CHALMERS
UNIVERSITY OF TECHNOLOGY

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

DAT405, lp2 2020, Module 4-5.

Reading material

Lecture 9

In Bishop (Pattern Recognition and Machine Learning):

Section 13 intro, 13.1 and 13.2.1 (rest of 13.2 is optional, but recommended)

In Sutton & Barto (Reinforcement Learning: An introduction):

Chapter 3

Simon Olsson

Repetition from last week and polls

Module 5.1: Markov models

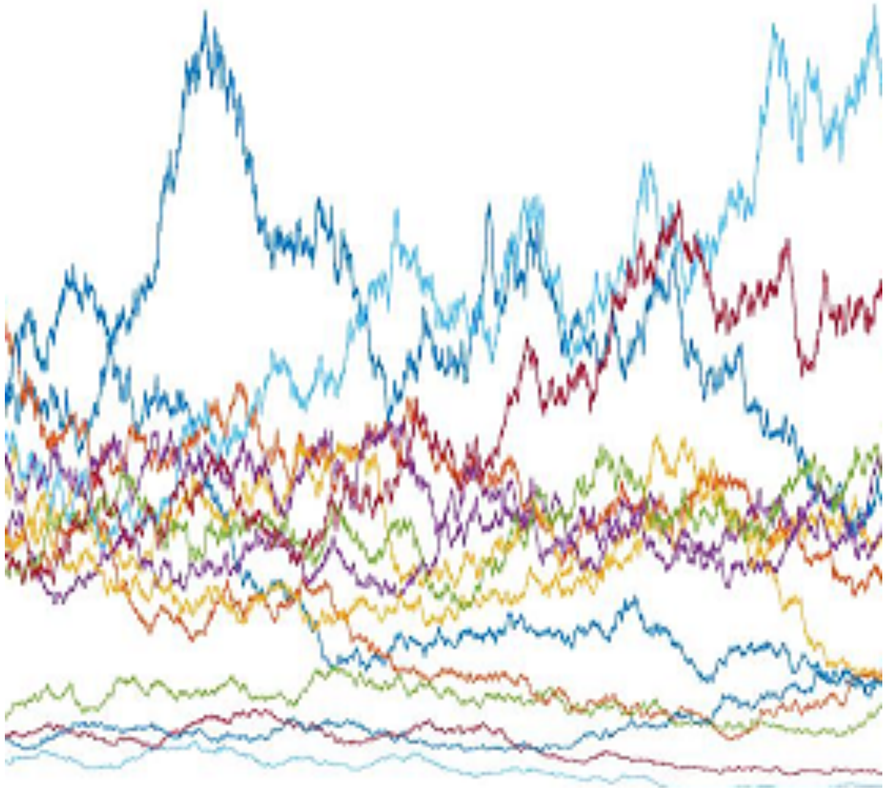
Markov processes

Random process

Is a function that varies with time and assigns the outcome of a ‘random experiment’ to each time-point, *t.*

Markov process

Is a *random process* where the next state only depends on the current state



Markov chains

A Markov chain

Is a Markov process that jumps between *discrete* states, often also at *discrete* time-points.

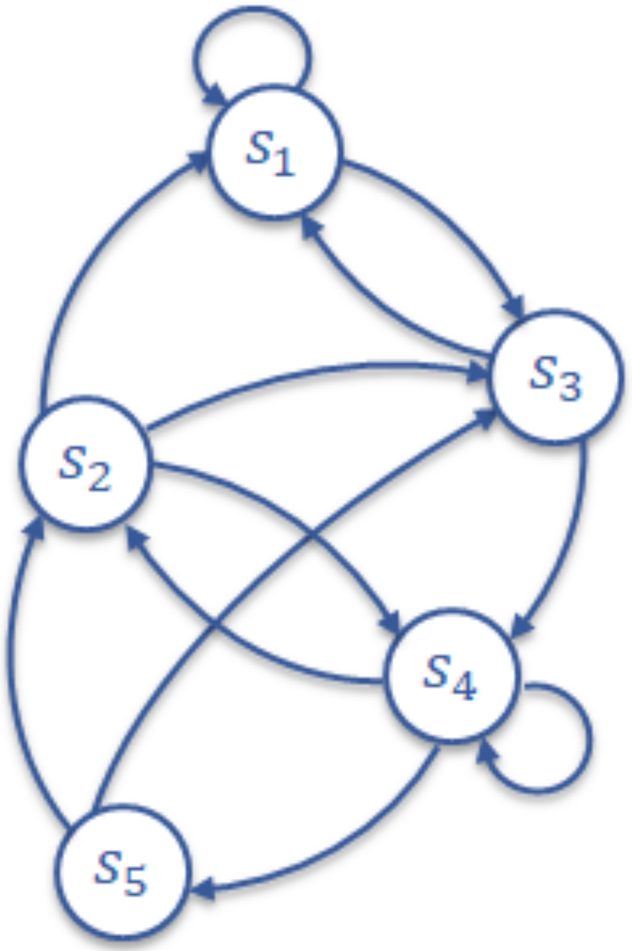
The Markov property

$$p(\text{next state} \mid \text{all previous states}) = p(\text{next state} \mid \text{previous state})$$

