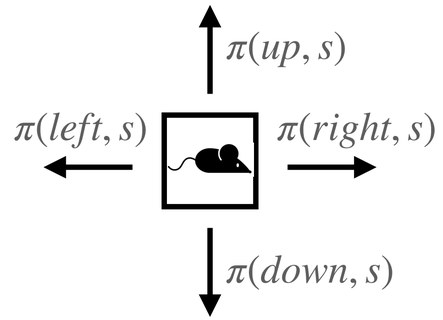
MDPs basis for model-based RL

Allows to specify all environment dynamics for decision-making problem:

$$P(s', r \mid s, a) = P(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$$







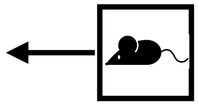
This contains a lot of information!

MDPs basis for model-based RL

$$P(s', r \mid s, a) = P(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$$

Allows to specify useful things like **state-action transition probabilities**:

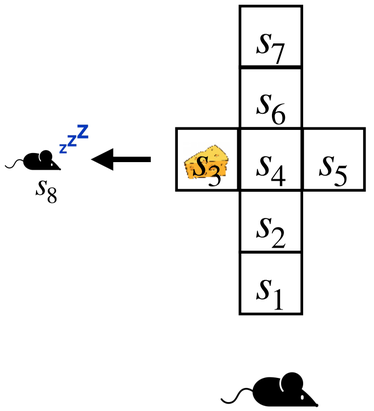
$$P(s' \mid s, a) = P(s_{t+1} = s' \mid s_t = s, a_t = a) = \sum_r P(s', r \mid s, a)$$



What happens if I move left?

$$P(s' \mid s, left) =$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	
s_1	1	0	0	0	0	0	0	0
s_2	0	1	0	0	0	0	0	0
s_3	0	0	0	0	0	0	0	1
s_4	0	0	1	0	0	0	0	0
s_5	0	0	0	1	0	0	0	0
s_6	0	0	0	0	0	1	0	0
s_7	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	1



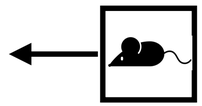
These transition probabilities depend on action!

MDPs basis for model-based RL

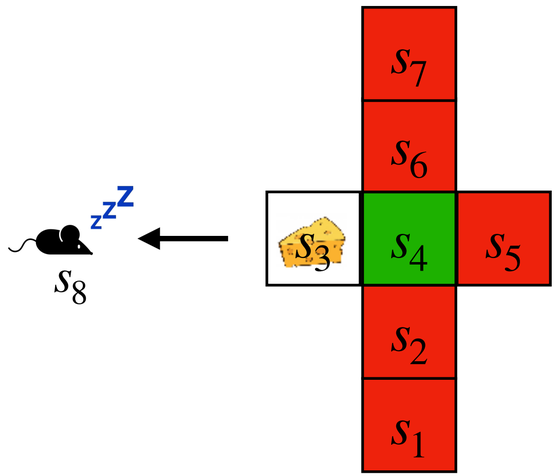
$$P(s', r \mid s, a) = P(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$$

Can also encode **expected next reward for state-action pair**
– which action is particularly good in which state?

$$r(s, a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] = \sum_r r \sum_{s'} P(s', r \mid s, a)$$



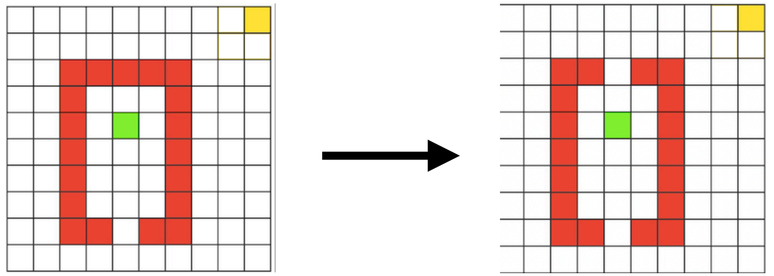
What reward will I get if I move left?



MDPs basis for model-based RL

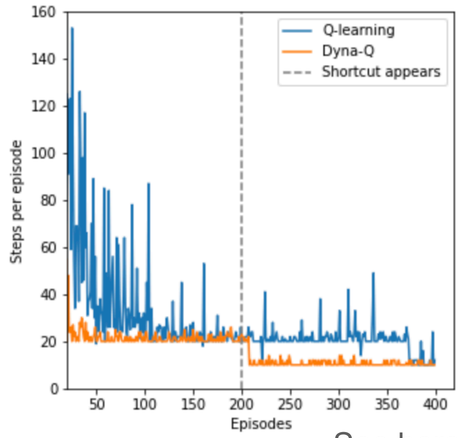
$$P(s', r \mid s, a) = P(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$$

How can we make use of such models of the world?



Learning

- Key idea: store experiences in world model $P(s', r \mid s, a)$
- Sample from this model to generate extra learning data
- This is called **DYNA-Q**



