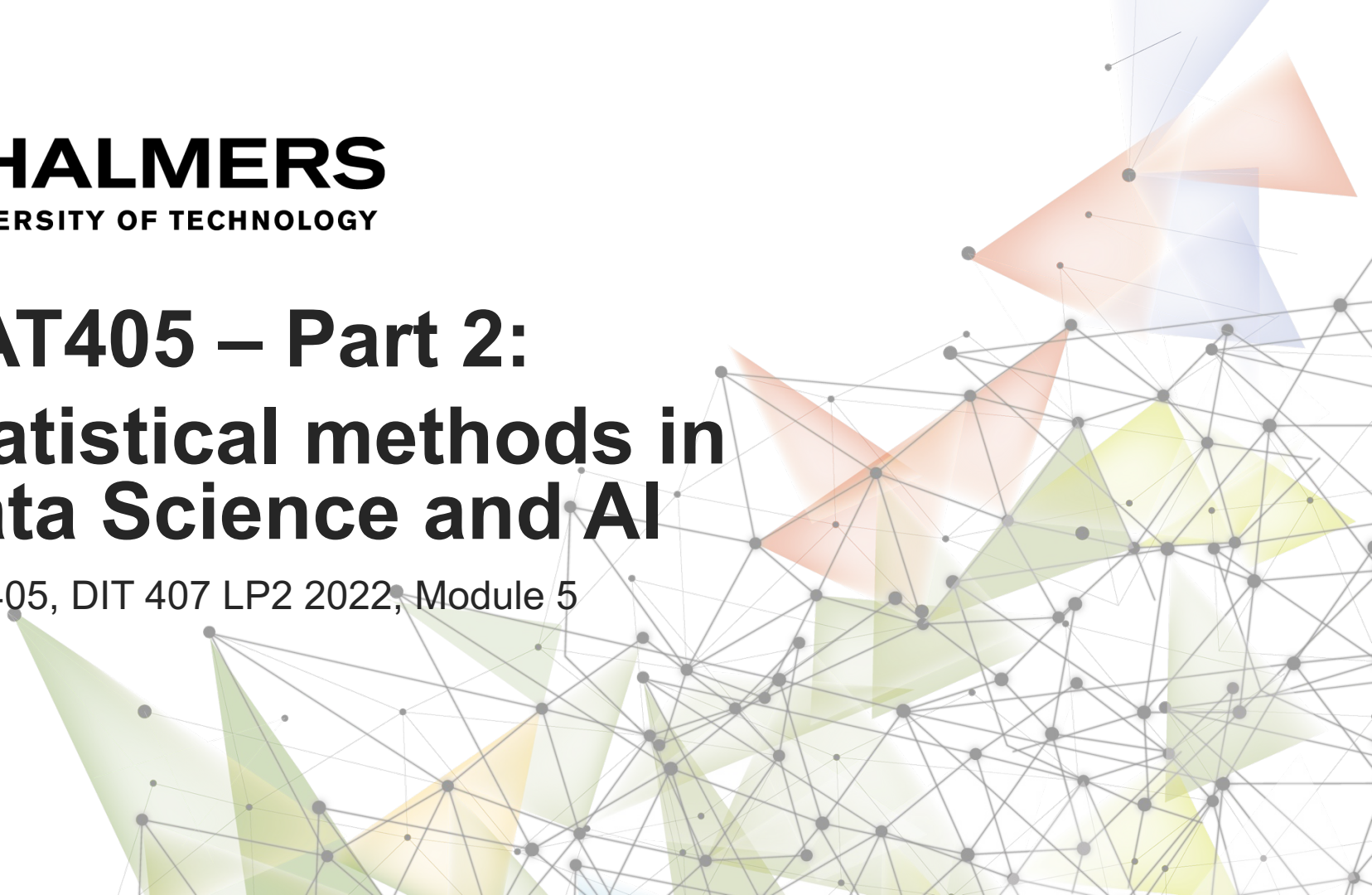
CHALMERS
UNIVERSITY OF TECHNOLOGY

DAT405 – Part 2:
Statistical methods in Data Science and AI

DAT405, DIT 407 LP2 2022, Module 5



CHALMERS
UNIVERSITY OF TECHNOLOGY

Module 5.1: Markov models

Today's topics:

- Markov models
- Hidden Markov models
- Markov Decision Processes
- Reinforcement learning in biology
- Reinforcement learning in computer science
- Exploiting and exploring
- Q learning

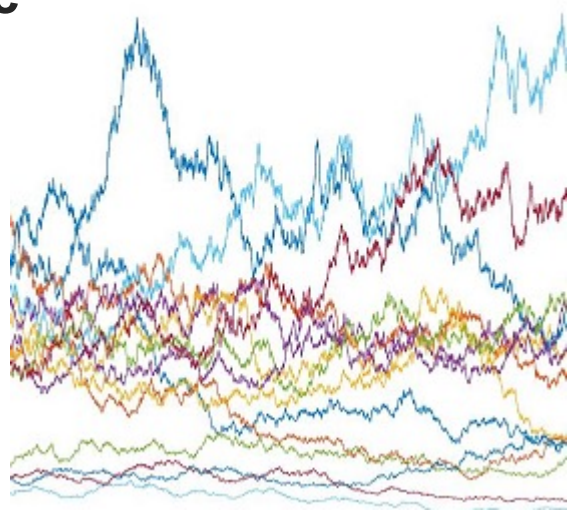
Random processes

- A ***random process*** is a sequence of random variables jumping between **states** at specific time points
- For a sequence of random variables

$$X_1, X_2, X_3, \ldots$$

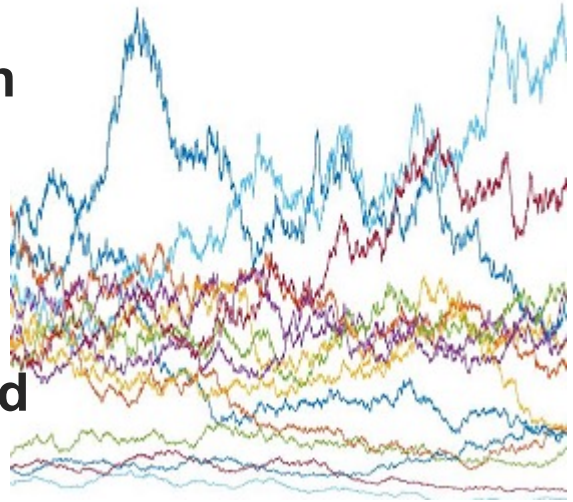
the distribution of the ***next*** variable is given by the ***transition probabilities***

$$\mathbb{P}(X_{t+1} | X_1, \ldots, X_t)$$



Random processes – examples

- Poisson processes: dealing with queues and waiting times
- Random walk and Brownian motion: used in stock trading
- Markov processes: used in a wide variety of classification applications
- Gaussian processes: used in regression and optimization
- Time series: used to model seasonal variations



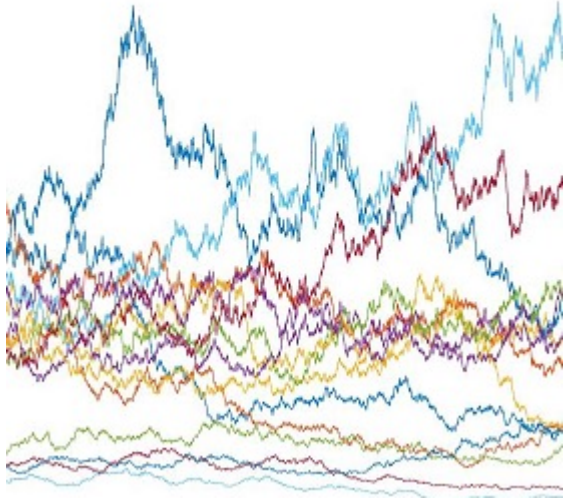
Markov processes

- **A Markov process is a random process where the next state only depends on the current state**

$$\mathbb{P}(X_{t+1}|X_1, \dots, X_t) = \mathbb{P}(X_{t+1}|X_t)$$

I.e. X_{t+1} only depends on X_t and is independent of $X_1, \dots, X_{t-1}$

Recall that we only need to keep the variables that X_{t+1} is dependent on.

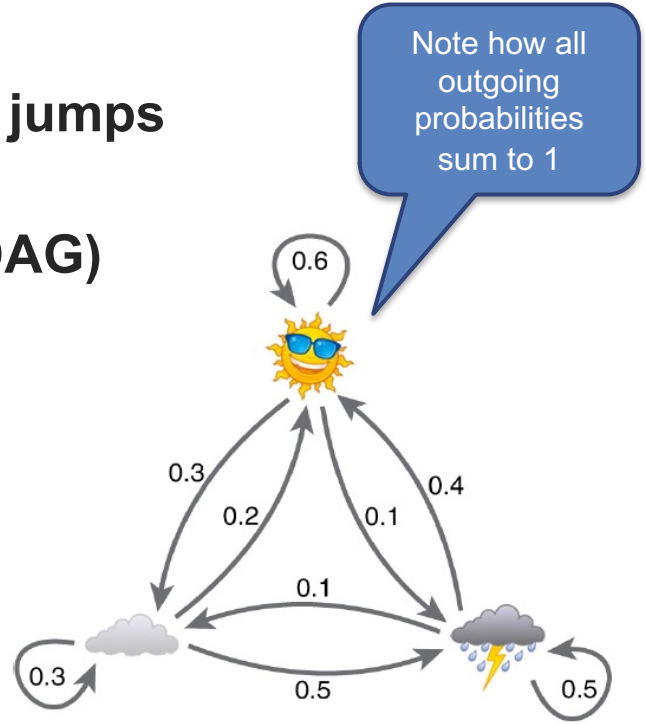


Markov chains

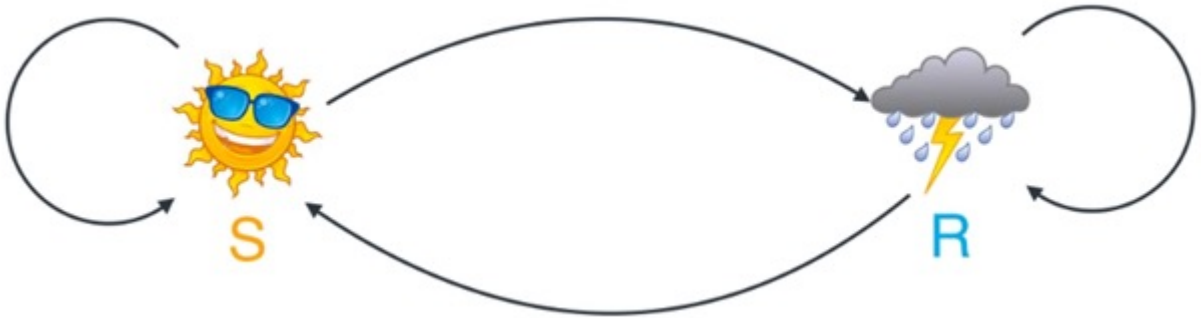
- A **Markov chain** is a Markov process that jumps between states at **discrete times**
- Represented as a finite directed graph (DAG)
  - Nodes = states
  - Edges = possible transitions
- The independence assumption

$$\mathbb{P}(\text{next state}|\text{all previous}) = \\ = \mathbb{P}(\text{next state}|\text{current state})$$

is called the ***Markov property***



Example: Markov chain



Sample space: S = sunny, R = rain

Example: Markov chain

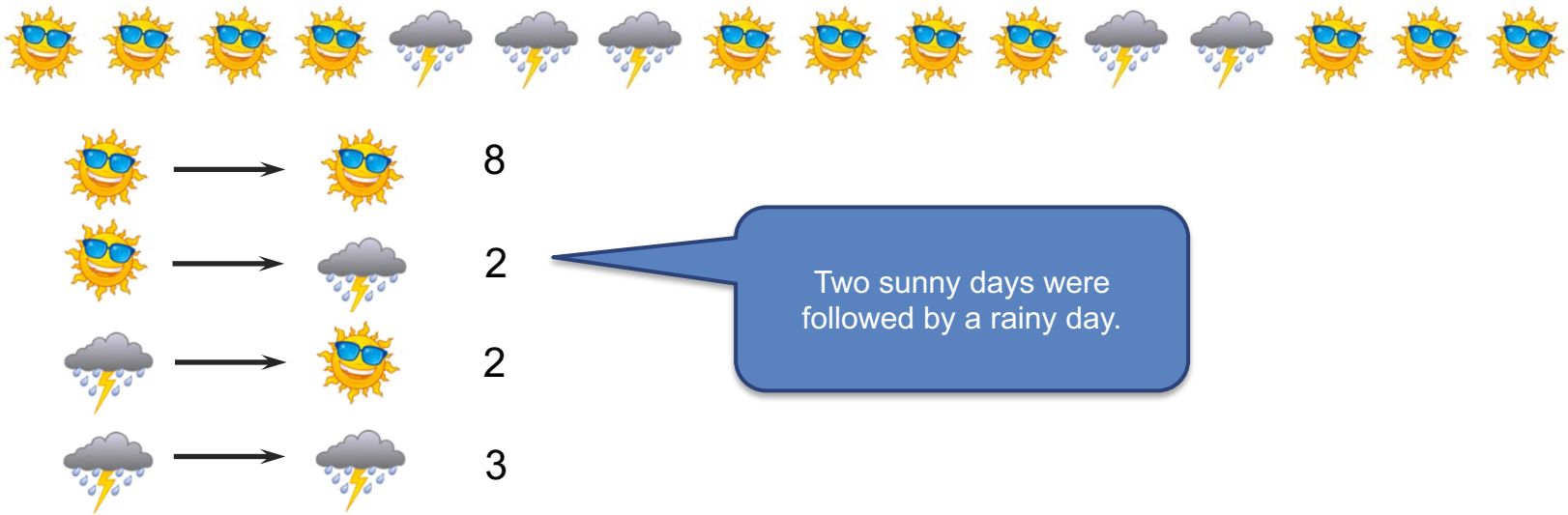
Starting with data:

- **Suppose we have observed the weather for 15 days**

Can we construct a Markov model from data? I.e. estimate the parameters?