$$p(x_t \mid x_{t-\delta}, x_{t-2\delta}, \ldots, x_0) = p(x_t \mid x_{x-\delta})$$

$$x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow \cdots \longrightarrow x_N$$

$$\mathcal{T} = \{s_1, s_1, s_2, \ldots, s_5\}$$



Visual representation of a Markov Chain
Nodes are states
Edges are possible transitions.

Markov chains — more formally

The random process, $x_1, x_2, x_3, \ldots$, adopts **one** of the states, $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$, at any given time.

$$x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow \ldots \longrightarrow x_N$$

The state, s_i, adopted by the process at, x_t, is given by *the transition probability*, $p(s_i \mid s_j)$, where s_j is the state adopted the process at, $x_{t-\delta}$.

$$\mathcal{T} = \{s_1, s_1, s_2, \ldots, s_5\}$$

The *initial state,* $x_1 = s_i$, of the Markov process is given by the *initial distribution,*

$$\pi_i = p(x_1 = s_i)$$

Markov chains — more formally

The **random process**, $x_1, x_2, x_3, \ldots$

The **state space** $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$.

The **transition probabilities**, $T_{ij} = p(s_i \mid s_j)$

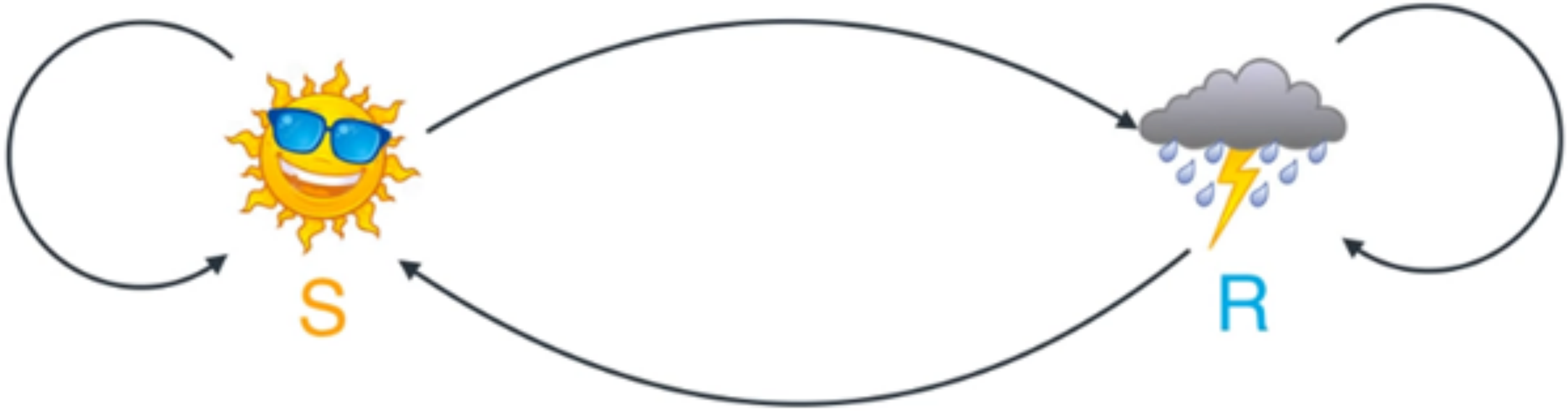
The **initial condition** $\pi_i = p(x_1 = s_i)$

$$x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow \ldots \longrightarrow x_N$$