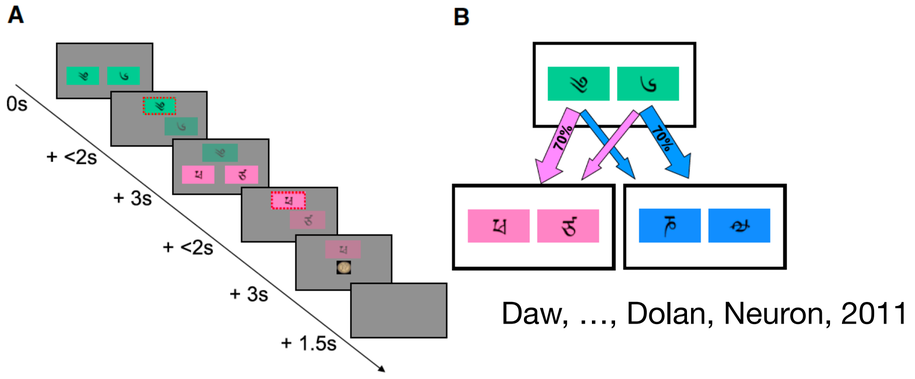
See here or here for more simulations

MDPs basis for model-based RL

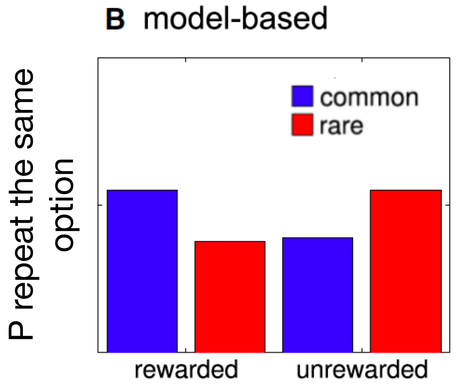
$$P(s', r \mid s, a) = P(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$$

How can we make use of such models of the world?

Planning and **action selection**



Daw, …, Dolan, Neuron, 2011



Model-based RL agent: repeat what is rewarding, but be clever

If you understand the transition probabilities, you can learn about several actions at the same time

MDPs basis for model-based RL

$$P(s', r \mid s, a) = P(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$$

How can we make use of such models of the world?

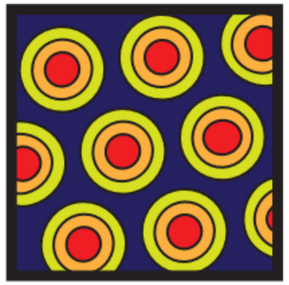
Efficient representations of the world - **cognitive maps** in the brain

Place Cell

(O'keefe & Nadel 1978)

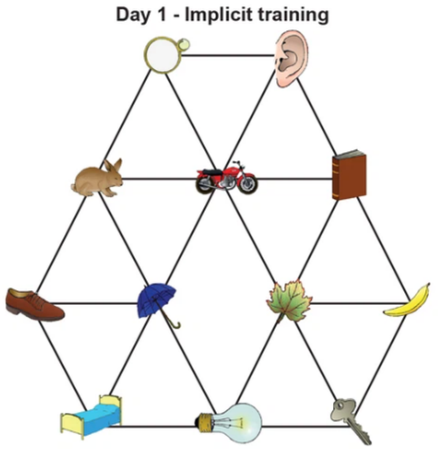
This looks like a **state**

Grid Cell



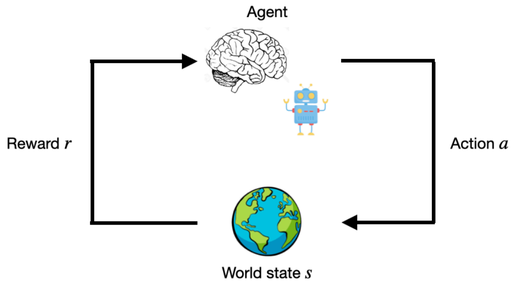
(Hafting et al., 2005)

This looks like a **transition probability**



Garvert, Dolan, Behrens eLife, 2017
Behrens et al., Neuron, 2018

MDPs interim summary



MDPs are **actively controlled Markov Processes**

- **Markov process** models evolution of states based on **Markov property** (current state summarises history)
- **Markov Reward Process** introduces concept of reward – allows to define *value* of states
- **Markov Decision Process** enables **active (optimal) control**

Allows us to model and provide full account of **environment dynamics** $P(s', r \mid s, a)$

- Fundamental for model-based decision-making

Provides basis for **optimal control** problems

- Substantial contribution to our understanding of (neural) computations underlying decision-making

So, how are MDPs useful in finding good actions?

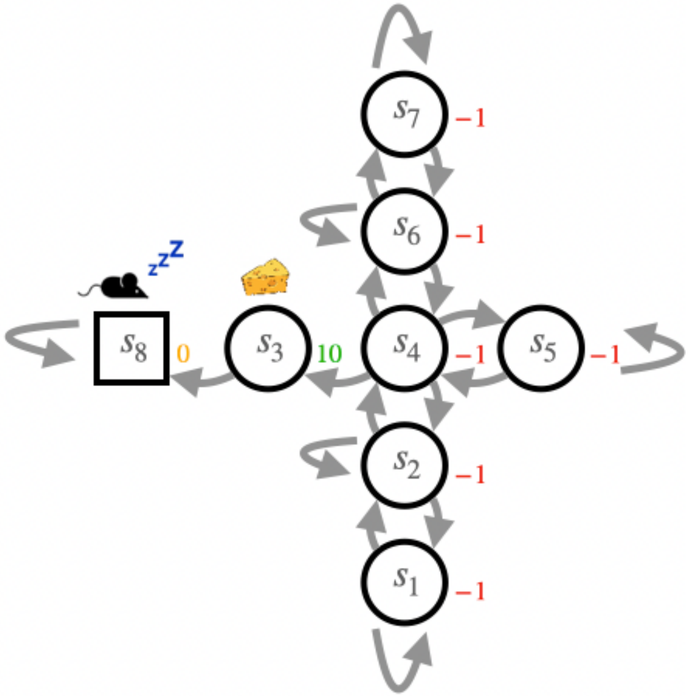
Solving MDPs

Solving MDPs: Transform return into value function

To solve MPDs, we can rely on two quantities:

‘**Return**’: $$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

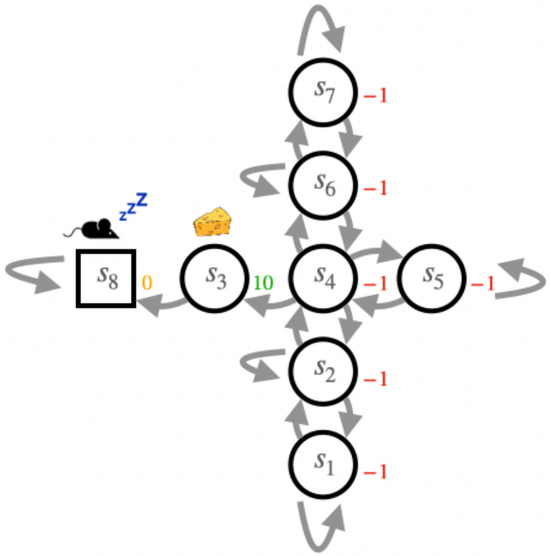
‘**Expected return**’ of a state, i.e. the **state-value function**: $$v(s) = \mathbb{E}[G_t \,|\, s]$$



Solving MDPs: Transform return into value function

Let's try to maximise returns $$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The value of states will depend on how an agent moves around!

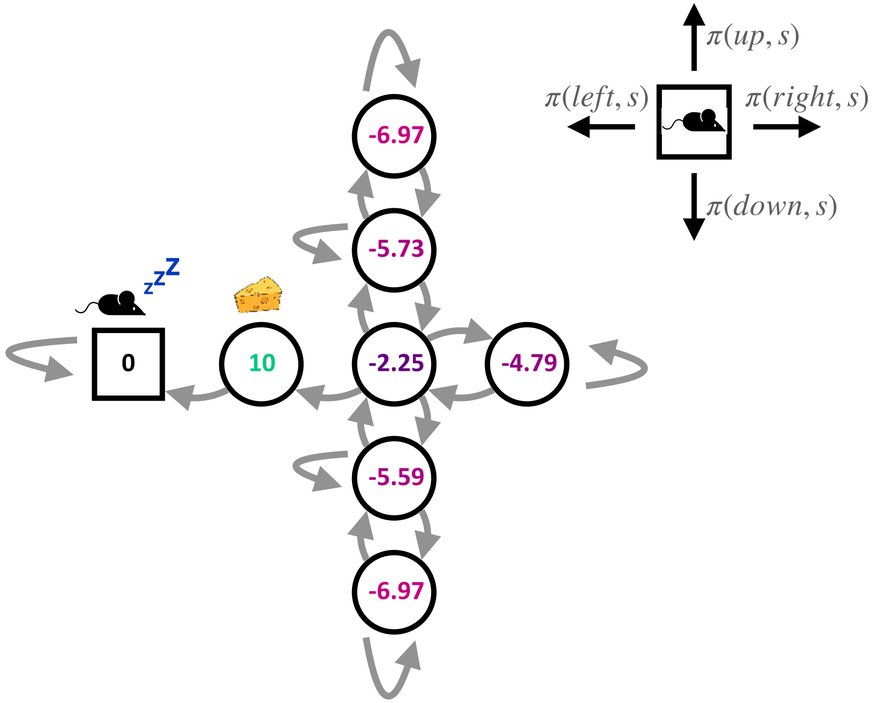


$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

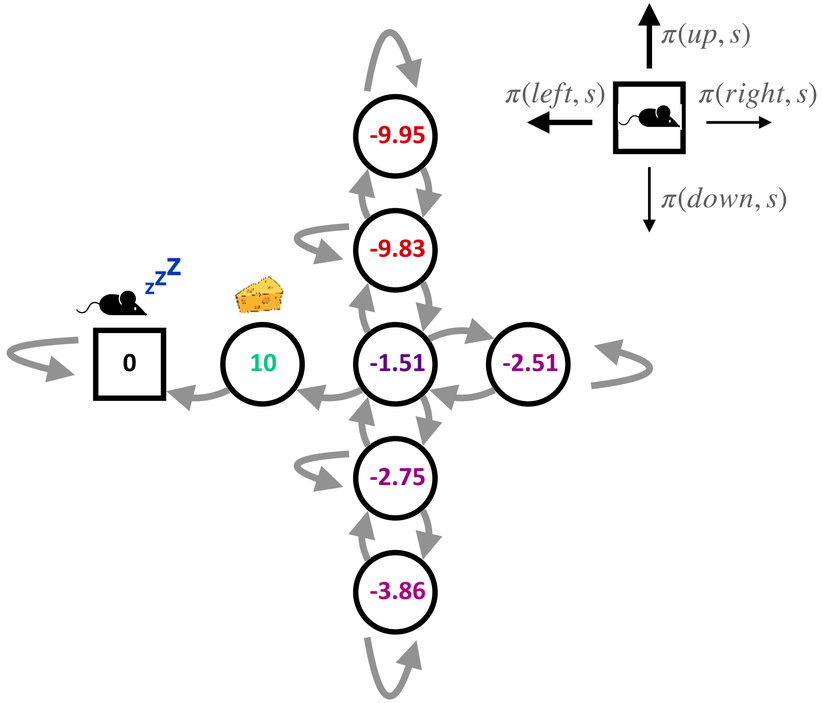
How good is a state under a given policy?

Solving MDPs: Transform return into value function

$v_\pi(s)$ for random policy ($\gamma = 0.9$)



$v_\pi(s)$ for bias to move up/left ($\gamma = 0.9$)



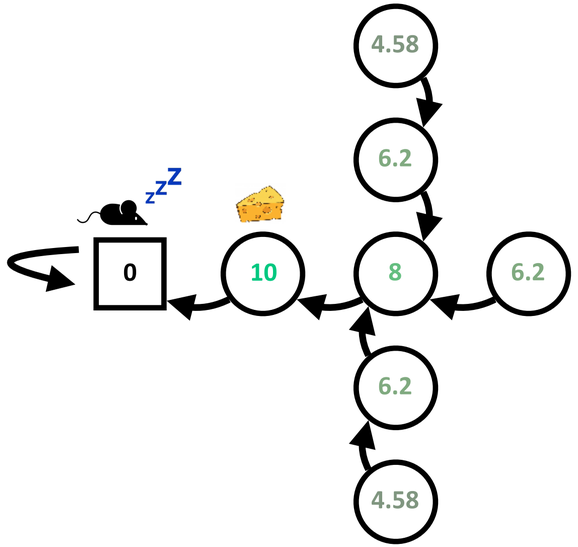
What's the point of this? We can use this approach to find the **best policy** that **maximises value**!