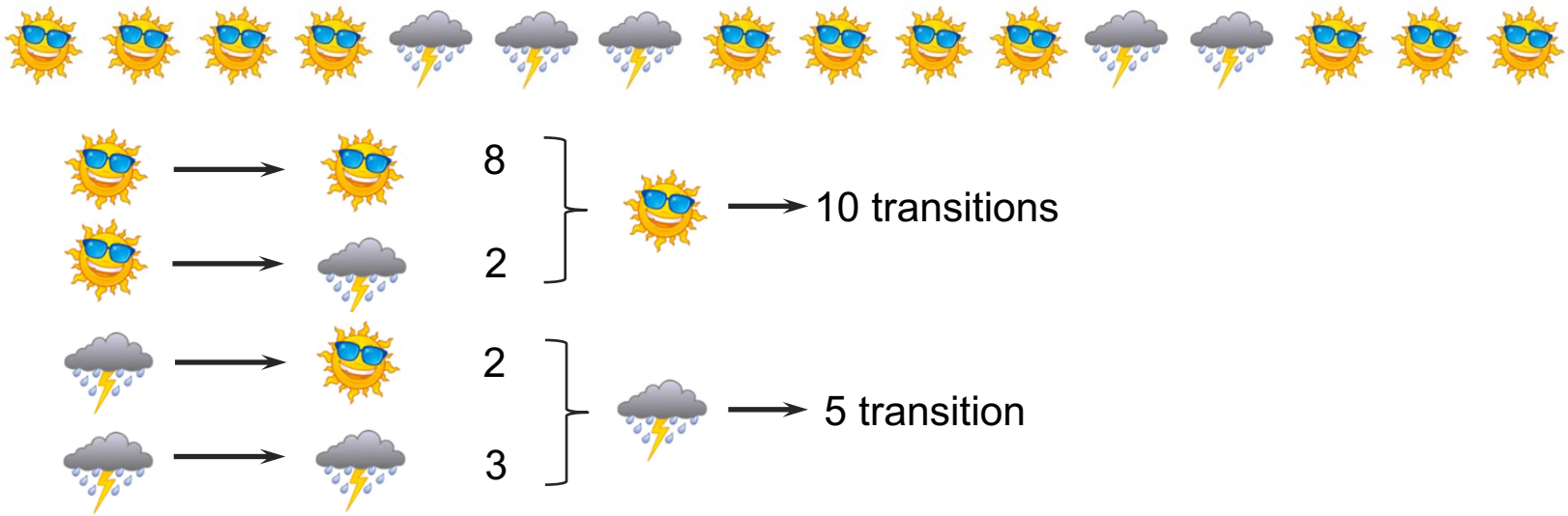
Example: Markov chain

We can estimate the transition probabilities by counting the occurrences of each transition in the data



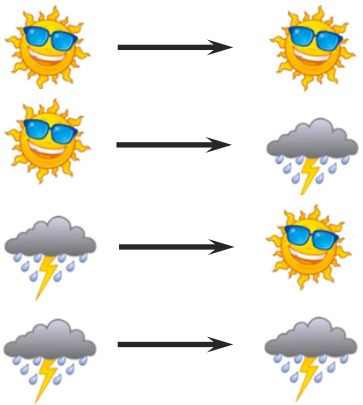
Example: Markov chain

We can estimate the transition probabilities by counting the occurrences of each transition in the data



Example: Markov chain

We can estimate the transition probabilities by counting the occurrences of each transition in the data



8/10

2/10

2/5

3/5

$$\mathbb{P}(\text{☀}|\text{☀}) = 0.8$$

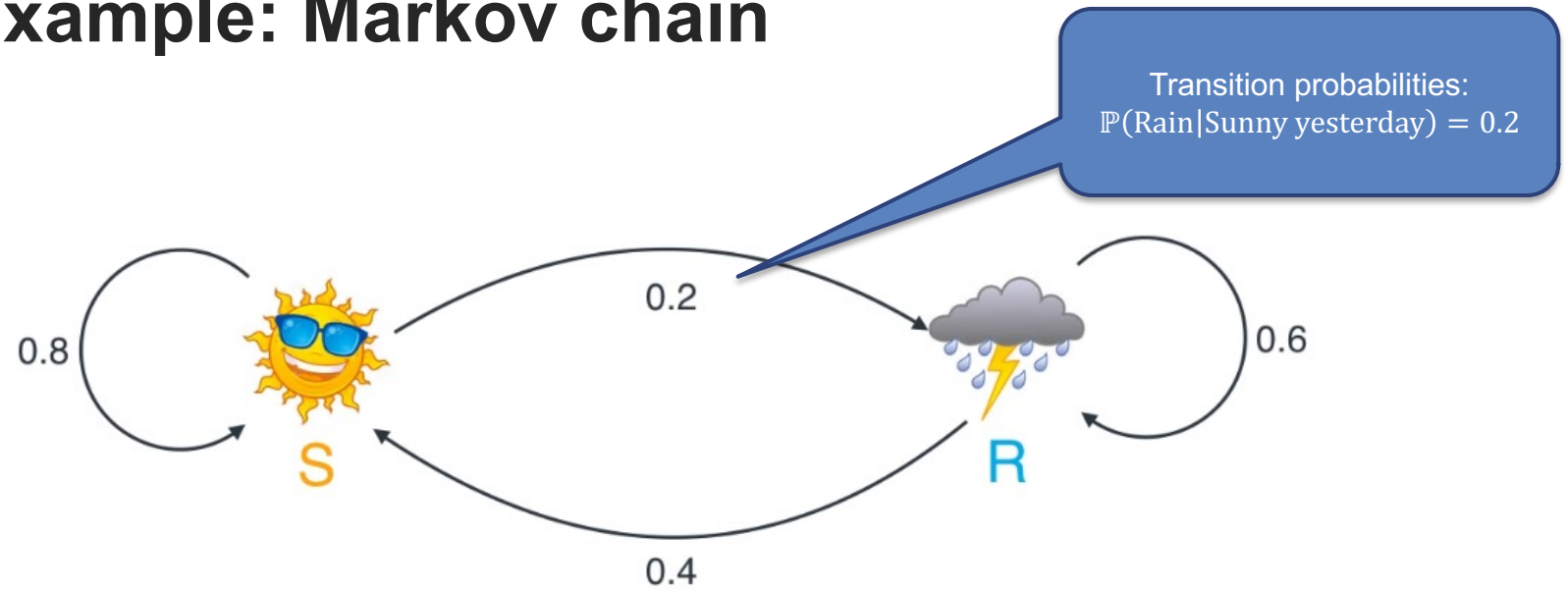
$$\mathbb{P}(\text{⛈}|\text{☀}) = 0.2$$

$$\mathbb{P}(\text{☀}|\text{⛈}) = 0.4$$

$$\mathbb{P}(\text{⛈}|\text{⛈}) = 0.8$$

Transiton probabilities

Example: Markov chain



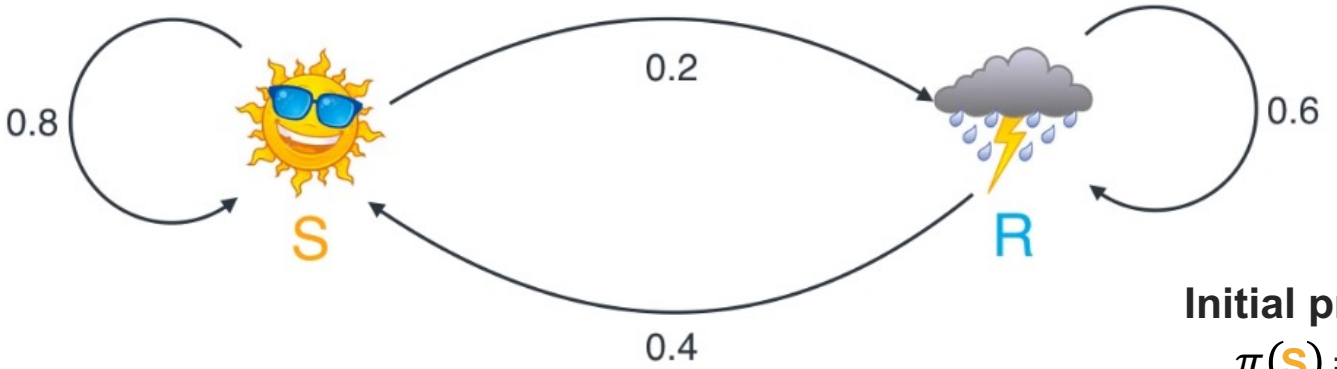
Example: Markov chain

But the probability of the first day has no transition probability.
For that we estimate the initial probabilities for each state.

		π_i
	10	10/15
	5	5/15
	15	1

- We count the number of occurrences of each state
- We can estimate the initial probabilities π_S and π_R

Example: Markov chain



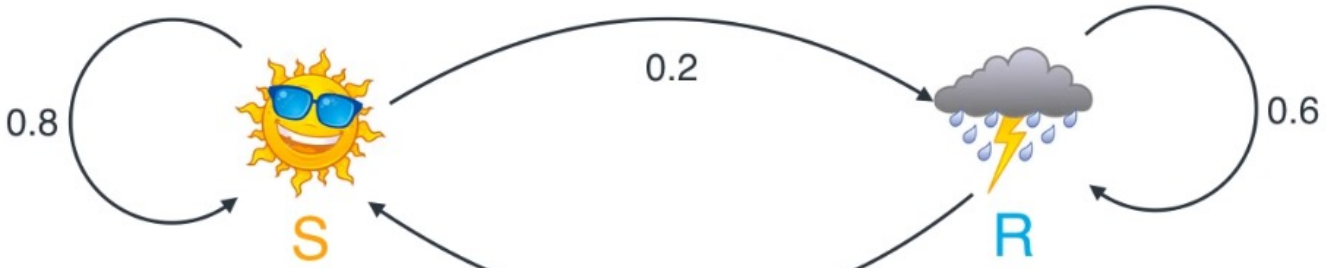
Initial probabilities

$\pi(S) = 2/3$

$\pi(R) = 1/3$

This is our complete Markov chain.

Example: Markov chain



The probability of sun tomorrow depends on the weather today

Initial probabilities

$\pi(\mathbf{S}) = 2/3$

$\pi(\mathbf{R}) = 1/3$

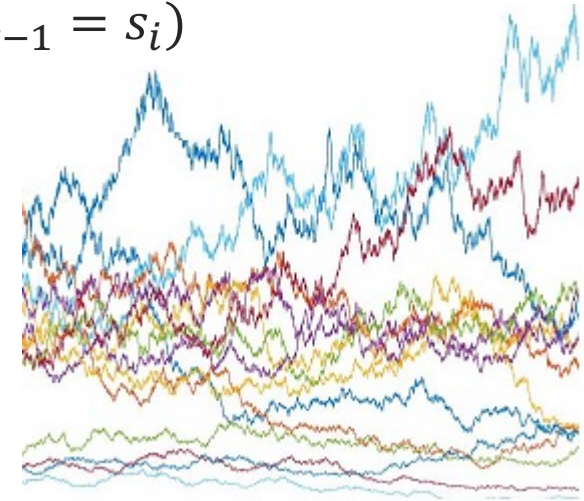
$$\mathbb{P}(\mathbf{S}\text{ tomorrow}) = 0.8 \cdot \mathbb{P}(\mathbf{S}\text{ today}) + 0.4 \cdot \mathbb{P}(\mathbf{R}\text{ today})$$

$$\mathbb{P}(\mathbf{R}\text{ tomorrow}) = 0.2 \cdot \mathbb{P}(\mathbf{S}\text{ today}) + 0.4 \cdot \mathbb{P}(\mathbf{R}\text{ today})$$

Markov chains – the model

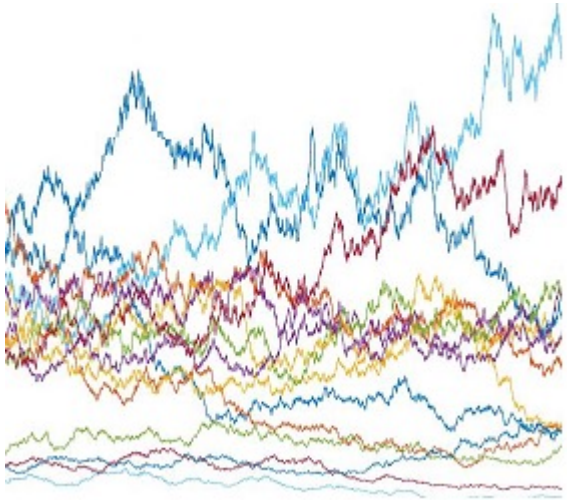
A random process $X_1, X_2, X_3, \ldots$

- **A state space:** $S = \{s_1, s_2, \ldots, s_N\}$
- **Transition probabilities:** $p_{ij} = \mathbb{P}(X_{t+1} = s_j | X_{t-1} = s_i)$
- **Initial distribution:** $\pi_i = \mathbb{P}(X_1 = s_i)$



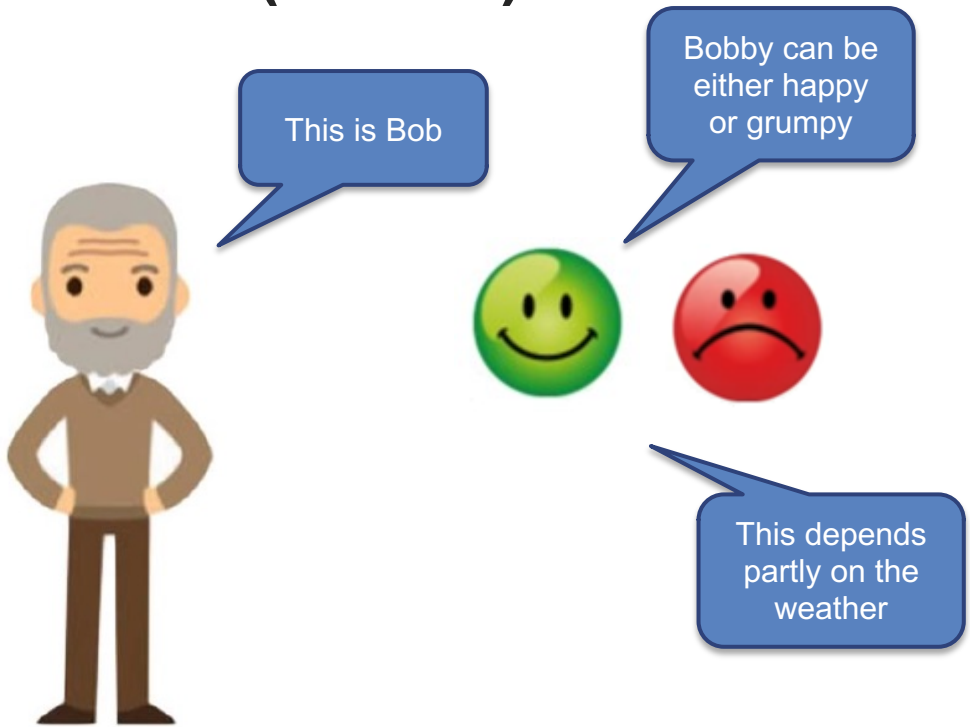
Hidden Markov models (HMMs)

Now the actual state of the process is *hidden*, and we only observe something that is *related* to the underlying state



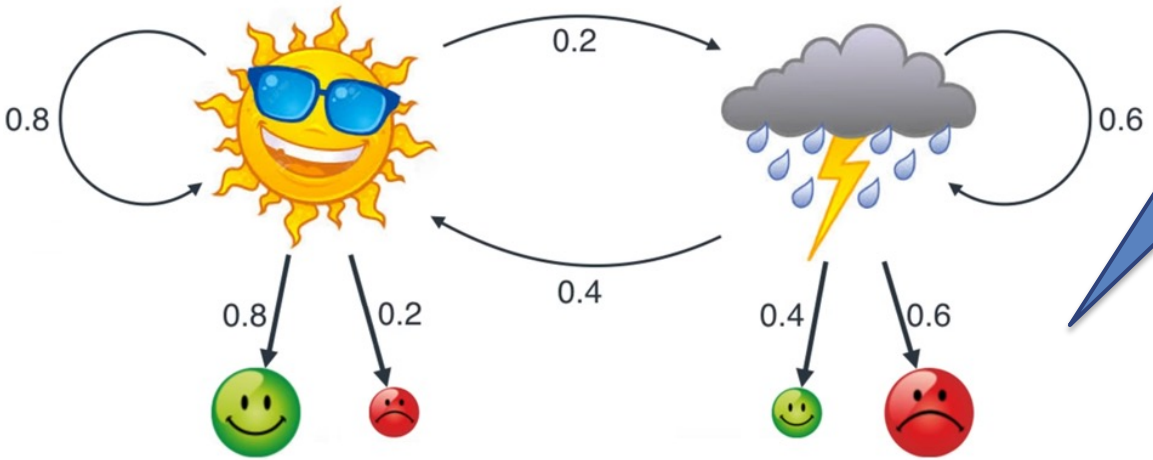
Hidden Markov models (HMMs)

- **Introducing Bob**



Hidden Markov models (HMMs)