$$\mathcal{T} = \{s_1, s_1, s_2, \ldots, s_5\}$$

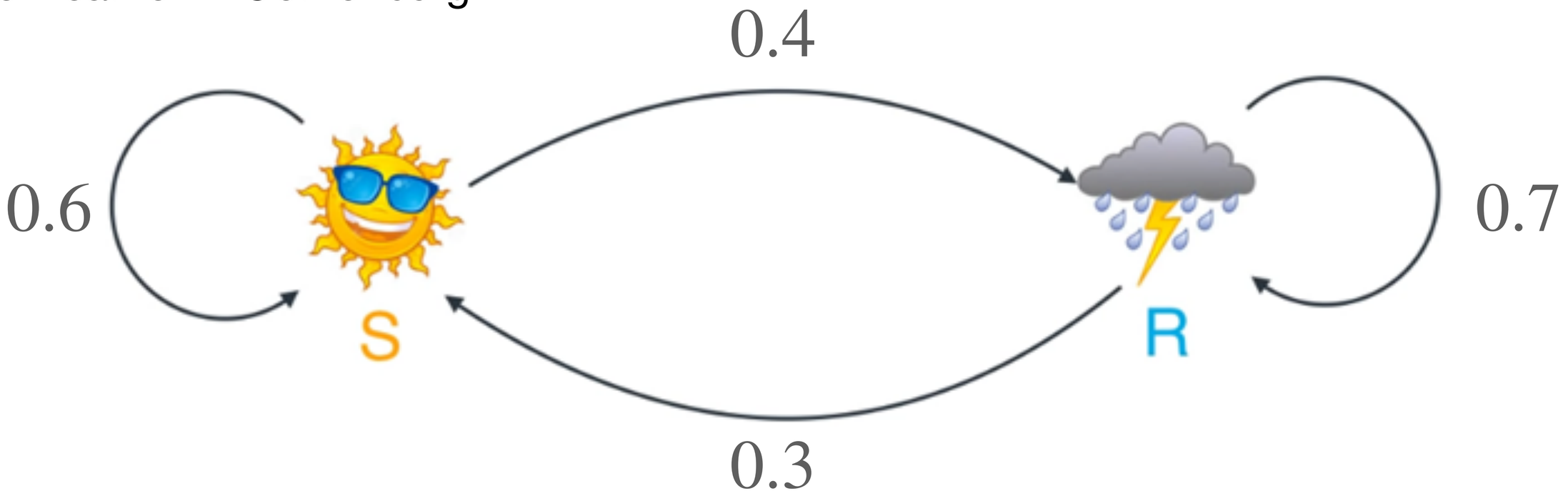
Markov chains — an example

The weather in Gothenburg



Markov chains — an example

The weather in Gothenburg



$$P = \begin{pmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{pmatrix}$$

Markov chains — estimation (1)

An *observation*, $x_1, x_2, x_3, \ldots$ from July 2020

The **state space** $\mathcal{S} = \{S, R\}$.

Markov chains — estimation (1)

An *observation*, $x_1, x_2, x_3, \ldots$ from July 2020

The **state space** $\mathcal{S} = \{S, R\}$.

We can estimate the initial distribution $\pi = (\pi_R, \pi_S)$

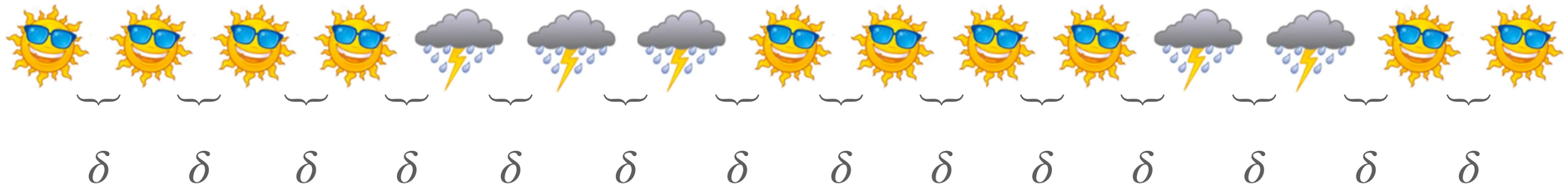
$$\pi_R = \frac{5}{15} = 1/3$$

$$\pi_S = \frac{10}{15} = 2/3$$

Markov chains — estimation (2)

An *observation*, $x_1, x_2, x_3, \ldots$

The **state space** $\mathcal{S} = \{S, R\}$.



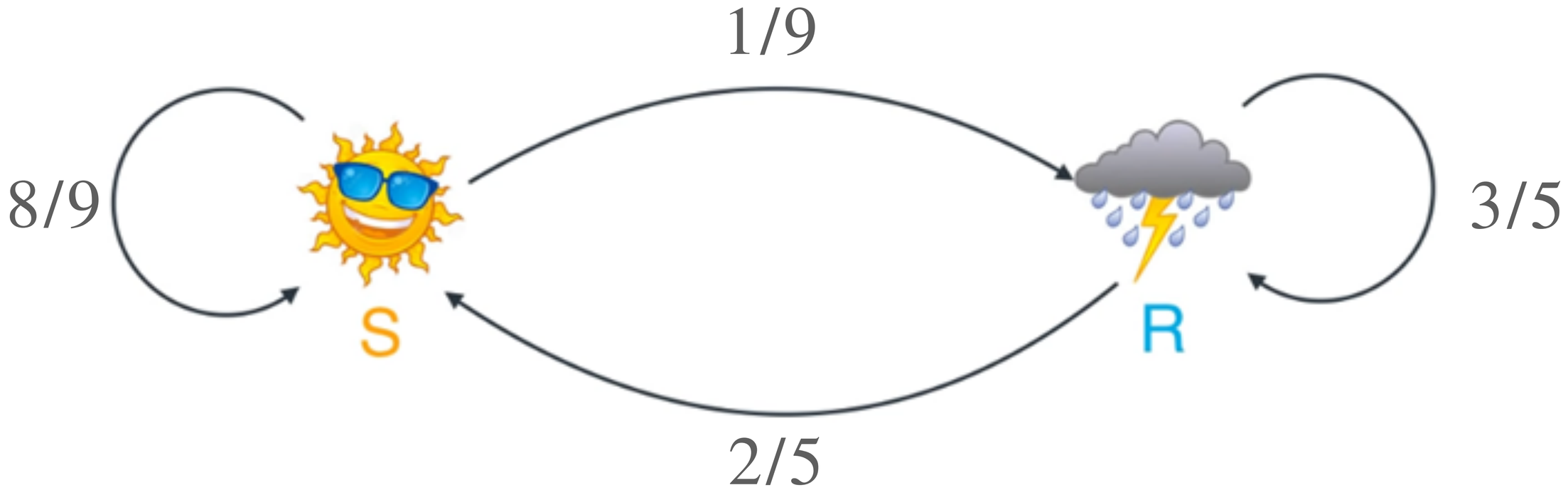
We can similarly estimate the transition probabilities