Solving MDPs optimally

We want the best policy, not just any policy - **optimal state value function**: $v_*(s) = max_\pi v_\pi(s)$

For every MDP, there is such an optimal policy
– but it's not trivial to find it

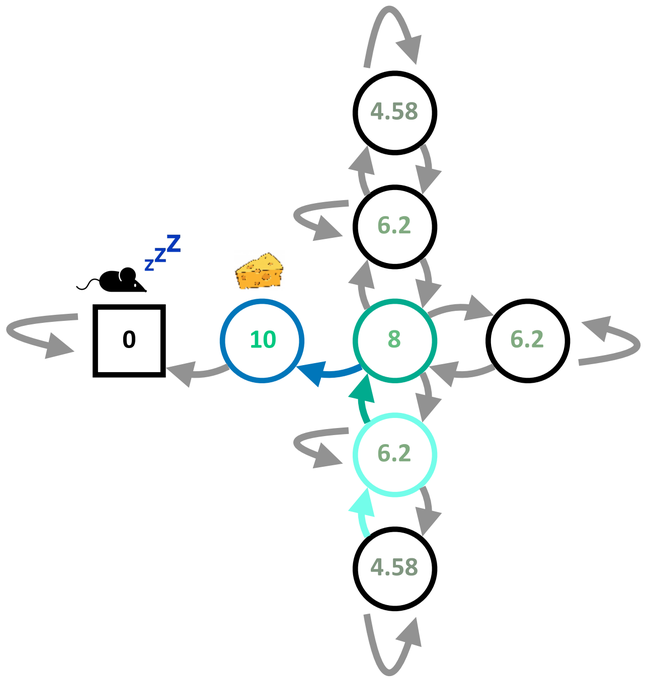
Obvious here – but computationally very
demanding in more realistic examples



Ways of solving MDPs optimally

The elegant way - **Bellman equation** (Bellman, 1957)