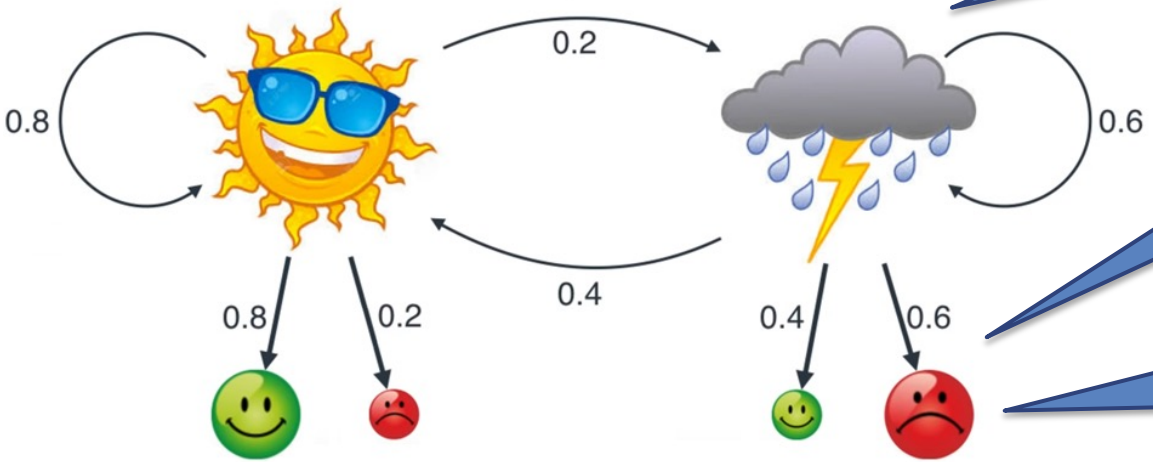
- **Bob's mood depends partly on the weather**



This is an HMM: it represents the weather and how Bob's mood depends on it

Hidden Markov models (HMMs)

- **Now assume we don't know the weather, but only observe Bob's mood**

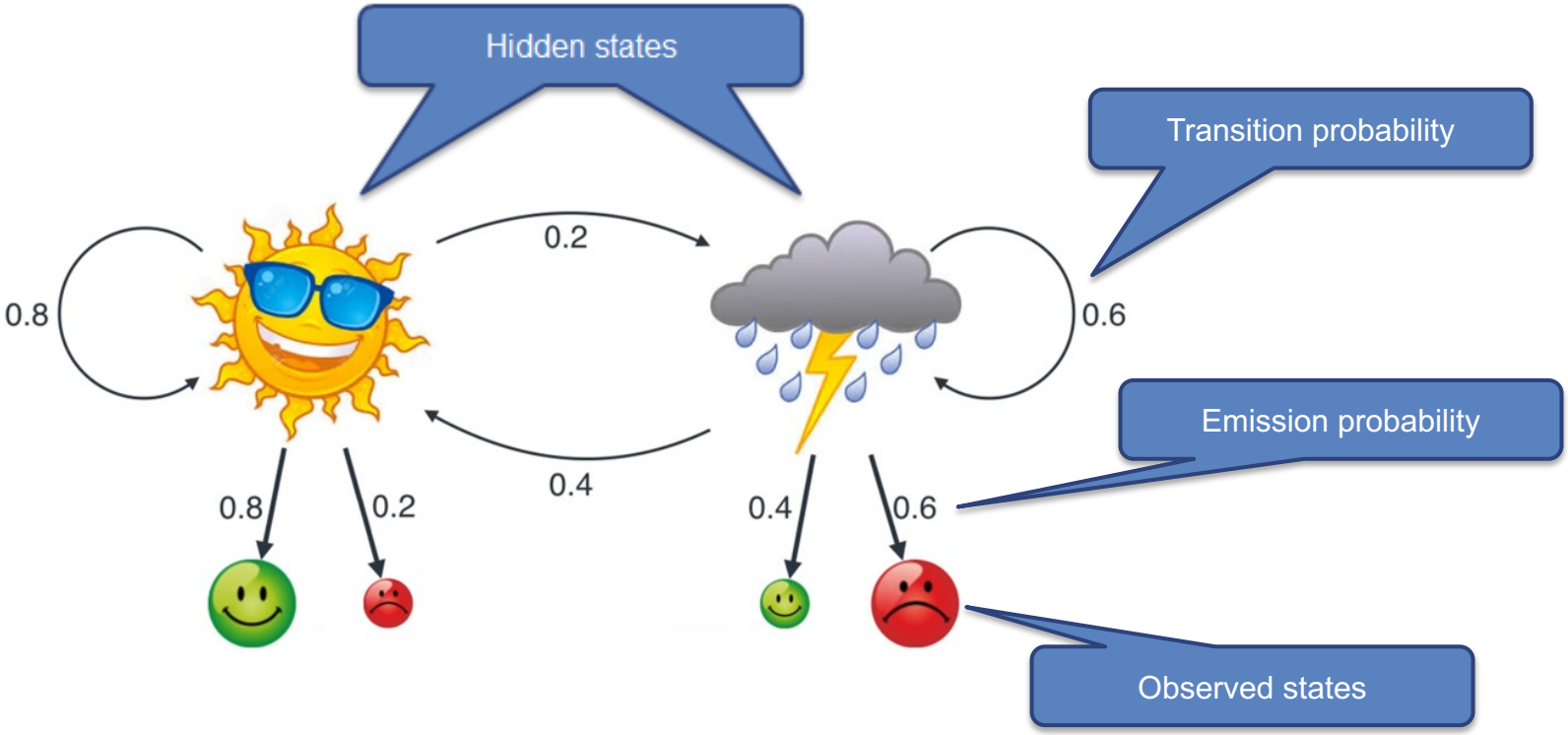


The weather is a hidden process, with states 'sunny' or 'rainy'

Bob's mood is the observed process

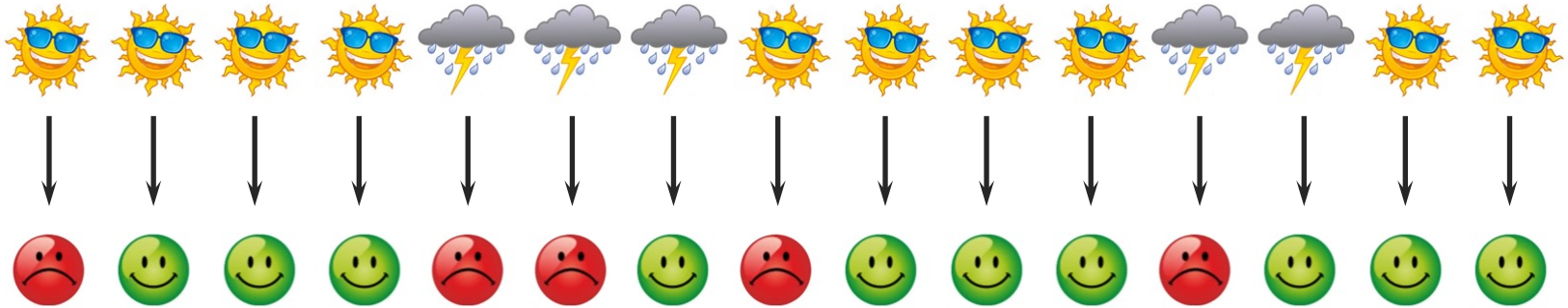
Given the model and the observed process, we can guess the most likely weather

HMMs terminology



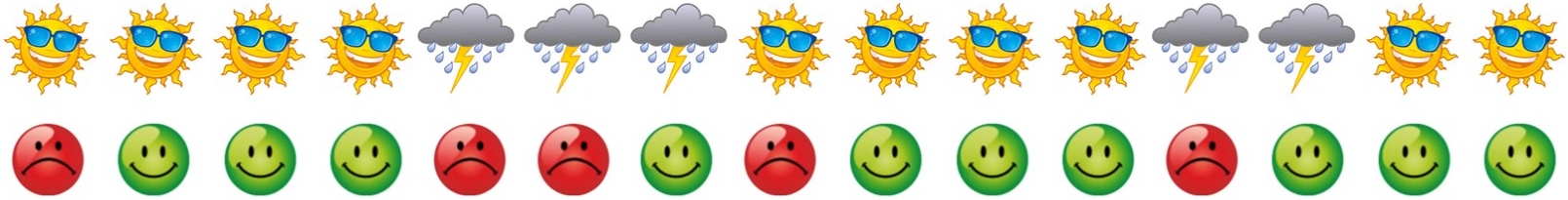
Hidden Markov models (HMMs)

- Now assume that our data include Bob's moods



Hidden Markov models (HMMs)

- **We can estimate the emission probabilities from data**



How many sunny days are Bob happy?

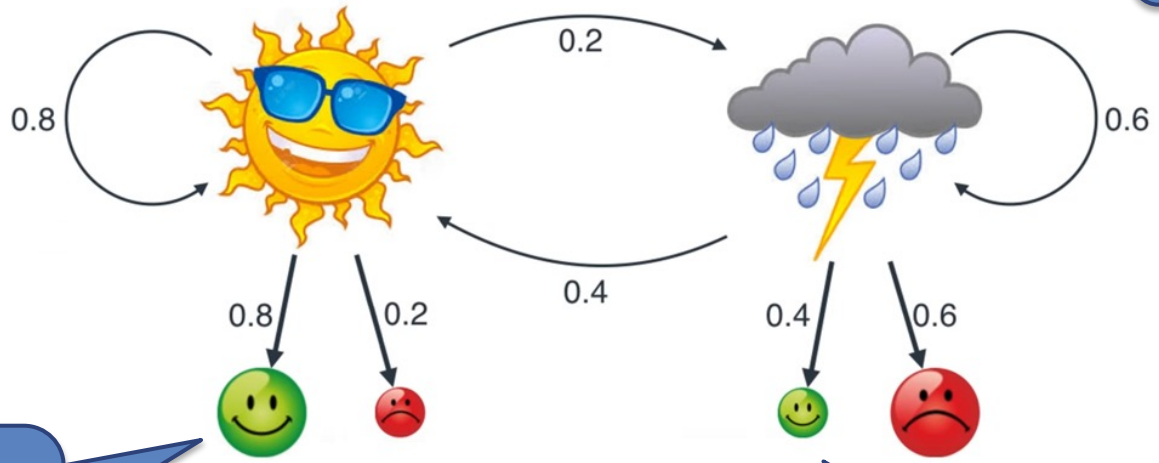
☀	→	8	0.8
☀	→	2	0.2

How many rainy days are Bob happy?

🌧	→	2	0.4
🌧	→	3	0.6

Hidden Markov models (HMMs)

Now we can answer several questions by analyzing the HMM.

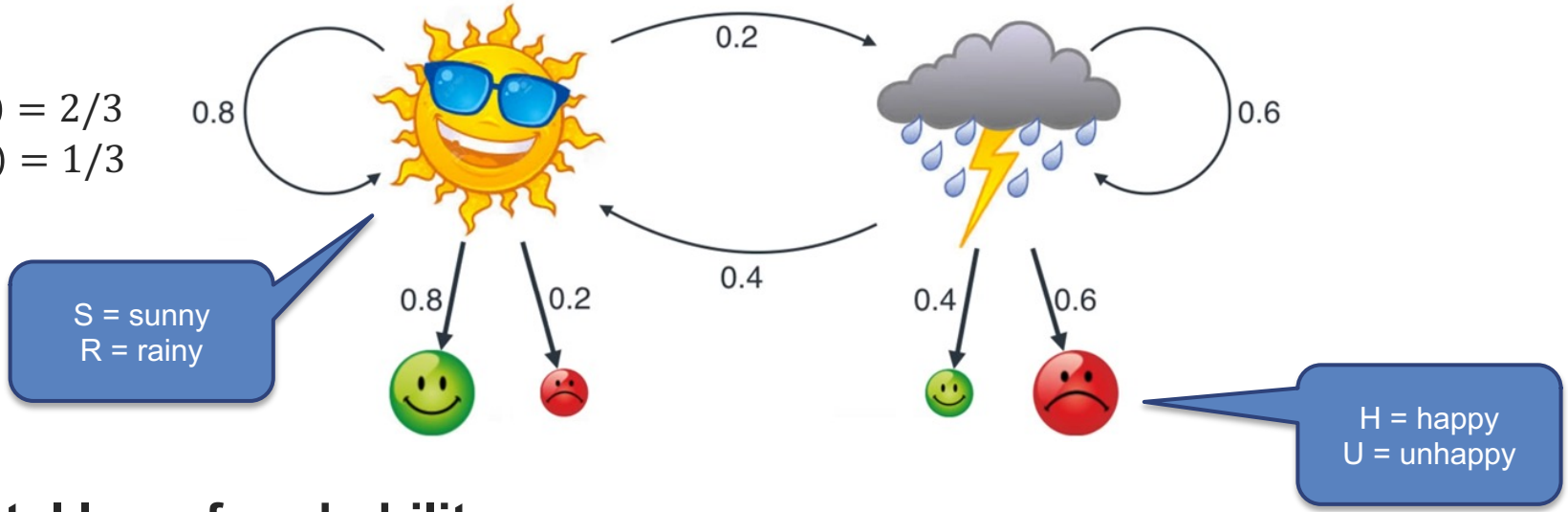


Given that Bob is happy, what is the probability of the weather being sunny?

What is the probability that Bob is happy?

Probability of Bob being happy (H)?

$\mathbb{P}(\mathbf{S}) = 2/3$
$\mathbb{P}(\mathbf{R}) = 1/3$

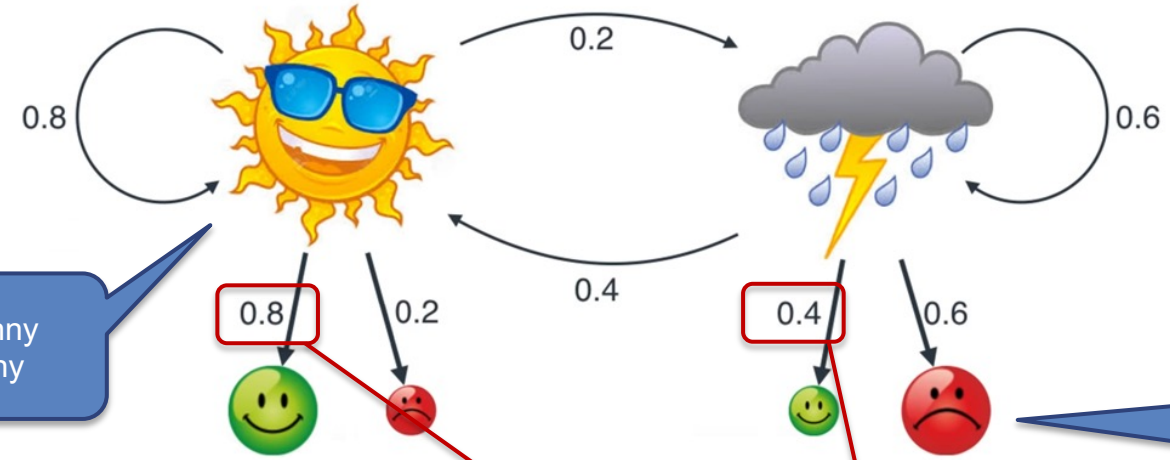


Total law of probability:

$$\mathbb{P}(\text{☺}) = \mathbb{P}(\text{☺}|\mathbf{S})P(\mathbf{S}) + \mathbb{P}(\text{☺}|\mathbf{R})P(\mathbf{R})$$

Probability of Bob being happy (H)?