$p_{ij} = p(\mathcal{S}_i \mid \mathcal{S}_j)$

$$p_{SS} = \frac{3+3+2}{9} = 8/9 \qquad p_{SR} = \frac{1}{9}$$

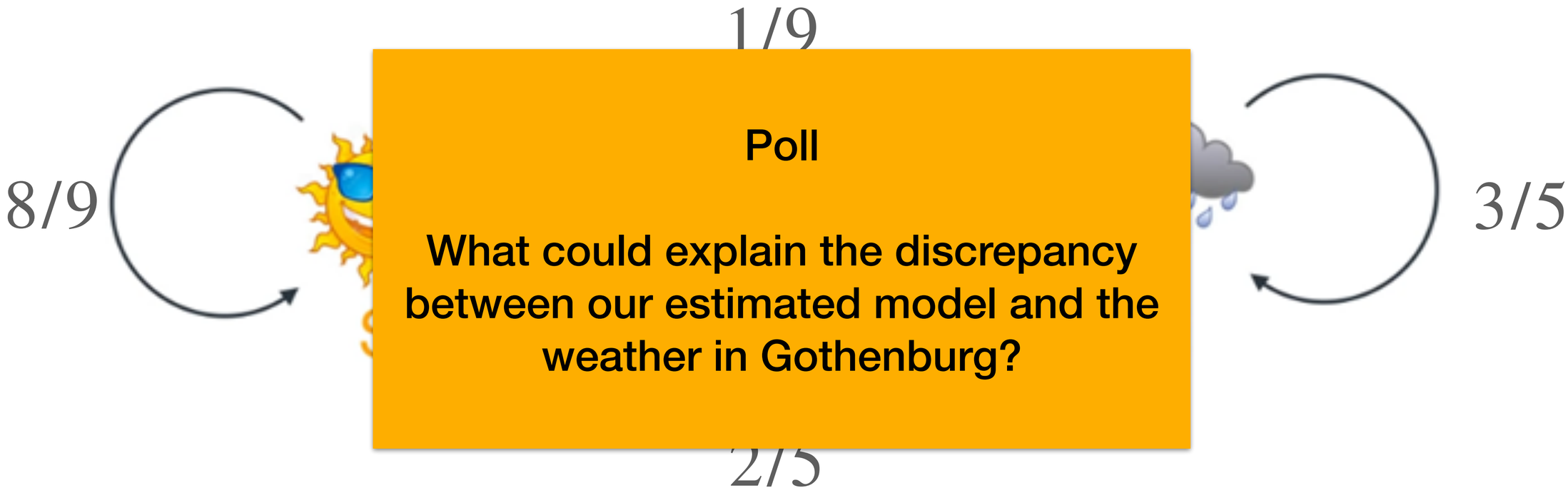
$$p_{RS} = \frac{2}{5} \qquad p_{RR} = \frac{3}{5}$$