- We only need a 1-step look-ahead



Value of the current state based on…

$$v_\pi(s) = \sum \pi(a \mid s) \sum P(s', r \mid s, a)[r + \gamma v_\pi(s')])$$

Policy

World model

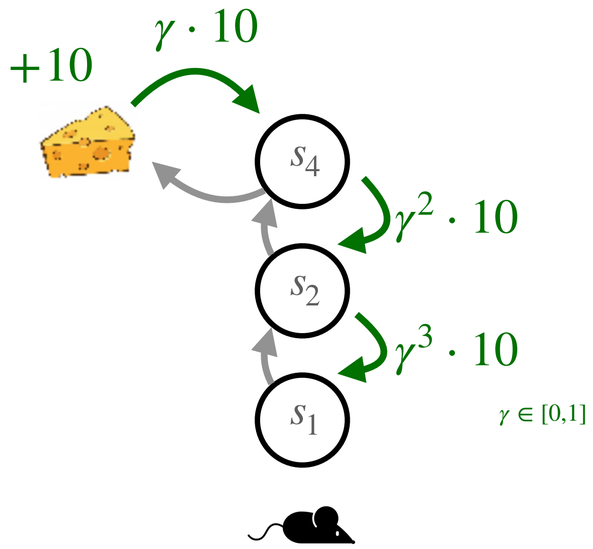
Reward and value of next state

Very good but computationally very demanding

Ways of solving MDPs optimally

The simple way - **TD learning**

- Just learn from experience



Value of the current state based on…

Prediction error

$$V(s,a) \leftarrow V(s,a) + \alpha \cdot (r + \gamma \cdot V(s',a) - V(s,a))$$

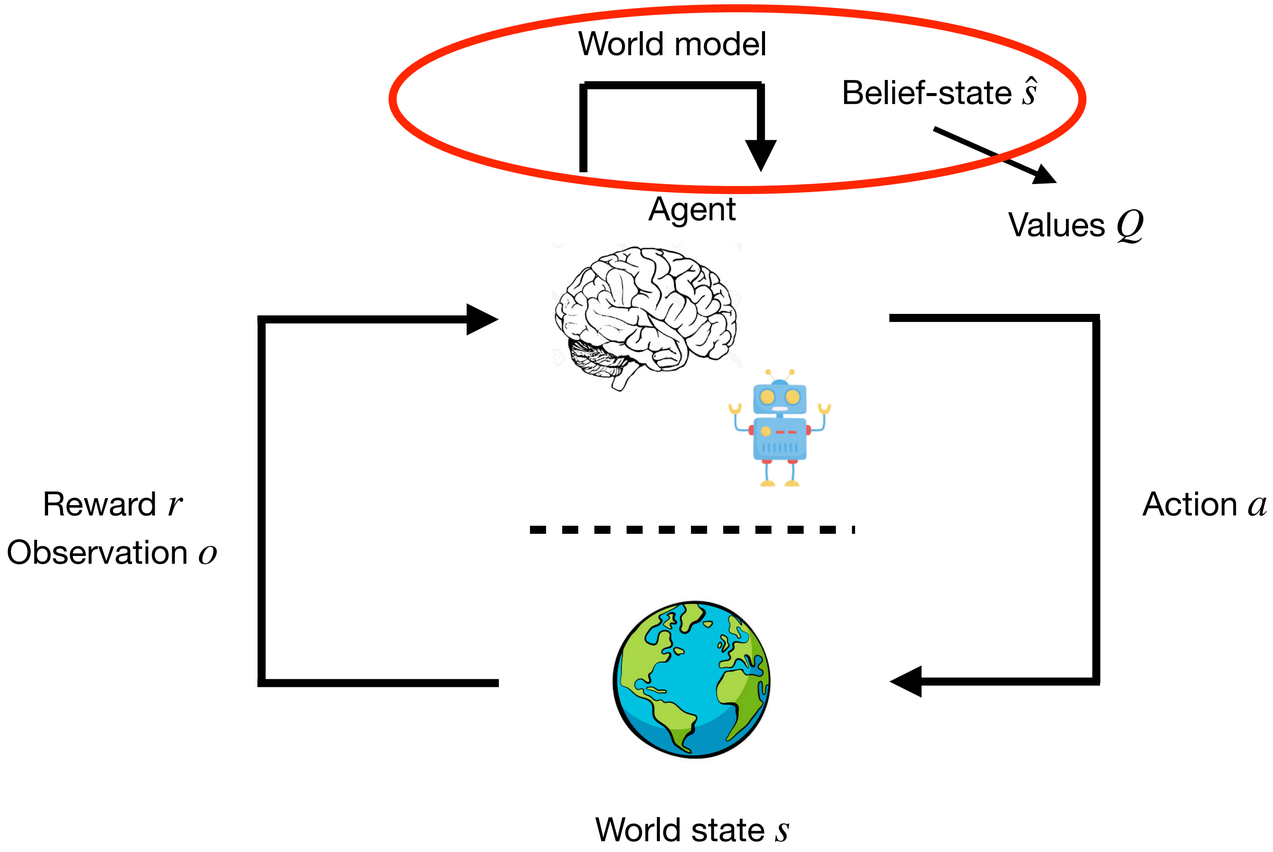
Learning rate

Discount rate

Simple and often efficient

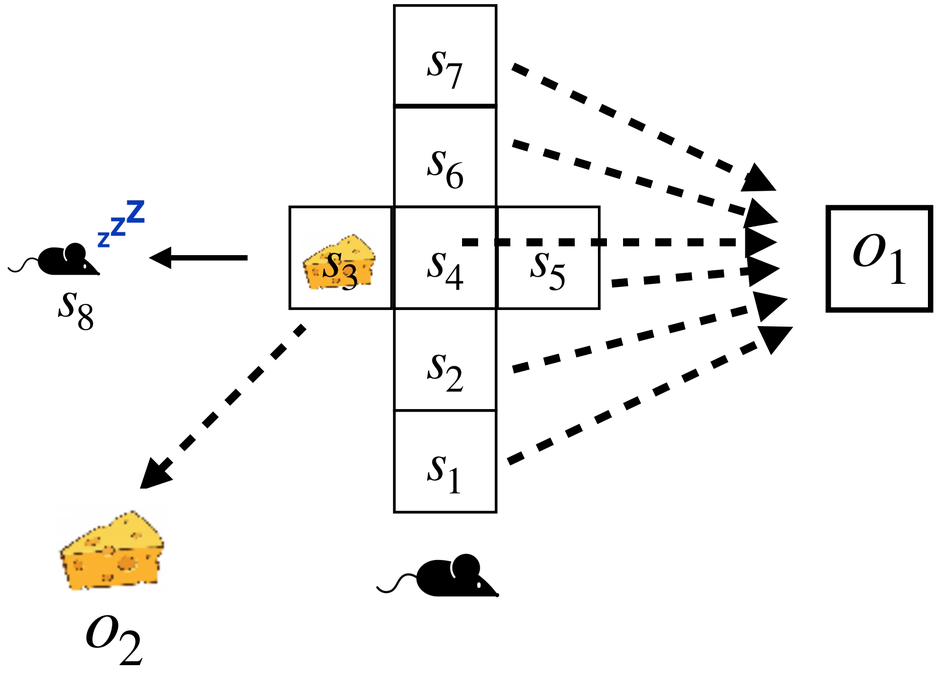
Extensions and Outlook

What happens if the world is not directly observable



Why?

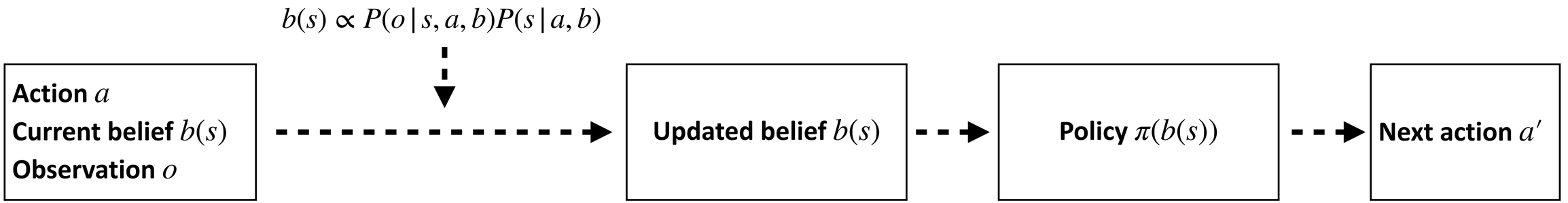
Often, states are not directly or *only partially* observable



Partially Observable Markov Decision Processes (POMDPs)

Need to incorporate inference on states in policy selection

- Relying on an **observation model** $P(o \mid s)$



Cassandra, Kaelbling & Littman, 1994

Think of a POMDP like an MDP with belief states

POMDPs in action

How did you decide whether to attend a talk at the CPC?