$\mathbb{P}(\mathbf{S}) = 2/3$
$\mathbb{P}(\mathbf{R}) = 1/3$



S = sunny
R = rainy

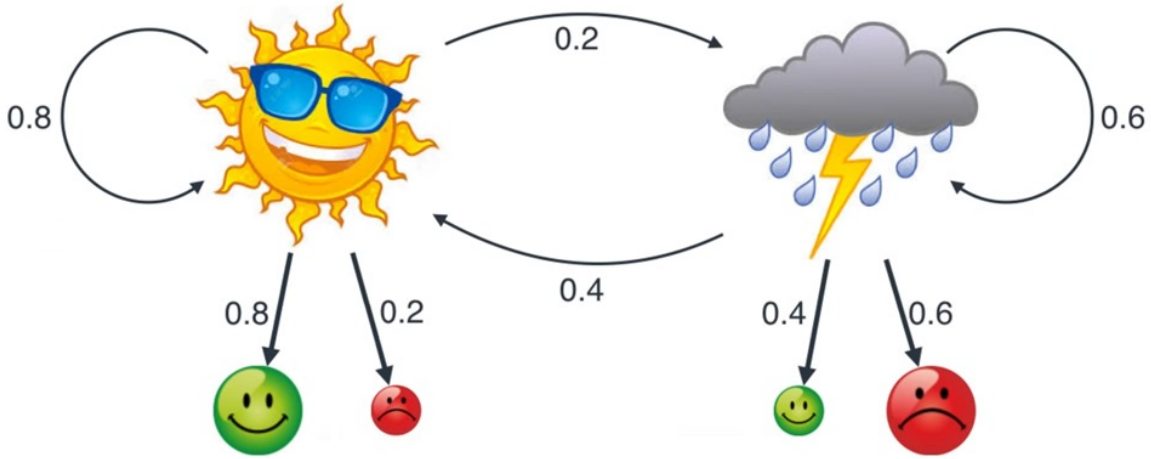
H = happy
U = unhappy

Total law of probability:

$$\mathbb{P}(\text{☺}) = \mathbb{P}(\text{☺}|\mathbf{S})\mathbb{P}(\mathbf{S}) + \mathbb{P}(\text{☺}|\mathbf{R})\mathbb{P}(\mathbf{R}) = 0.8 \cdot \frac{2}{3} + 0.4 \cdot \frac{1}{3} = \frac{2}{3}$$

Prior probabilities of Bob's mood

$\mathbb{P}(\mathbf{S}) = 2/3$
$\mathbb{P}(\mathbf{R}) = 1/3$



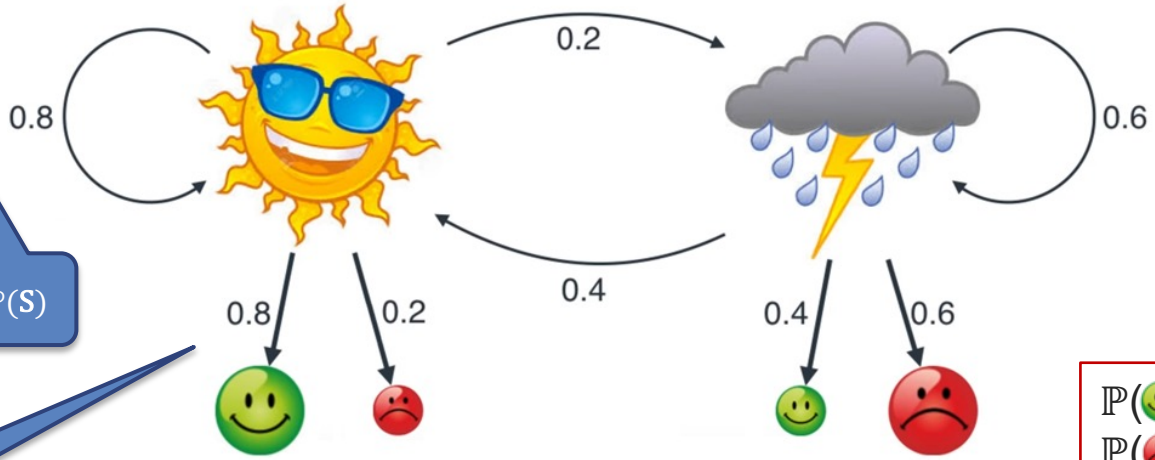
Prior distribution

$\mathbb{P}(\text{☺}) = 2/3$
$\mathbb{P}(\text{☹}) = 1/3$

Probability of sunshine if Bob is happy?

$\mathbb{P}(\mathbf{S}) = 2/3$
$\mathbb{P}(\mathbf{R}) = 1/3$

We have $\mathbb{P}(\mathbf{S})$



And we just computed $\mathbb{P}(\mathbf{H})$

$\mathbb{P}(\text{☺}) = 2/3$
$\mathbb{P}(\text{☹}) = 1/3$

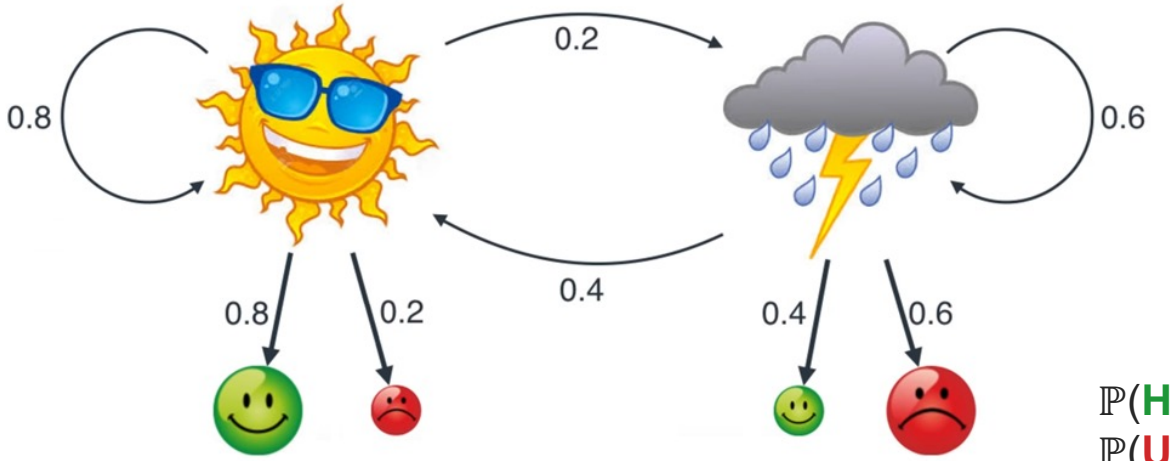
And here is $\mathbb{P}(\mathbf{H}|\mathbf{S})$

So we can use Baye's rule for this

$\mathbb{P}(\mathbf{S}|\mathbf{H}) = ?$

Probability of sunshine if Bob is happy?

$\mathbb{P}(\mathbf{S}) = 2/3$
$\mathbb{P}(\mathbf{R}) = 1/3$



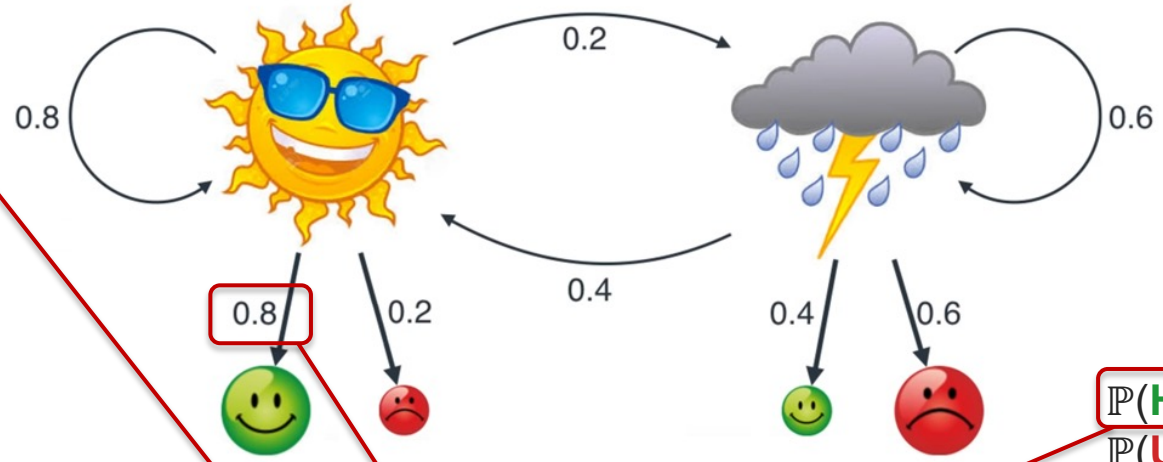
$\mathbb{P}(\mathbf{H}) = 2/3$
$\mathbb{P}(\mathbf{U}) = 1/3$

Bayes' rule:

$$\mathbb{P}(\mathbf{S}|\mathbf{H}) = \frac{\mathbb{P}(\mathbf{H}|\mathbf{S})P(\mathbf{S})}{\mathbb{P}(\mathbf{H})}$$

Probability of sunshine if Bob is happy?

$\mathbb{P}(\mathbf{S}) = 2/3$
$\mathbb{P}(\mathbf{R}) = 1/3$



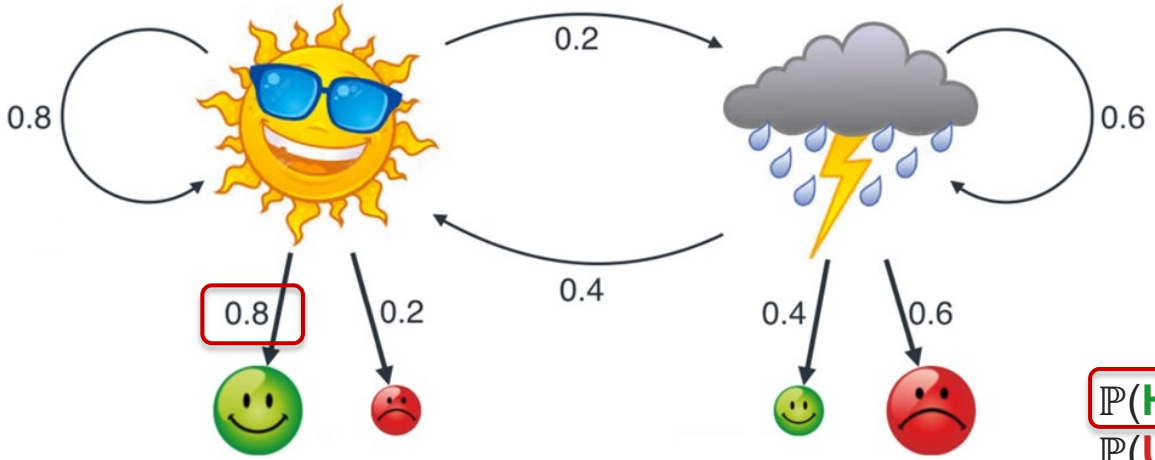
$\mathbb{P}(\mathbf{H}) = 2/3$
$\mathbb{P}(\mathbf{U}) = 1/3$

Bayes' rule:

$$\mathbb{P}(\mathbf{S}|\mathbf{H}) = \frac{\mathbb{P}(\mathbf{H}|\mathbf{S})P(\mathbf{S})}{\mathbb{P}(\mathbf{H})}$$

Probability of sunshine if Bob is happy?

$\mathbb{P}(\mathbf{S}) = 2/3$
$\mathbb{P}(\mathbf{R}) = 1/3$

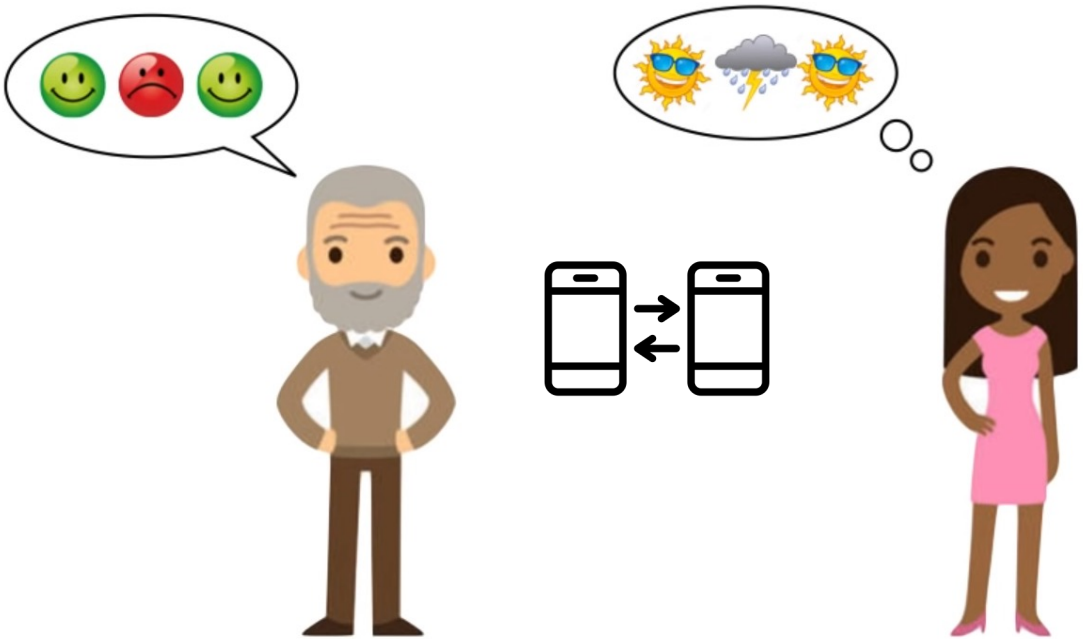


$\mathbb{P}(\mathbf{H}) = 2/3$
$\mathbb{P}(\mathbf{U}) = 1/3$

Bayes' rule:

$$\mathbb{P}(\mathbf{S}|\mathbf{H}) = \frac{\mathbb{P}(\mathbf{H}|\mathbf{S})P(\mathbf{S})}{\mathbb{P}(\mathbf{H})} = \frac{0.8 \cdot (2/3)}{2/3} = 0.8$$

Introducing Alice



Alice lives with Bob, but she's abroad for two days and they speak on the phone. On Wednesday she thought he was happy and on Thursday grumpy. They didn't talk about the weather, but she knows what his HMM looks like. From that information she tries to figure out what the weather might have been back home.

Alice predicts the most likely weather sequence

Wednesday

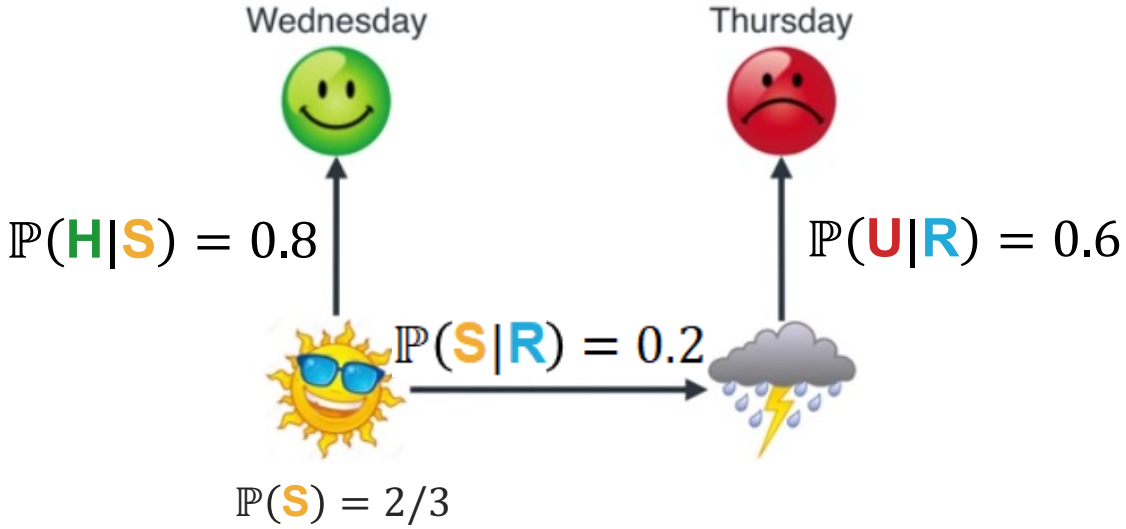
Thursday

If Bob seemed happy on the Wednesday and grumpy on the Thursday, what is the most likely weather sequence?