Markov chains — Our estimate



$$P = \begin{pmatrix} 8/9 & 1/9 \\ 2/5 & 3/5 \end{pmatrix}$$

Markov chains — Our estimate



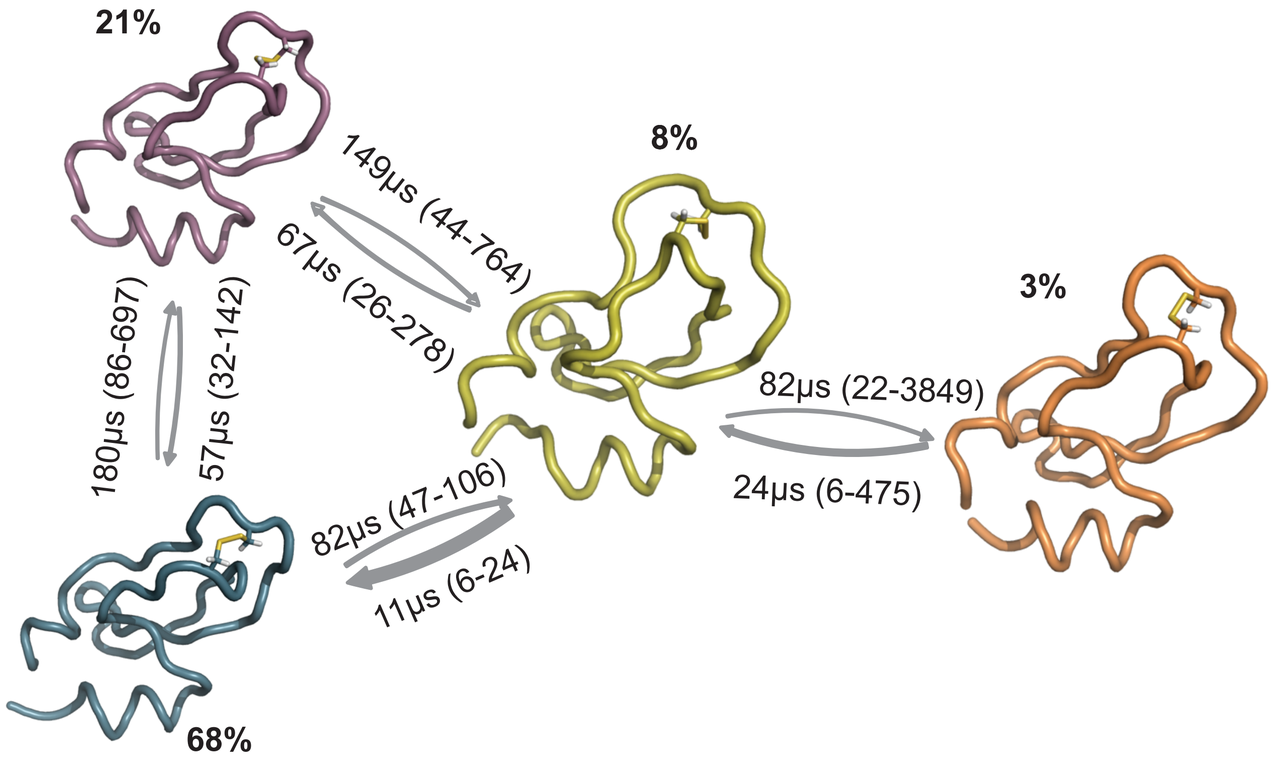
Poll

What could explain the discrepancy between our estimated model and the weather in Gothenburg?

$$P = \begin{pmatrix} 8/9 & 1/9 \\ 2/5 & 3/5 \end{pmatrix}$$

Markov chains — Applications

Molecular dynamics
Analysis of molecular dynamics simulation data from HPC facilities to gain biological and physical insights



Hidden Markov models

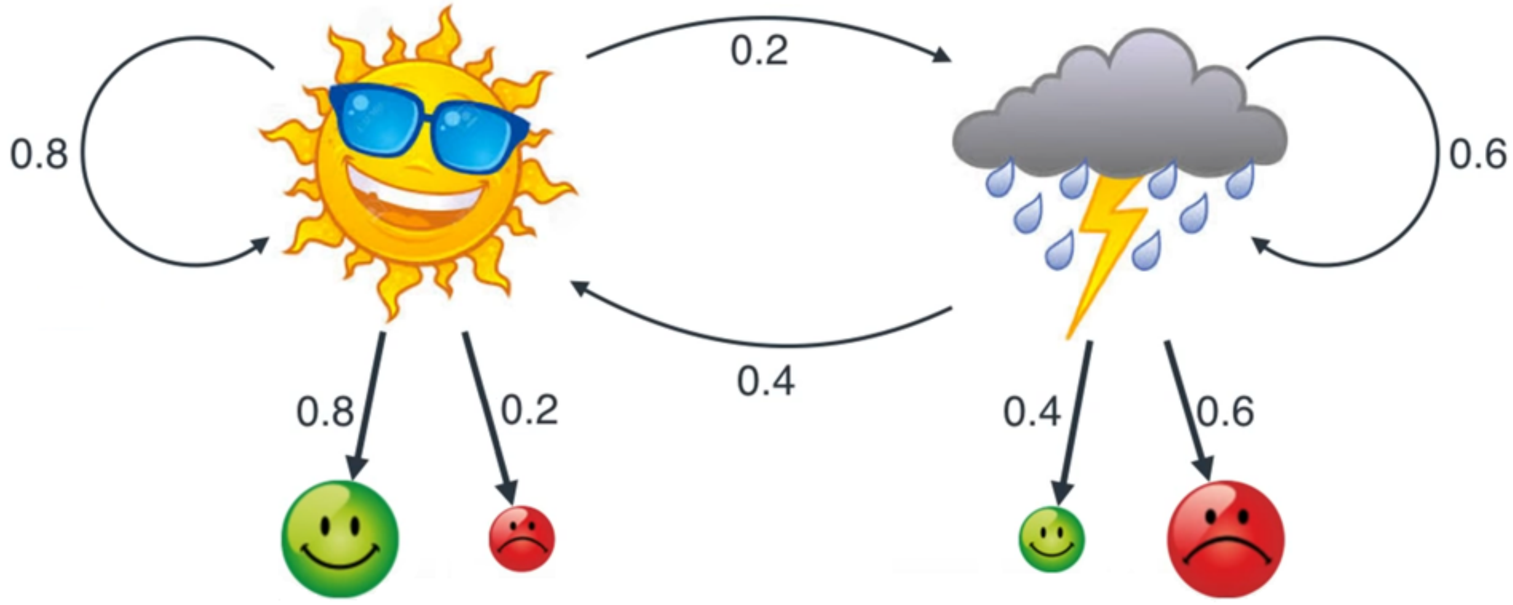
Bob’s friends say his mood changes with the weather.