Attend a talk

Go for a swim with your friends

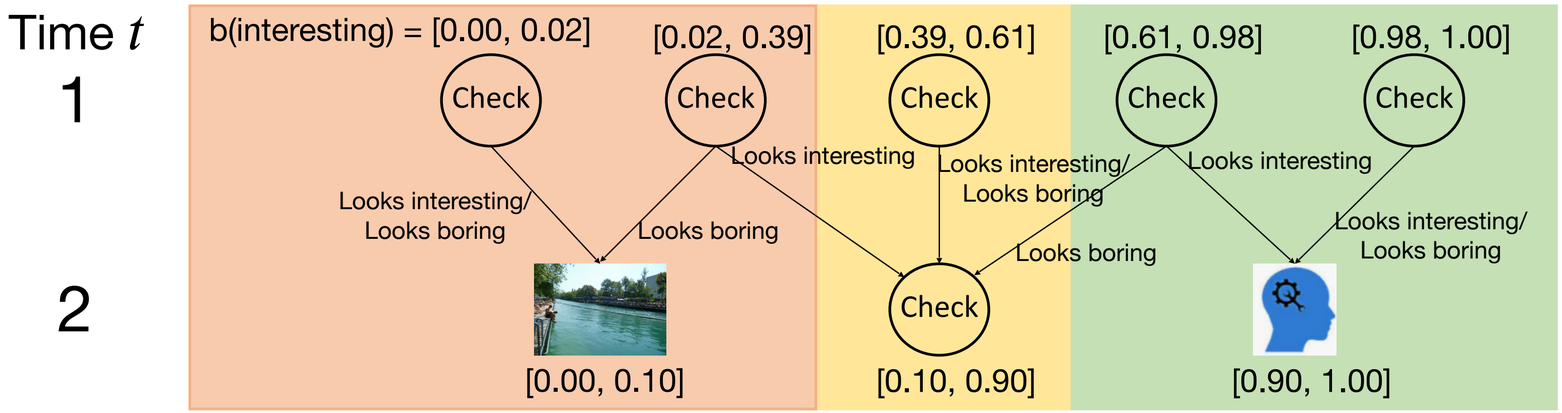
It’s very rewarding to attend a talk that is interesting, but more rewarding to go for a swim if the talk is less interesting

How do you decide?

- You could check the programme to reduce uncertainty about how interesting the talk will be
- Checking might be costly

[You should attend every talk at the CPC]

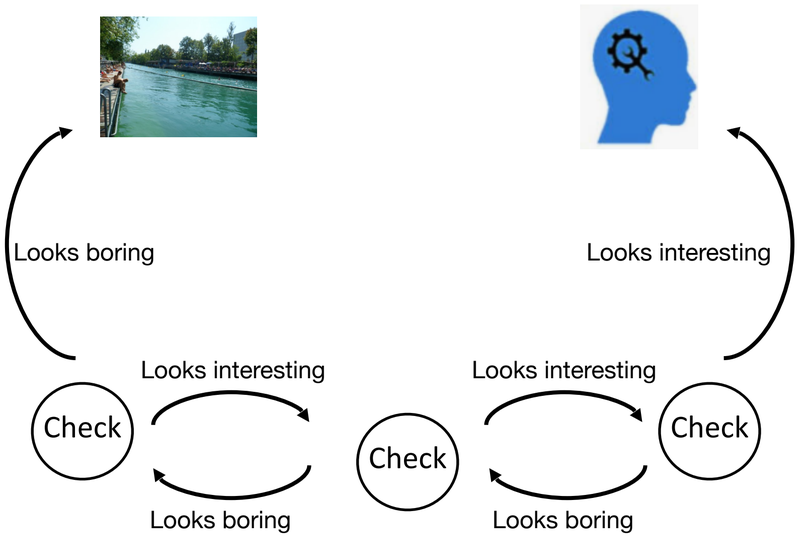
The value of information



Five different ‘policy trees’

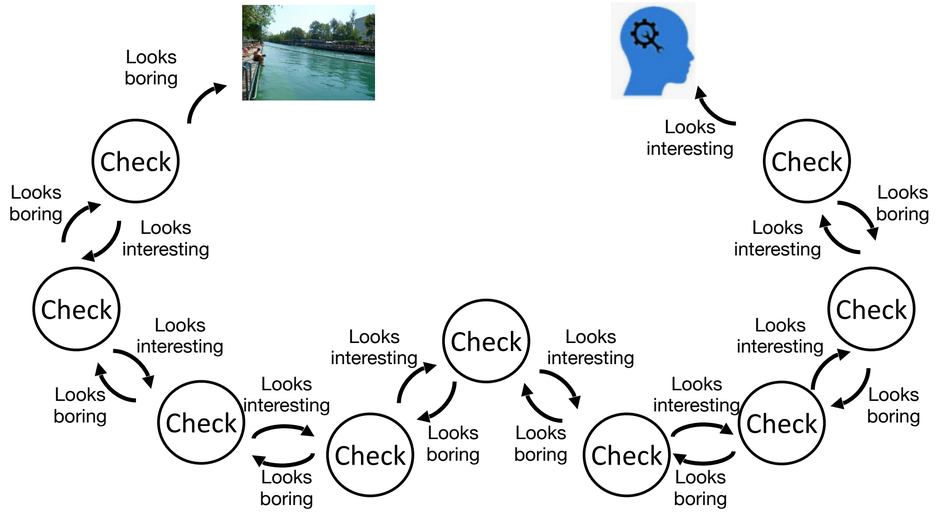
See Kaelbling, Littman & Cassandra, 1998

Optimal infinite-horizon policy drawn as a **plan graph**:



This strongly depends
on the parameterisation!