All four combinations are possible, but which has the highest probability?

Sunny followed by rainy? Or rainy both days? Or sunny both days? Or rainy and then sunny?

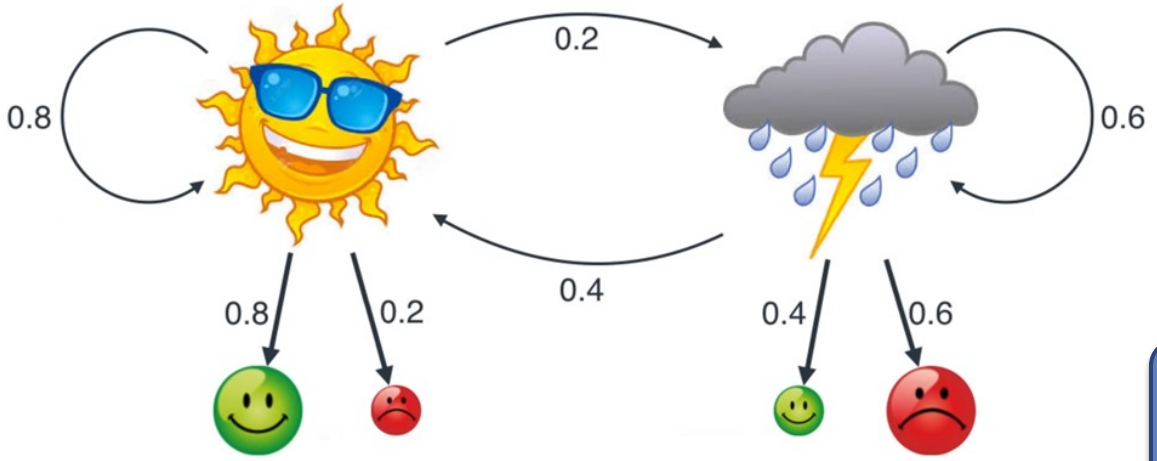
Likelihood of the hidden sequence



What is the probability of ”sunny” followed by ”rainy”, given Bob’s mood? $\mathbb{P}(X_1 = \mathbf{S}, X_2 = \mathbf{R}|Y_1 = \mathbf{H}, Y_2 = \mathbf{U}) = ?$

Hidden and visible states

$\mathbb{P}(\mathbf{S}) = 2/3$
$\mathbb{P}(\mathbf{R}) = 1/3$



$\mathbb{P}(\mathbf{H}) = 2/3$
$\mathbb{P}(\mathbf{U}) = 1/3$

Observed process

Hidden process

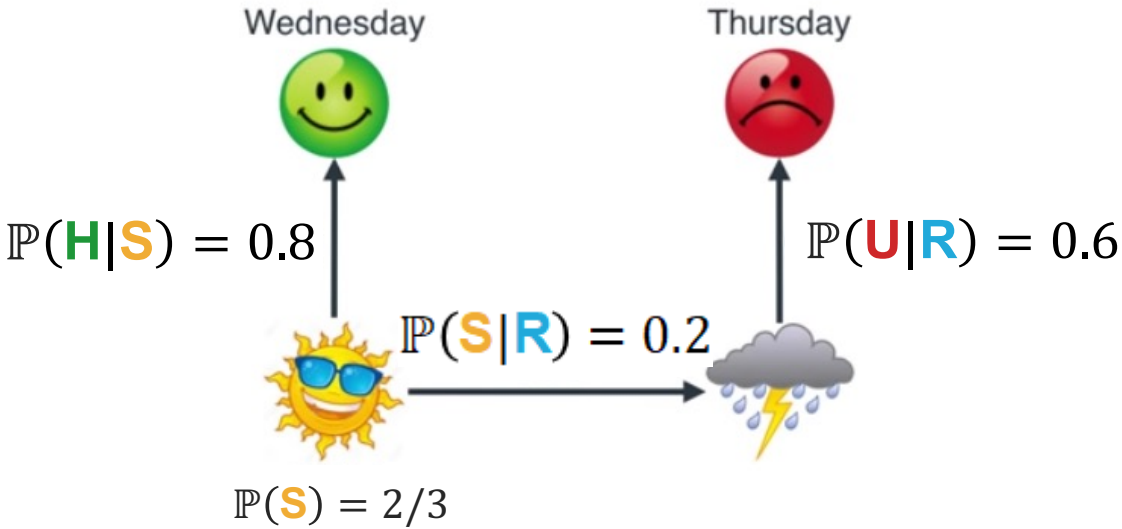
Alice only observes Bob's mood, without direct knowledge about the weather.

Chain rule – refresher

Factorization of joint probabilities (chain rule)

$$\mathbb{P}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}) = \mathbb{P}(\boldsymbol{A}|\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})\mathbb{P}(\boldsymbol{B}|\boldsymbol{C}, \boldsymbol{D})\mathbb{P}(\boldsymbol{C}|\boldsymbol{D})\mathbb{P}(\boldsymbol{D})$$

Likelihood of the hidden sequence



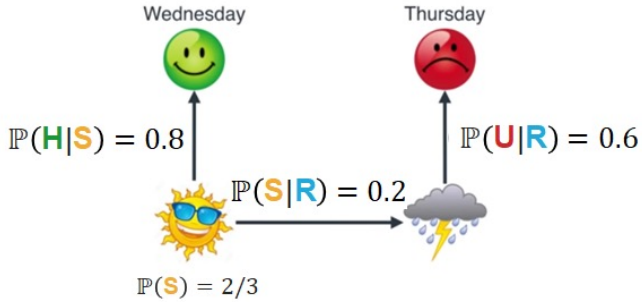
What is the probability of ”sunny” followed by ”rainy”, given Bob’s mood?

Likelihood of the hidden sequence

Wednesday

Thursday

$$\mathbb{P}(X_1 = \mathbf{S}, X_2 = \mathbf{R} | Y_1 = \mathbf{H}, Y_2 = \mathbf{U}) = ?$$



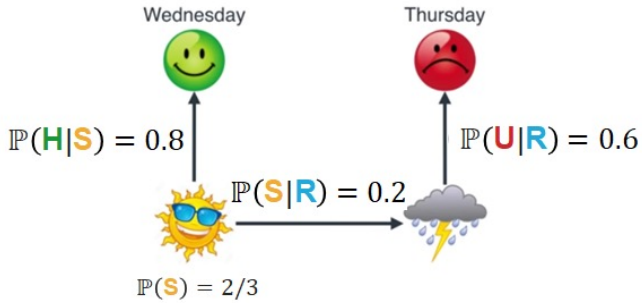
Likelihood of the hidden sequence

Wednesday

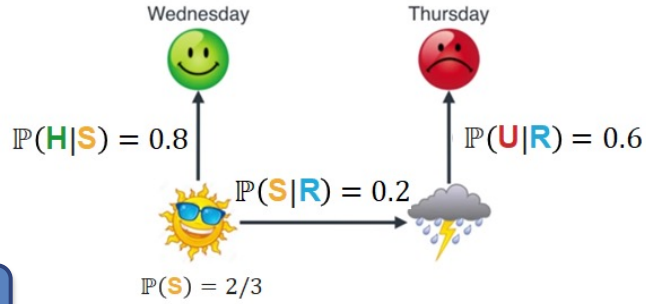
Thursday

$$\mathbb{P}(X_1 = \mathbf{S}, X_2 = \mathbf{R} | Y_1 = \mathbf{H}, Y_2 = \mathbf{U}) =$$

$$= \frac{\mathbb{P}(X_1 = \mathbf{S}, X_2 = \mathbf{R}, Y_1 = \mathbf{H}, Y_2 = \mathbf{U})}{\mathbb{P}(Y_1 = \mathbf{H}, Y_2 = \mathbf{U})}$$



Likelihood of the hidden sequence



Wednesday

Thursday

$$\mathbb{P}(X_1 = \mathbf{S}, X_2 = \mathbf{R} | Y_1 = \mathbf{H}, Y_2 = \mathbf{U}) =$$

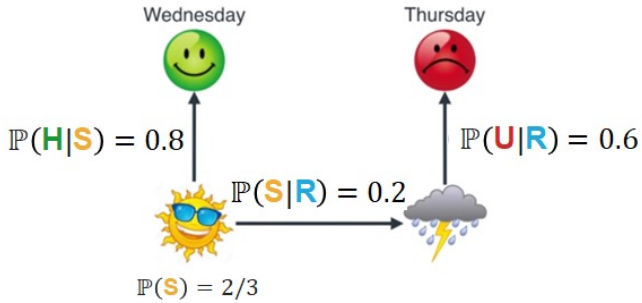
$$= \frac{\mathbb{P}(X_1 = \mathbf{S}, X_2 = \mathbf{R}, Y_1 = \mathbf{H}, Y_2 = \mathbf{U})}{\mathbb{P}(Y_1 = \mathbf{H}, Y_2 = \mathbf{U})}$$

Proportional to

$$\sim \mathbb{P}(X_1 = \mathbf{S}, X_2 = \mathbf{R}, Y_1 = \mathbf{H}, Y_2 = \mathbf{U})$$

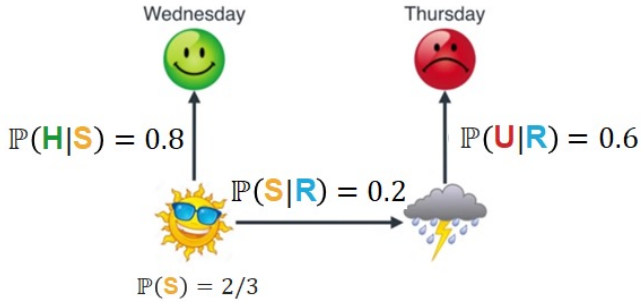
Not needed to determine optimal underlying state sequence.

Likelihood of the hidden sequence



$$\mathbb{P}(X_1, X_2, Y_1, Y_2) =$$

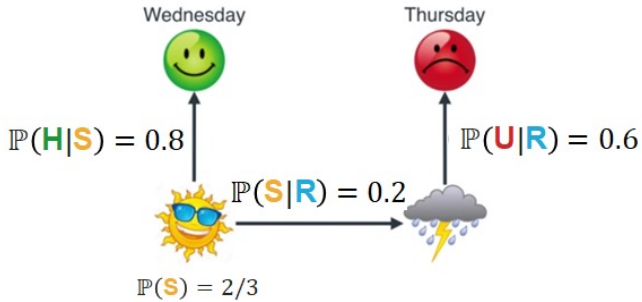
Likelihood of the hidden sequence



$$\mathbb{P}(X_1, X_2, Y_1, Y_2) = \mathbb{P}(Y_2|X_1, X_2, Y_1)\mathbb{P}(X_2|X_1, Y_1)\mathbb{P}(Y_1|X_1)\mathbb{P}(X_1) =$$

Only dependencies

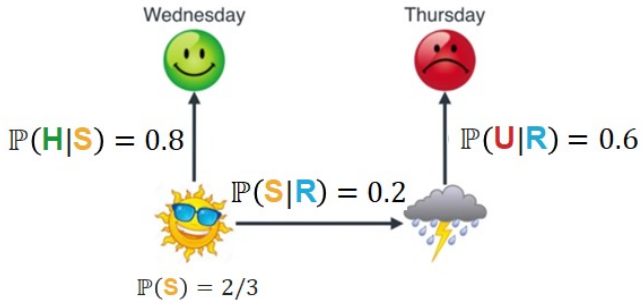
Likelihood of the hidden sequence



$$\mathbb{P}(X_1, X_2, Y_1, Y_2) = \mathbb{P}(Y_2|X_1, X_2, Y_1)\mathbb{P}(X_2|X_1, Y_1)\mathbb{P}(Y_1|X_1)\mathbb{P}(X_1) =$$

$$= \mathbb{P}(Y_2|X_2)\mathbb{P}(X_2|X_1)\mathbb{P}(Y_1|X_1)\mathbb{P}(X_1)$$

Likelihood of the hidden sequence



$$\mathbb{P}(X_1, X_2, Y_1, Y_2) = \mathbb{P}(Y_2|X_1, X_2, Y_1)\mathbb{P}(X_2|X_1, Y_1)\mathbb{P}(Y_1|X_1)\mathbb{P}(X_1) =$$

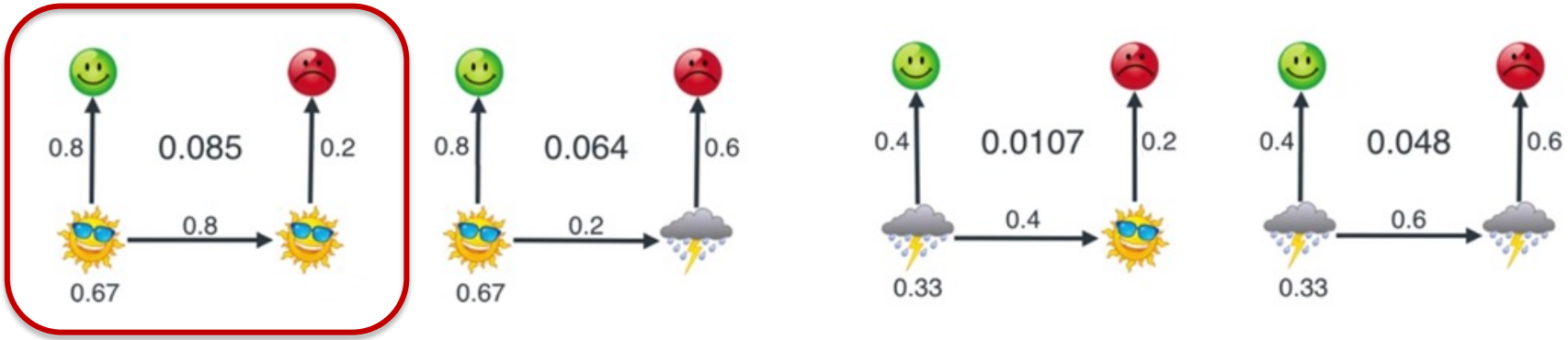
$$= \mathbb{P}(Y_2|X_2)\mathbb{P}(X_2|X_1)\mathbb{P}(Y_1|X_1)\mathbb{P}(X_1)$$

$$= \mathbb{P}(U|R)\mathbb{P}(R|S)\mathbb{P}(H|S)\mathbb{P}(S)$$

$$= 0.6 \cdot 0.2 \cdot 0.8 \cdot (2/3) = 0.064$$

Maximum likelihood

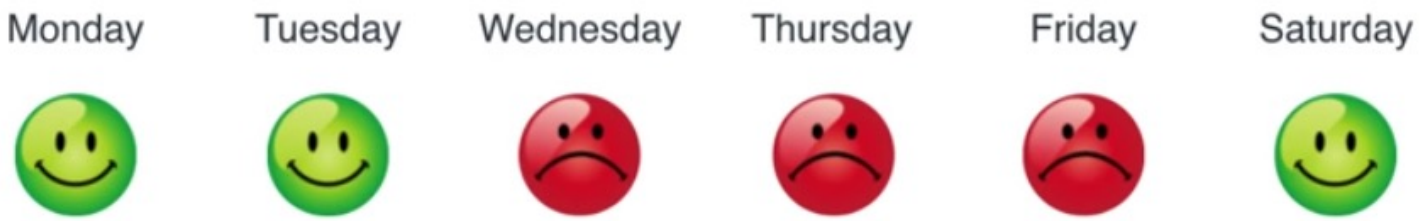
Do the same to all possible weather sequences



The winner is ”sunny” followed by ”sunny”

Maximum likelihood

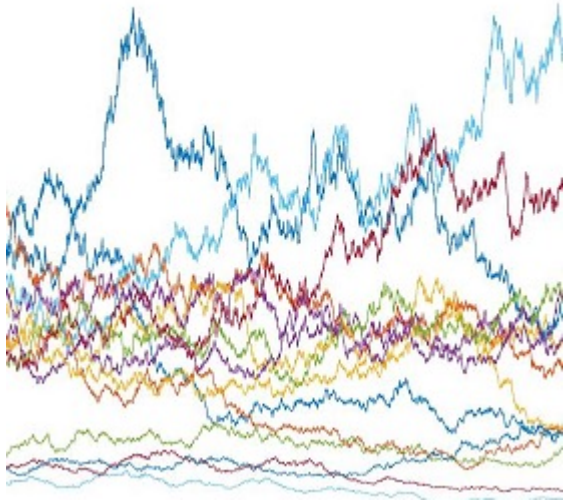
General problem: given an observed sequence, what is the most likely hidden sequence?