Hidden Markov models

Bob’s friends say his mood changes with the weather.

Poll
How can we check this?

Hidden Markov models



Hidden variable
Weather

Observed variable
Bob’s mood

Hidden Markov models — details

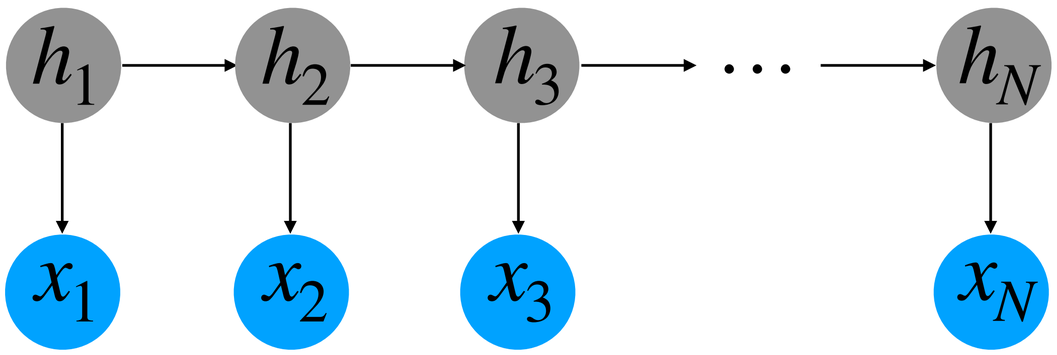
In *Hidden Markov models* (HMM), we distinguish between

a Hidden random process

a Markov chain, $h_1, h_2, \ldots$. Hidden because it is generally *not* observed, and the

Observed process

$x_1, x_2, \ldots$ takes a random state conditioned on the state of the hidden process. Typically non-Markovian.