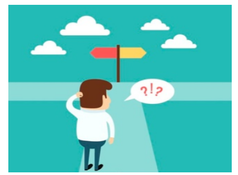
Plan graph for **reduced checking reliability**:



Why is this relevant for computational psychiatry?

Computational phenotyping – infer mechanisms that underlie suboptimal behaviour

- Suboptimal (learning of) **preferences/reward function**
- Failures of **inference on hidden states**
- Suboptimal **model configuration**
  - Action/state spaces
  - Wrong/noisy observation model, transition probabilities
- Failures of **goal-directedness/precision**
- Failures of **information gain**



Take home messages

Markov Decision Processes provide a rich framework for modelling choice behaviour

- **Markov property**
- Full specification of **dynamics** of the environment (model-based decision-making)

Various solutions of MDPs in the context of **maximising the expected return**

- **Bellman equation, TD learning**

POMDPs: Belief-based MDPs based on a generative model of the world

- Account for state uncertainty and the value of information

Highly relevant for computational psychiatry/neuroscience

- Allow to derive '**optimal actions**' and deviations thereof
- Investigate **where things can break**

Thank you

Additional slides

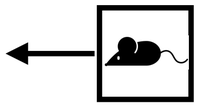
Basics: Markov Decision Processes (MDPs)

MDPs basis for model-based RL

$$P(s', r \mid s, a) = P(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$$

Can also encode **expected reward for state-action-next state triplet**
– which state-to-state transition is particularly good for a given action?

$$r(s, a, s') = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s'] = \sum_r r \frac{P(s', r \mid s, a)}{P(s' \mid s, a)}$$



What happens if I move left and how good will it be?

$$r(s, left, s') =$$

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	zZZ
s_1	−1	0	0	0	0	0	0	0
s_2	0	−1	0	0	0	0	0	0
s_3	0	0	0	0	0	0	0	0
s_4	0	0	10	0	0	0	0	0
s_5	0	0	0	−1	0	0	0	0
s_6	0	0	0	0	0	−1	0	0
s_7	0	0	0	0	0	0	−1	0
zZZ	0	0	0	0	0	0	0	0

So, how are MDPs useful in finding good actions?

Solving MDPs

Solving MDPs optimally

We can simplify the problem:

$$
\begin{aligned}
G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \\
&= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

Recursive definition of return

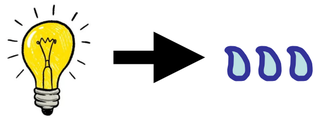
Recursive definition of value of state under given policy – **Bellman equation** (Bellman, 1957)

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t \mid s_t] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid s_t] \\
&= \sum \pi(a \mid s) \sum P(s', r \mid s, a)[r + \gamma v_\pi(s')])
\end{aligned}
$$

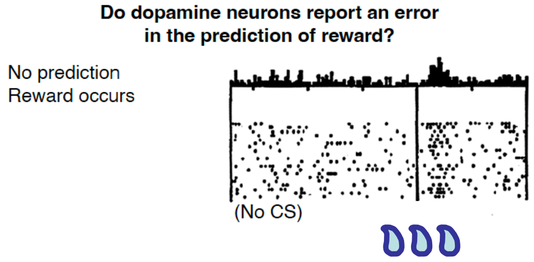
Key insight: you only need to **look one time step ahead** when computing the value!

TD Learning

There are simpler approaches that do not require a model of the world - most influential: **TD learning**

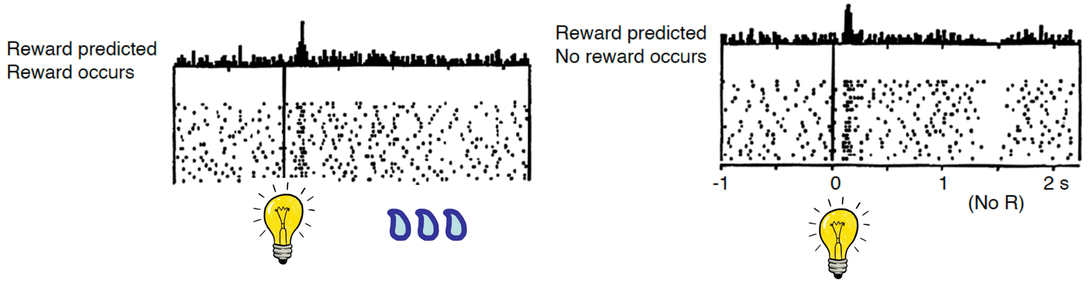


Dopamine neurons signal immediate reward



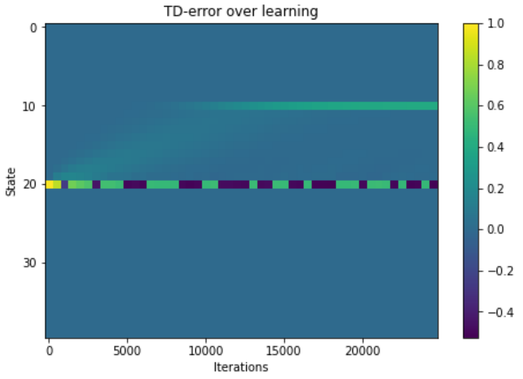
BUT: after training…

- DA signal reward prediction



Prediction error

$$V(s_t) \leftarrow V(s_t) + \alpha \cdot (r + \gamma \cdot V(s_{t+1}) - V(s_t))$$