Hidden Markov models (HMMs)

A Hidden Markov model (HMM) consists of two related processes

- **A hidden process: Markov chain on the underlying state space**
- **An observed process: with outputs depending on the underlying state (typically not Markov)**



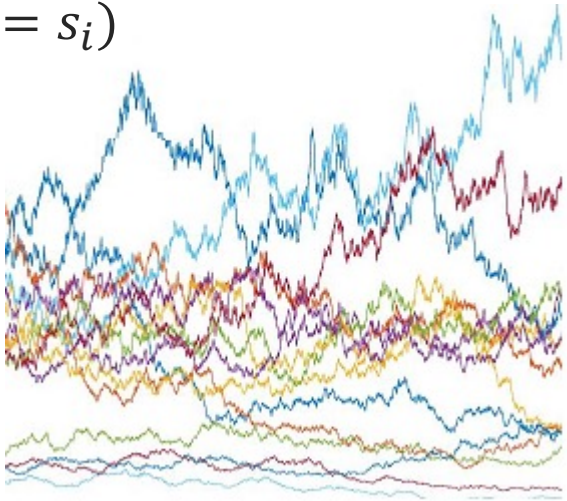
HMMs – components

A hidden Markov chain $X_1, X_2, X_3, \ldots$

- **A state space:** $S = \{s_1, s_2, \ldots, s_N\}$
- **Transition probabilities:** $p_{ij} = \mathbb{P}(X_{t+1} = s_j | X_t = s_i)$
- **Initial distribution:** $\pi_i = \mathbb{P}(X_1 = s_i)$

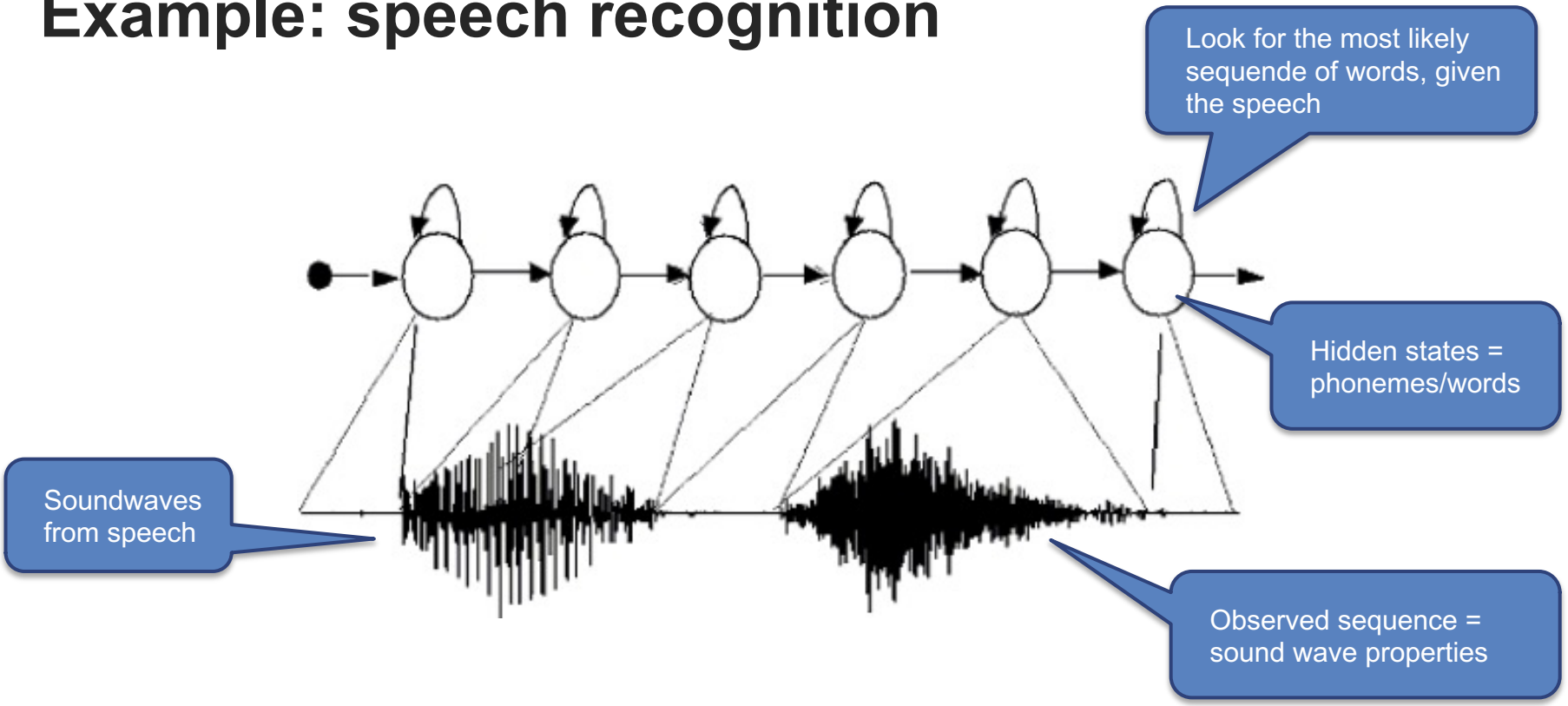
An observed process $Y_1, Y_2, Y_3, \ldots$

- **Emission set:** V **(discrete or continuous)**
- **Emission probabilities:**
  $b_i(y) = \mathbb{P}(Y_t = y | X_t = s_i)$ **for** $y \in V$

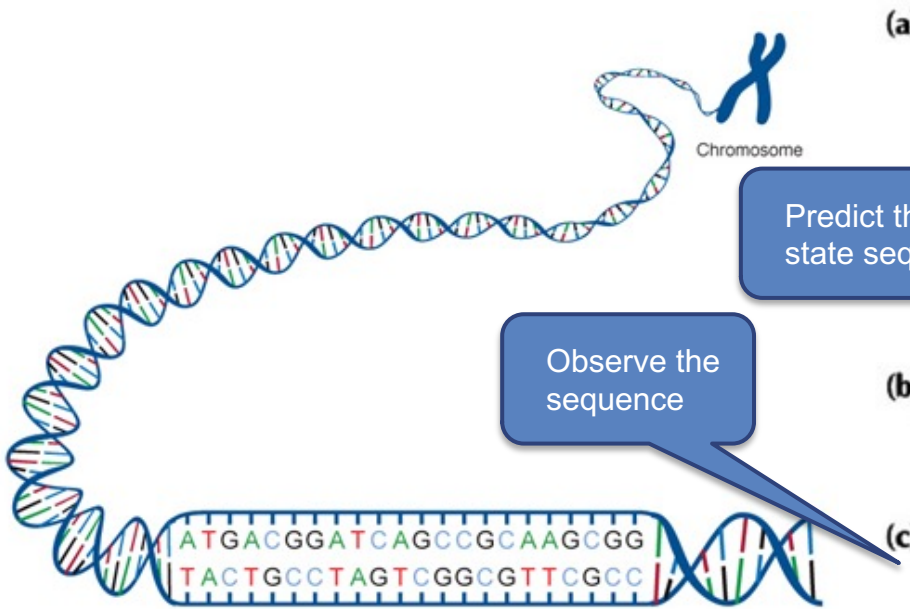


Example: speech recognition

Example: speech recognition



Example: DNA analysis

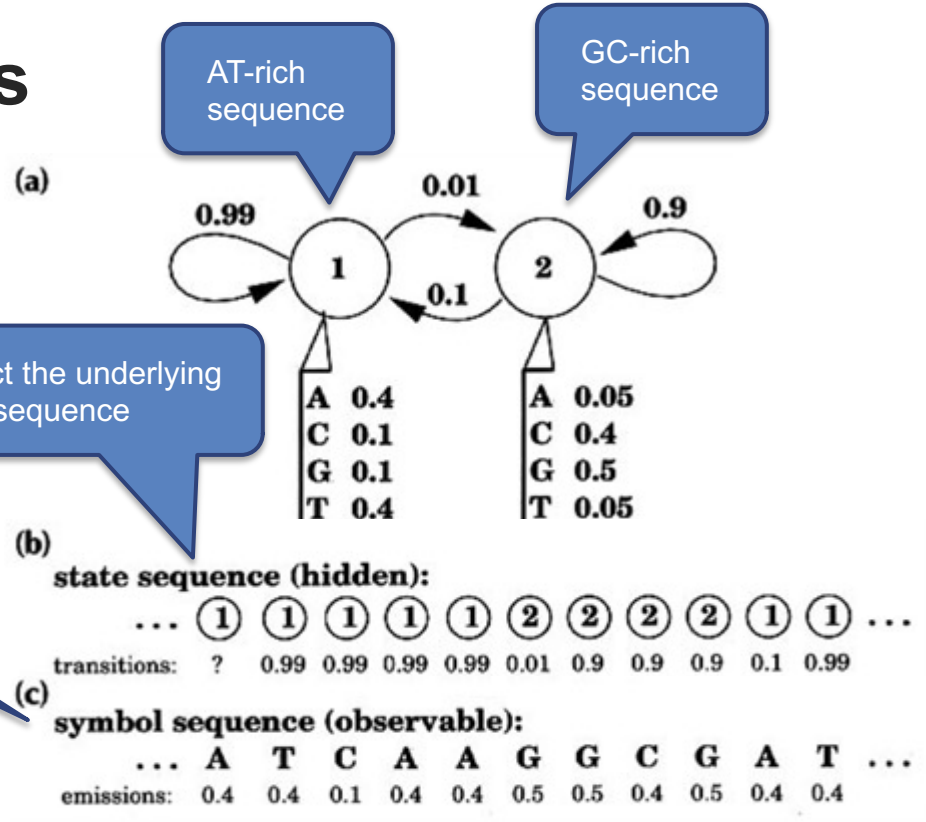


Observe the sequence

Predict the underlying state sequence

AT-rich sequence

GC-rich sequence



Example: robot localization

Given the robot pereptions, find the most likely location.

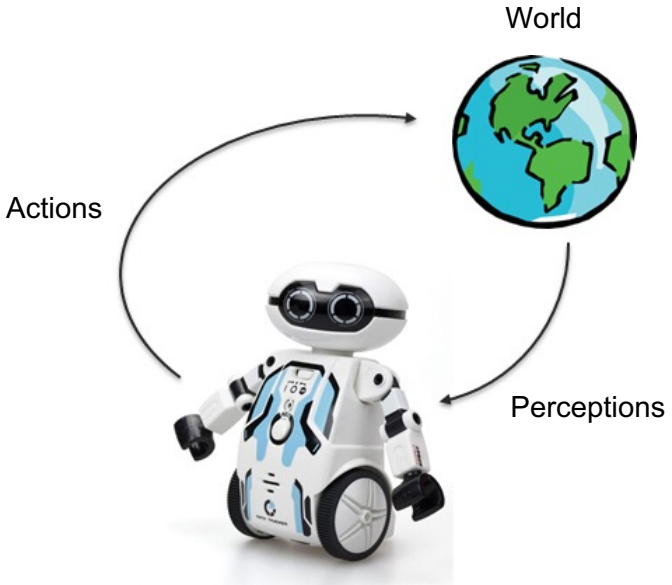
Observed sequence = distances to objects

Hidden states = locations

Markov Decision Processes

A Markov Decision Process (MDP) is a Markov Chain extended with actions and rewards.

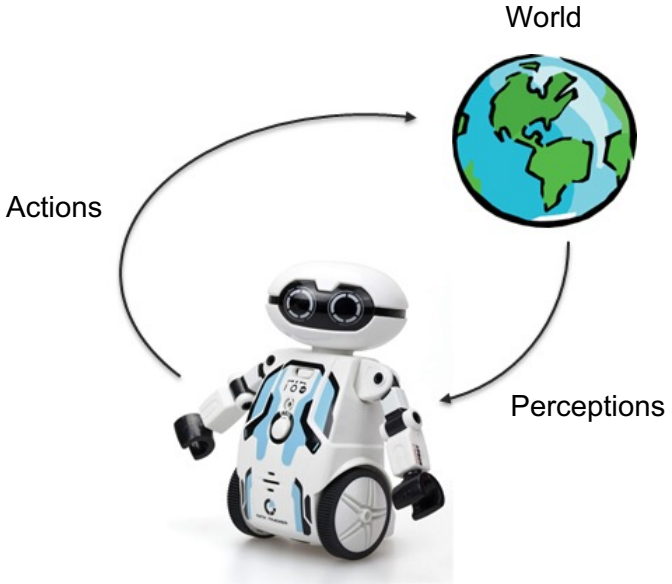
- **Given the environment/world state,**
  - **an *agent* takes an action**
  - **the world jumps to a new state**
  - **the agent receives a reward**



Markov Decision Processes: actions

Examples of actions

- Video game: move Left/Right/Up/Down
- Trading robot: Buy stock/Sell stock/Idle
- Treatment recommendation system: Do test/Give medication/Wait
- Person: Eat/Drink/Sleep/Work



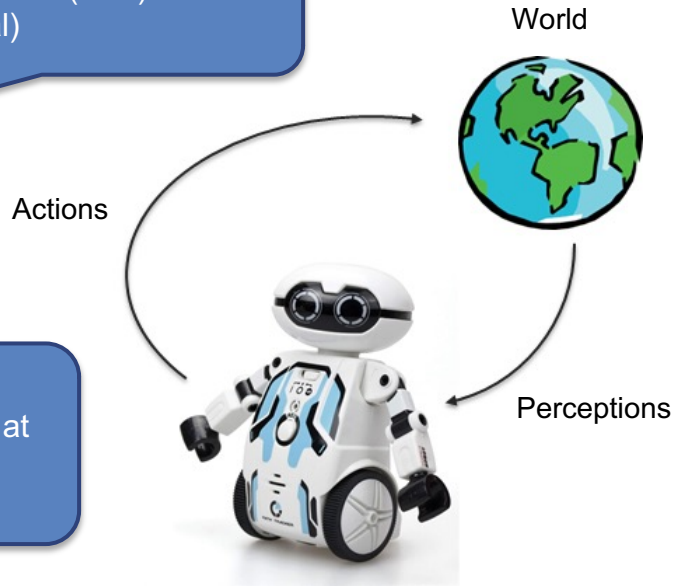
Markov Decision Processes: rewards

Examples of rewards

- Video game: score
- Trading robot: income
- Treatment recommendation system: health status
- Person: serotonin levels

Rewards can be positive, or negative (cost) or zero (neutral)

Rewards can be given at transitions or at states



Solving an MDP

Example: pshychological test

- A color is shown on the screen
  - Green, red or blue
- Select button a_1 or a_2
- Candies are given or removed if the chosen button is right or wrong
- How much candy can you collect in 5 minutes?