Hidden Markov models — formally

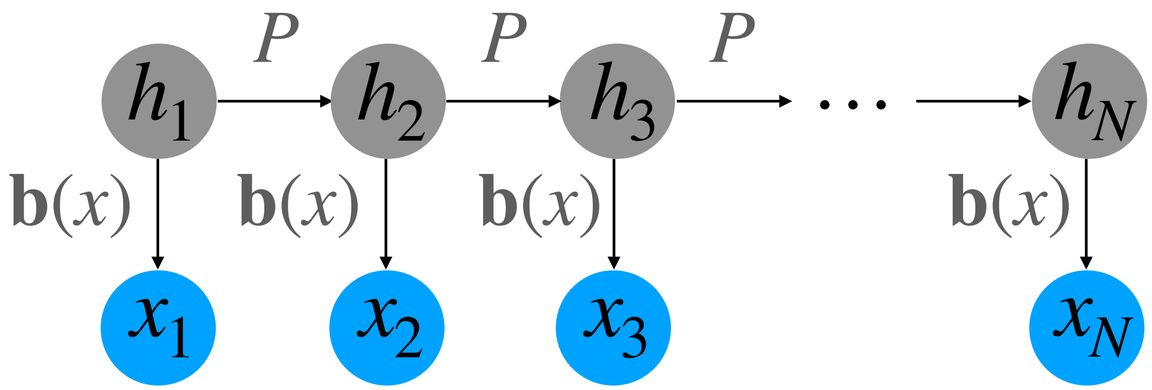
Hidden Markov models (HMM), we have

a Hidden Markov chain, $h_1, h_2, \ldots$

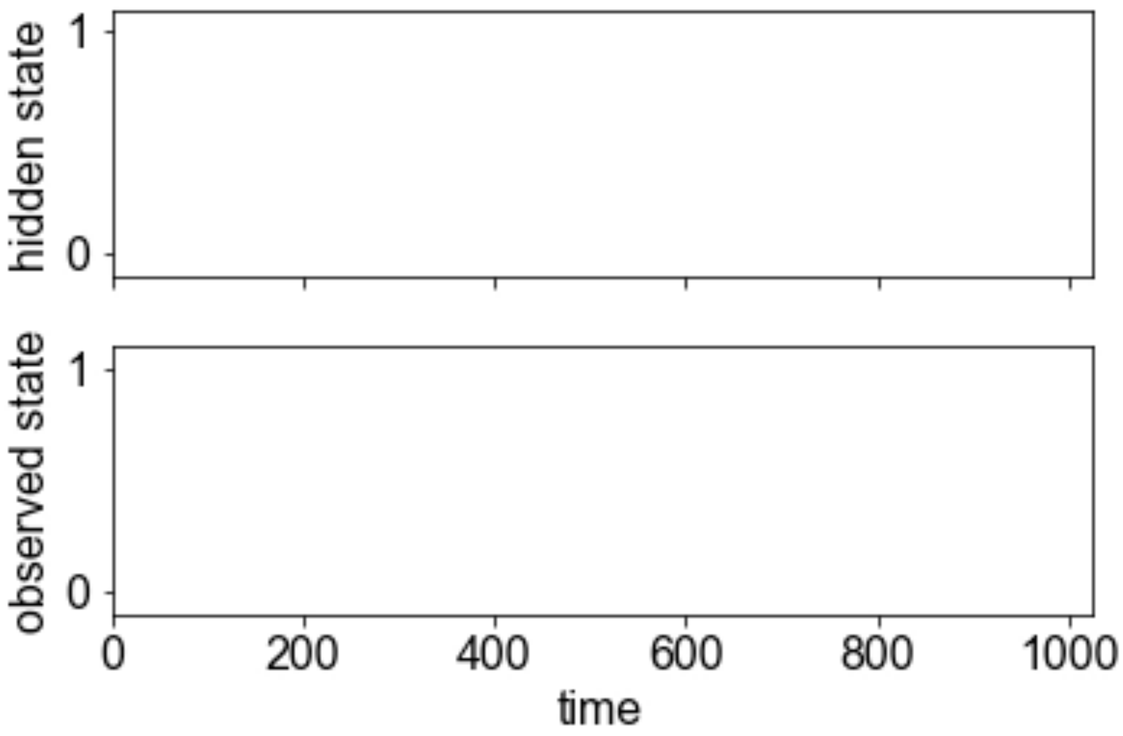
With the **state space** $\mathcal{S} = \{s_1, s_2, \ldots s_N\}$ and **transition probabilities** $p_{ij} = p(h_t = i \mid h_{t-\delta} = j)$, and the **initial condition** $\pi_i = p(h_1 = s_i)$ and