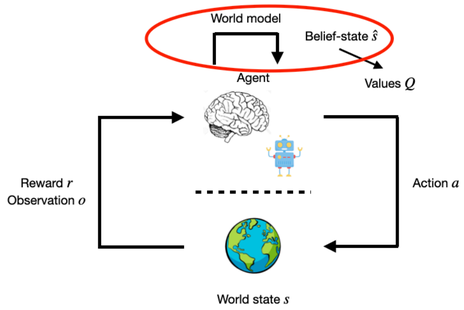


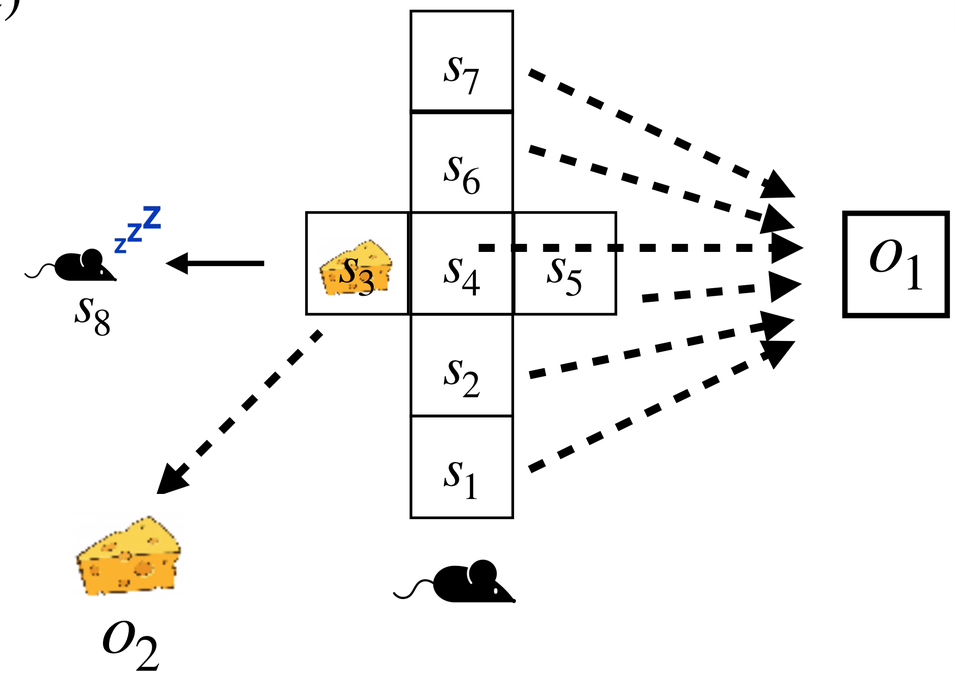
Extensions and Outlook

Partially Observable Markov Decision Process

A **Partially Abservable Markov Decision Process** is defined based on

- A (finite) state space S
- Transition probabilities $P(s_{t+1} = s' \mid s_t = s, a_t = a)$
- A Reward Function $R_s = \mathbb{E}[r_t \mid s_t = s, a_t = a]$
- A Discount Factor $\gamma \in [0,1]$
- A (finite) action space A
- Finite set of **observations** O
- **Observation model** $P(o \mid s)$





Environment dynamics specify a **Hidden Markov Model** (Markov Process + observation model)

The value of information

aka should you watch a talk at CPC or go for a swim with your friends? - Let's solve this with a POMDP:

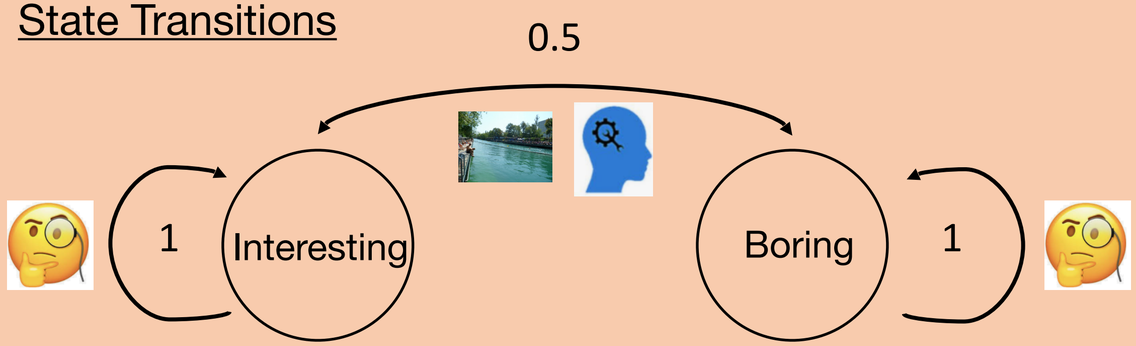
Rewards

		State	
		Talk interesting	Talk boring
Action	Attend talk	+10	-100
	Go swimming	-100	+10
	Check	-1	-1

'Checking validity'

		State	
		Talk interesting	Talk boring
Observation	This looks interesting	0.85	0.15
	This looks boring	0.15	0.85

State Transitions



This is actually the '*Tiger problem*' (Kaelbling, Littman & Cassandra, 1998)

See also Rigoux CPC slides 2015: https://www.tnu.ethz.ch/de/teaching/past-semesters/hs2016/cpcourse2016

Further Reading

MDP and RL:

- Sutton & Barto: Reinforcement Learning. MIT Press, Cambridge, MA, 1998/2018 (esp. chapter II)
- RL lecture David Silver at UCL: https://www.youtube.com/watch?v=lfHX2hHRMVQ
- Neuromatch lectures on RL and optimal control: https://github.com/NeuromatchAcademy/course-content/tree/main/tutorials#w2d4---optimal-control
- RL course material: https://github.com/schwartenbeckph/RL-Course

POMDPs

- Kaelbling, Littman, Cassandra: Planning and acting in partially observable stochastic domains. Artificial Intelligence, 1998
- Cassandra, Kaelbling, Littman: Acting Optimally in Partially Observable Stochastic Domains. AAAI-94 Proceedings, 1994
- Ghavamzadeh, Mannor, Pineau, Tamar: Bayesian Reinforcement Learning: A Survey. arXiv, 2016