State

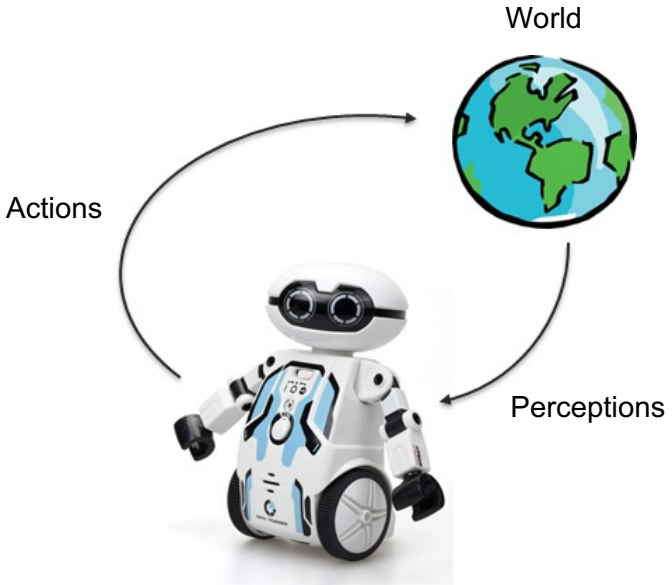
Action

Reward

Interaction

- An MDP describes the interaction between the agent and the environment
- Usually the agent does not have full knowledge of the environment
- The agent interacts with the environment to learn from it

Reinforcement learning

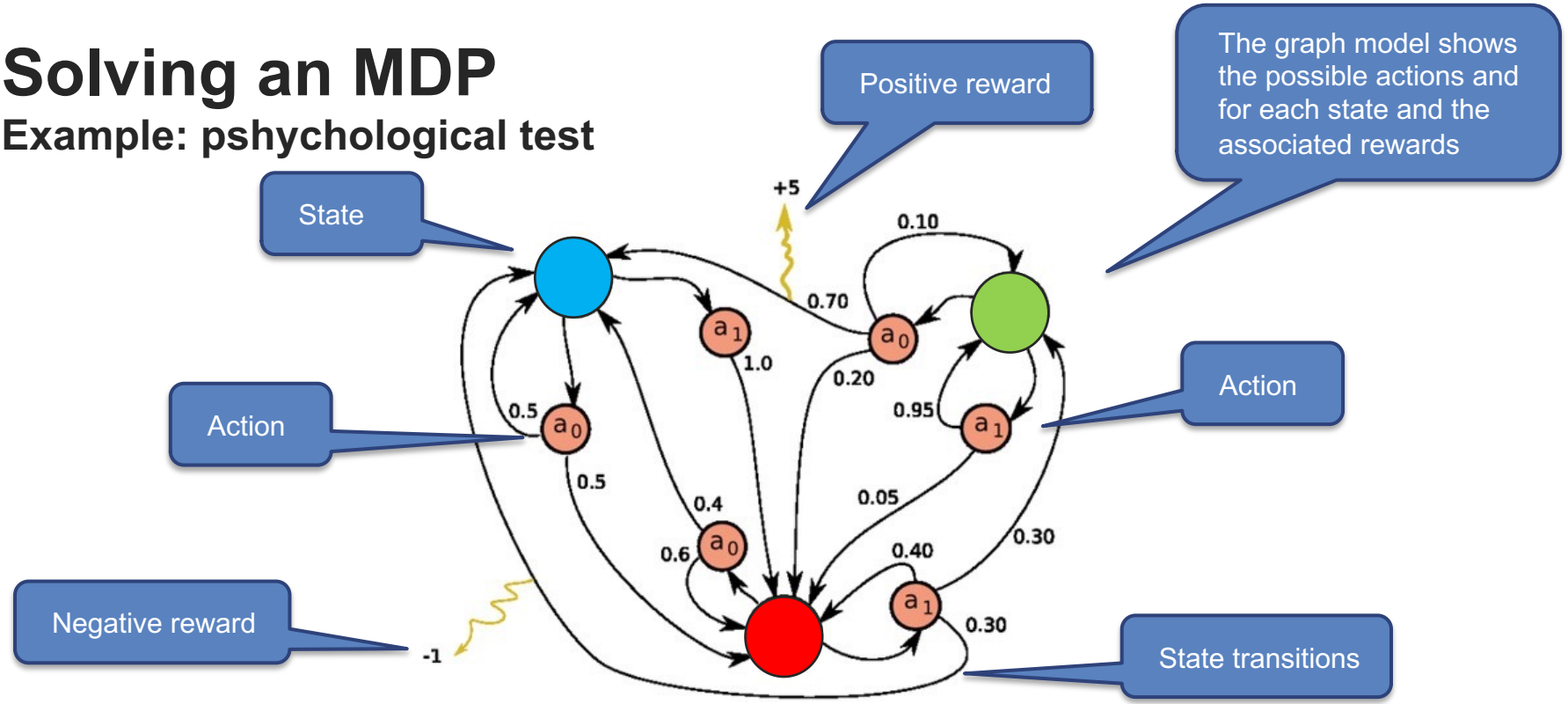


Solving an MDP means that …

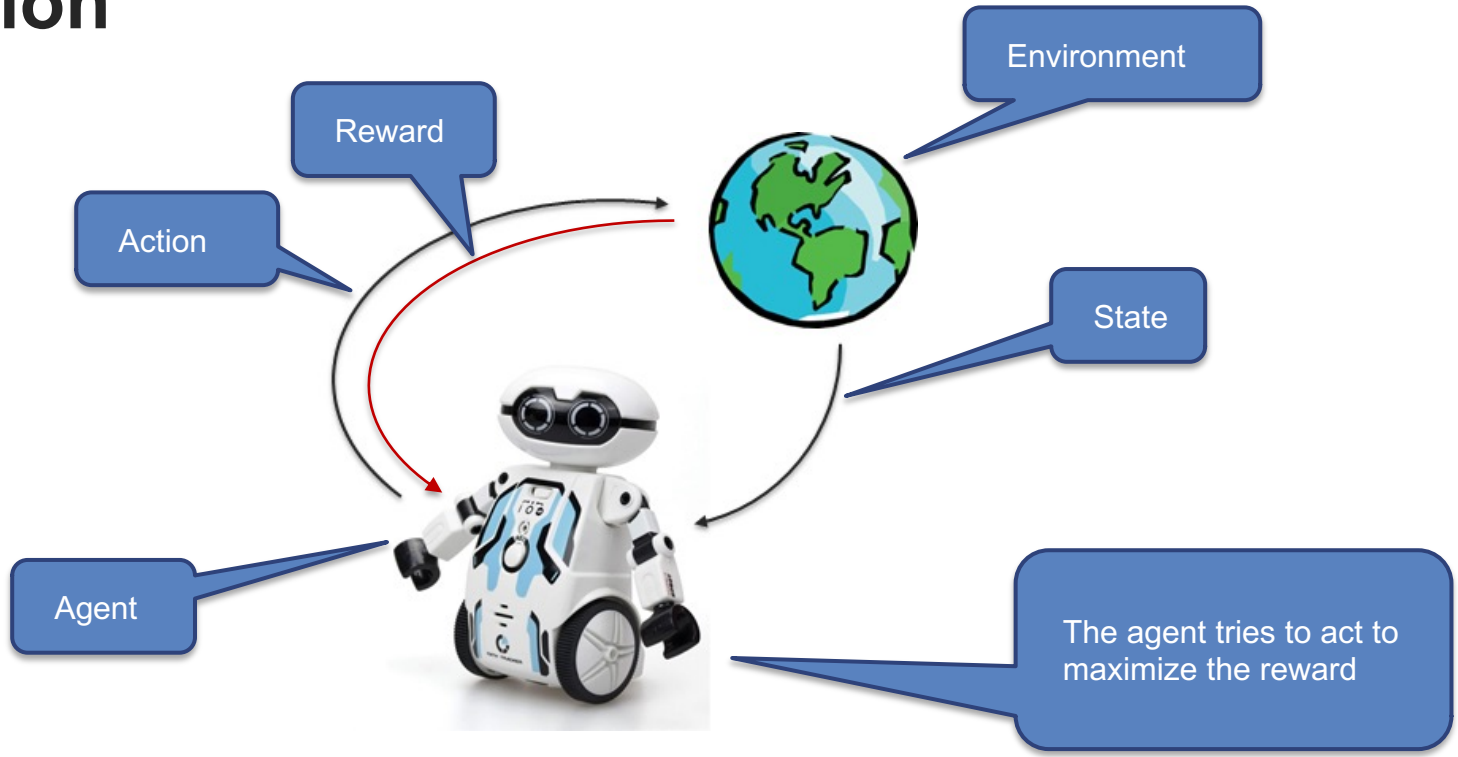
- … for each state in the state space, we compute the expected accumulated rewards from starting in that state
- The *value* of the state = expected accumulated rewards when starting in that state

Solving an MDP

Example: pshychological test



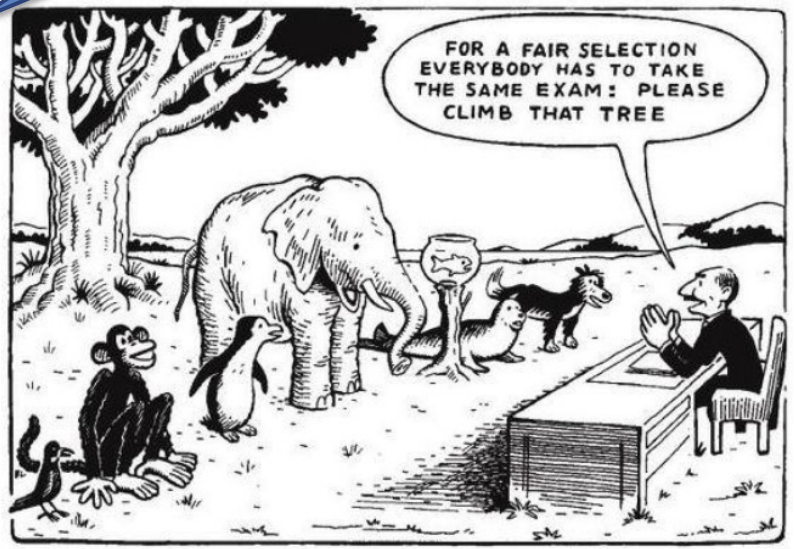
Interaction



Markov Decision processes – objective

- **Find a policy that gives the maximum reward in the long run**
- **A *policy* is a set of rules for which action to take in each state. E.g.**
  - **In state S_0: take action a_1**
  - **In state S_1: take action a_0**
  - **In state S_2: take action a_1**
- **$\pi(s) = a$: in state s, take action a**

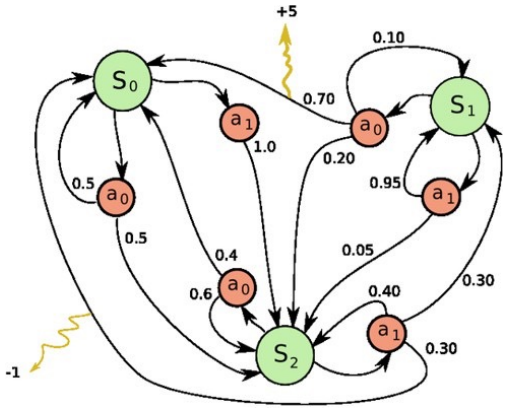
A policy can be deterministic or random



Challenge

- The challenge is to learn from experience and eventually find a policy (an optimal behavior) that gives the maximum expected reward
- Note that focusing on a reward in the short run or in the long run may result in different policies

Accumulated rewards



Any policy defines an interaction sequence:

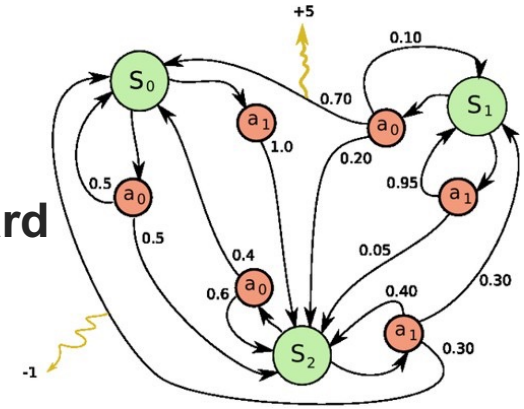
$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$$

This gives the following ***accumulated reward*** from time *t* onward:

$$r_t + r_{t+1} + r_{t+2} + \cdots$$

If the sequence is finite, then the sum converges. If it is infinite, then the sum may diverge.

Weighting of long term rewards

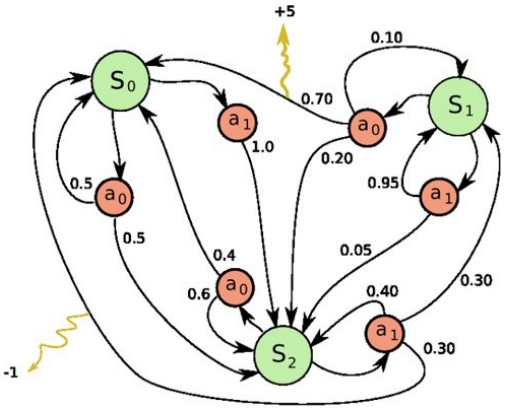


- In many cases, reward earlier is better than reward later.
- We include that by introducing a *discount factor* $\gamma \in [0, 1)$ in the model.
- We define the *utility* of a reward sequence $r_t, r_{t+1}, r_{t+2}, \ldots$ as

$$U_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

  since $\gamma < 1$, the sum converges even if the sequence is infinite!
- If $\gamma = 0$, then only the immediate reward matters. If $\gamma = 0.99$, then future rewards are very important.

Optimal policy

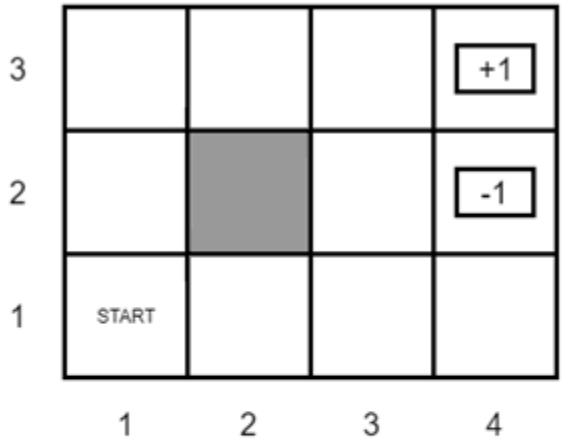


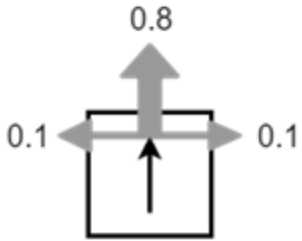
- Fix an MDP, a begin state s, and a discount factor γ
- For each possible policy, we can compute the expected utility (or value)
- The optimal policy is the one with maximum expected utility
- Since there are only finitely many policies, it is always possible to find an optimal policy – but not always feasible

Example: grid-world

This robot world can be viewed as an MDP.

- **States: 11 squares (1 unavailable)**
- **Actions: 4 moves**
  - **P(action performed) = 0.8**
  - **P(no action) = 0.2**
- **Rewards: 0, -1, or +1, as shown**

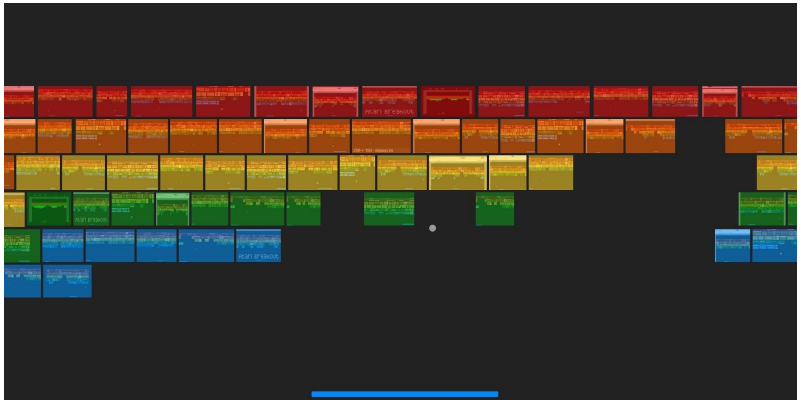




Example: Breakout

Breakout can be viewed as an MDP.

- **States: current configuration**
- **Actions: left, right**
- **Reward: score additions**



Bandit problems

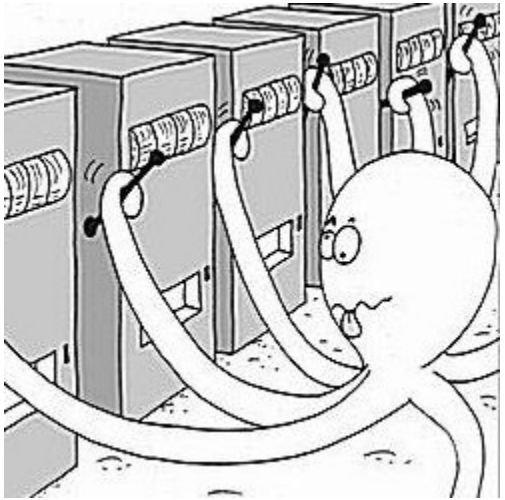
- An agent chooses between *k* actions in order to maximize the cumulative reward
- The agent chooses between *k* slot machines and want the one with the best payout
- Main challenge is to find the right mix of exploitation and exploration

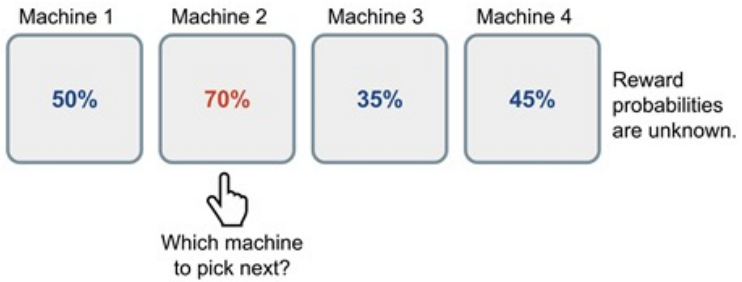
Bandit problems

- Can be used as conceptual models in a variety of problems
- Bandit problems are successfully modelled as MDPs

Bandit problems

- A multi-armed bandit problem is an MDP
  - *k* available slot machines
  - Action: which slot to pick next
  - Reward: the winnings of that slot
- Goal: to maximize the reward





Reinforcement learning in biology

- **Reinforcement learning is learning by trial and error, via rewards and punishments**
- **Almost all animals use it, including humans**

Reinforcement learning in biology

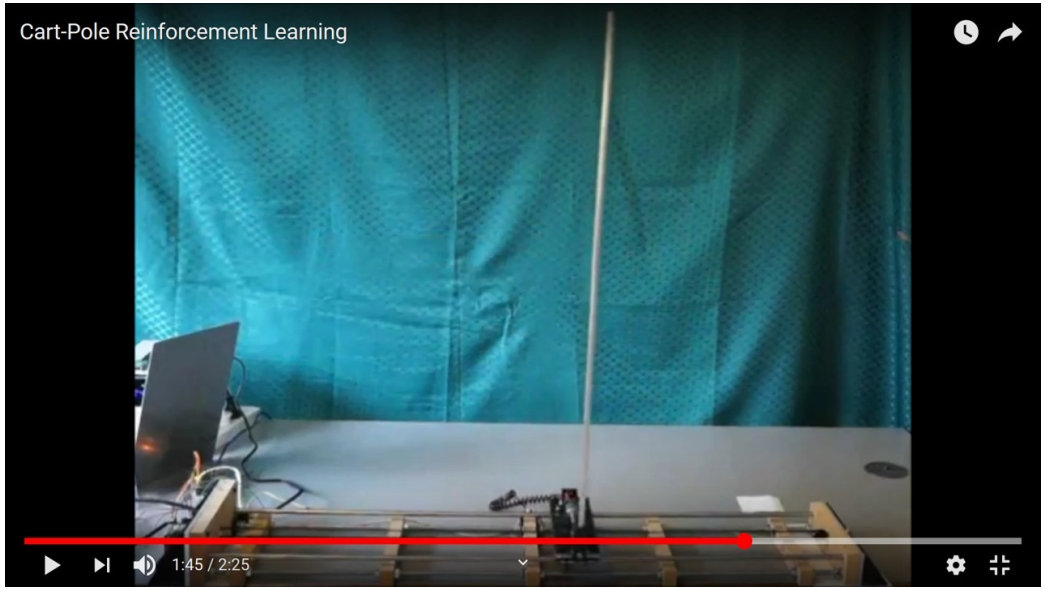
It is known that learn more efficiently from rewards along, without including punishments.

Dogs can learn advanced tricks from treats

Reinforcement learning in computer science

Snake robot. Gets reward when it moves forward.

Reinforcement learning in computer science

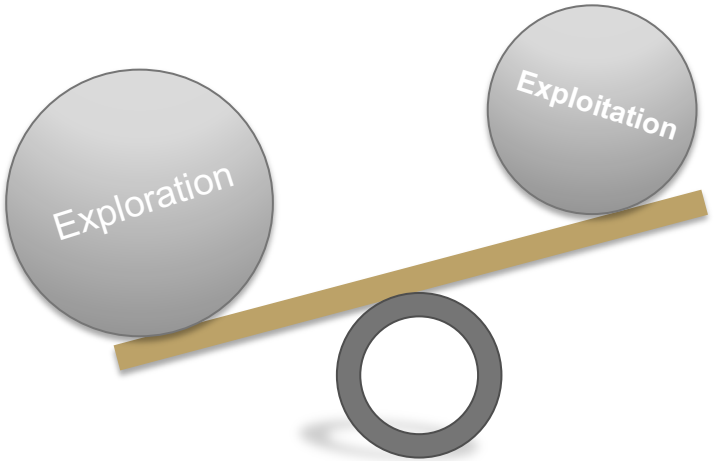


- States: 32 states of the form (angle, angle speed).
- Actions: move left or right.
- Reward: angle

Pole balancing.

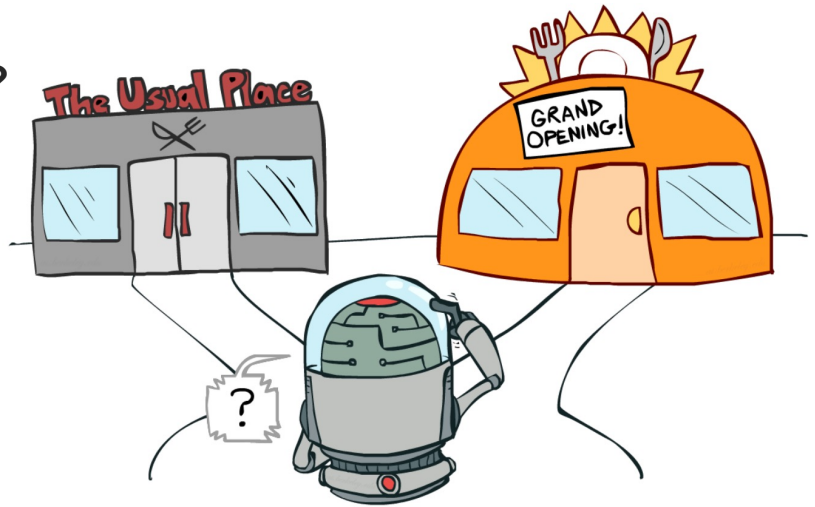
Exploration-exploitation

- **Explore**: taking a random action
- **Exploit**: taking the action that seems best at the moment (greedy action)



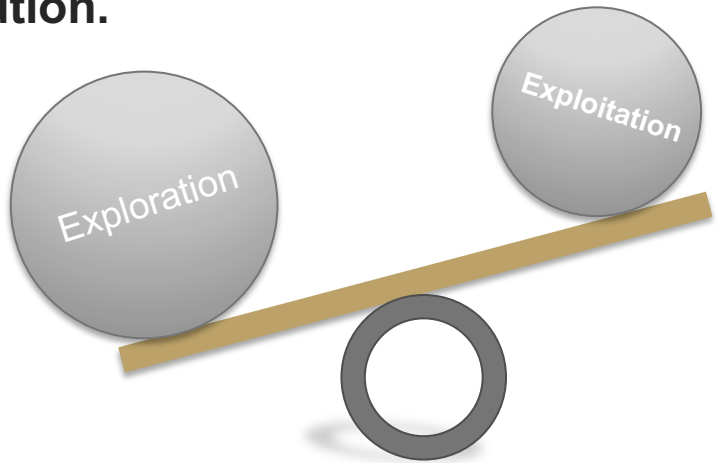
Exploration-exploitation trade-off

- Pick berries at the usual place or try a new one?
- Go to the usual restaurant or try a new one?
- Stay in your apartment or move to a new one?
- Stay on your job or move on?
- Stay in this town and country or move on?
- Take your usual walk or try a new route?



What is the point of exploring?

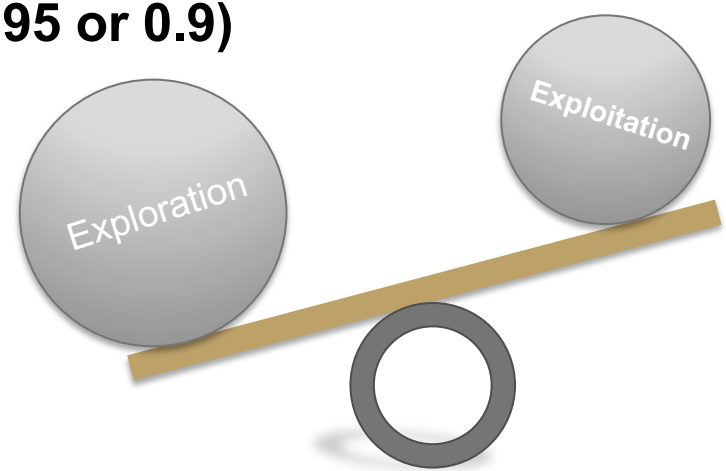
- If we exploit all the time (choosing historically best), and never try new moves, many actions and states will remain unknown.
- Exploration critical to find the optimal solution.
- If we explore all the time, we do not use accumulated experience to improve.
- A mix of exploitation and exploration is needed for optimal performance in the long run.



Epsilon-greedy algorithms

At each interaction

- **Explore with probability $1 - \epsilon$ (typically 0.05 or 0.1)**
- **Exploit with probability ϵ (typically 0.95 or 0.9)**
- **$0 \leq \epsilon \leq 1$: choice controls the level of exploration**



Q-learning

- A very general algorithm that can be used in many different algorithms: moving a snake, control a power plant, play video games…
- In Q-learning (Q = ”quality”) we assign a value $Q(s, a)$ to each initial state-action pair (s, a).
- We then want to find a policy (a set of state-action decision rules) $\pi(s) = a$ such that

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

$\pi(s)$: the action to take in state s according to policy π

argmax: the action a that maximizes $Q(s, a)$

- i.e. in state s we choose the action a that maximizes $Q(s, a)$

Important observation

U_t: Utility when starting at time t

Note the iterative pattern in the utility computation

$$U_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \cdots$$
$$= r_t + \gamma[r_{t+1} + \gamma \cdot r_{t+2} + \cdots]$$
$$= r_t + \gamma \cdot U_{t+1}$$

U_{t+1}: Utility for the same reward sequence, but when starting at time $t + 1$

Q-learning

- **Q-learning is an iterative algorithm with update formula at time t**

Current estimate

The value of the action giving the largest return on the next state

$$Q^{k+1}(s_t, a_t) = Q^k(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \cdot \max_a Q^k(s_{t+1}, a) - Q^k(s_t, a_t) \right]$$

Updated value

Learning rate

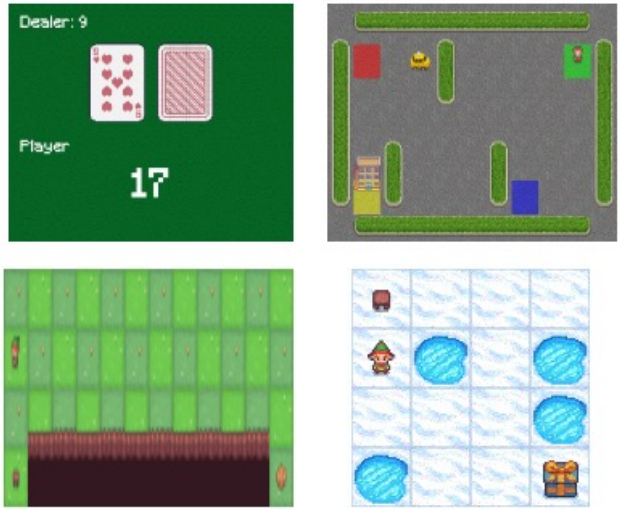
Discount rate

Q-learning

- Q-learning eventually finds the optimal policy for any finite MDP.
- It might take extremely long though.
- Moreover, table-based Q-learning has limited scalability.

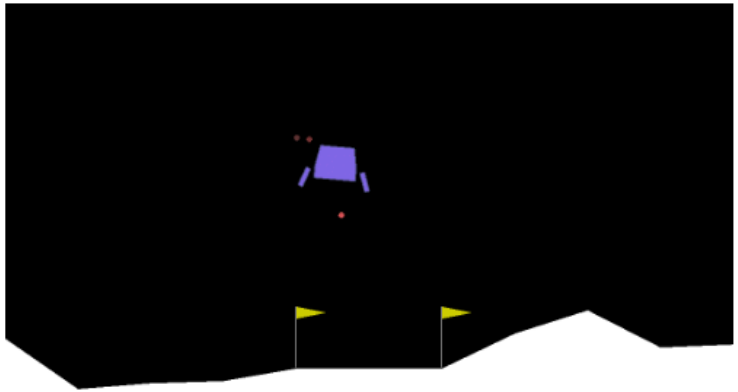
This week's assignment

- Markov decision processes in general
- Q-learning in an OpenAI gym environment



OpenAI gym

- Environment simulator modules
- Interact with the environment through various attributes



See
https://www.gymlibrary.dev/api/core/
for details

```
import gym
env = gym.make("LunarLander-v2", render_mode="human")
observation, info = env.reset(seed=42)
for _ in range(1000):
   action = policy(observation)  # User-defined policy function
   observation, reward, terminated, truncated, info = env.step(action)

   if terminated or truncated:
      observation, info = env.reset()
env.close()
```