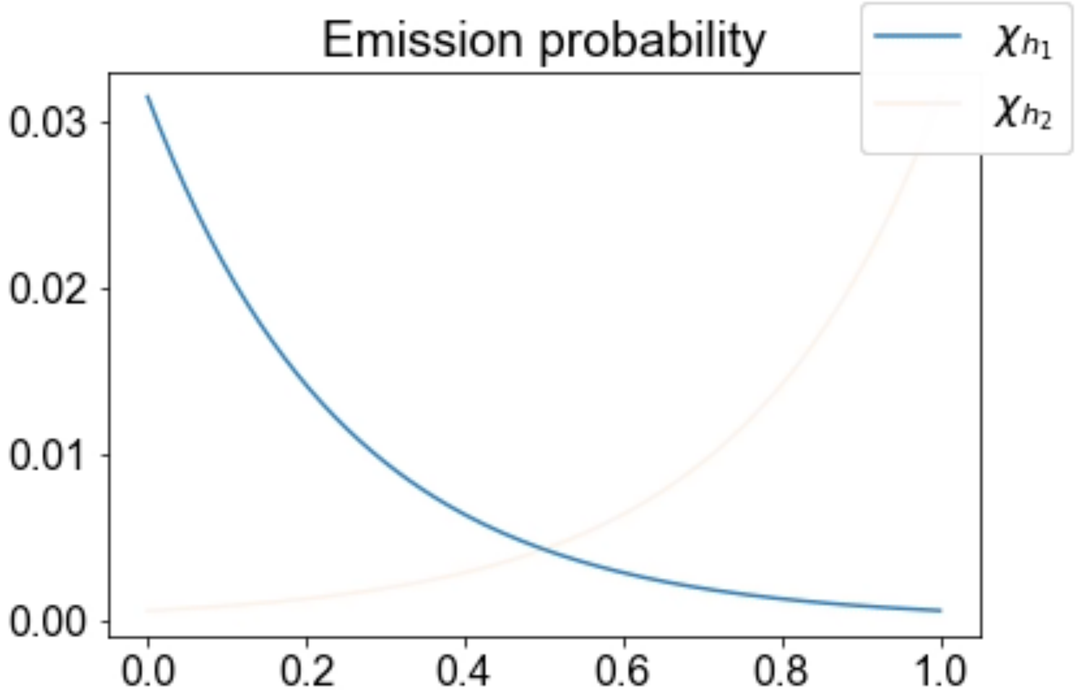
an Observed process, $x_1, x_2, \ldots$

on the Emission set, V, which is either discrete or continuous, with emission probabilities, $b_i(x) = p(x_t = x \mid h_t = s_i)$.

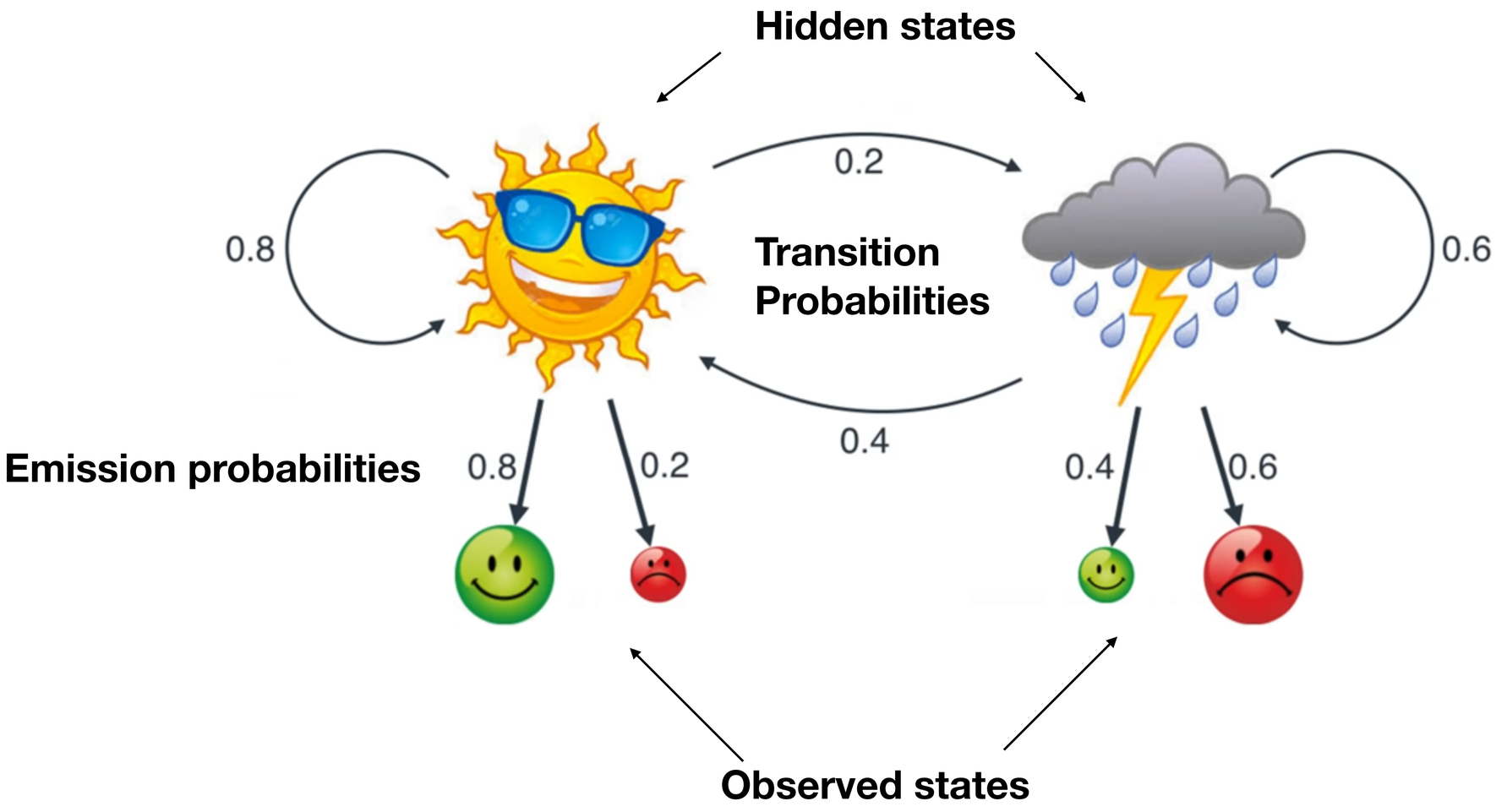


Hidden Markov models — illustration





Hidden Markov models

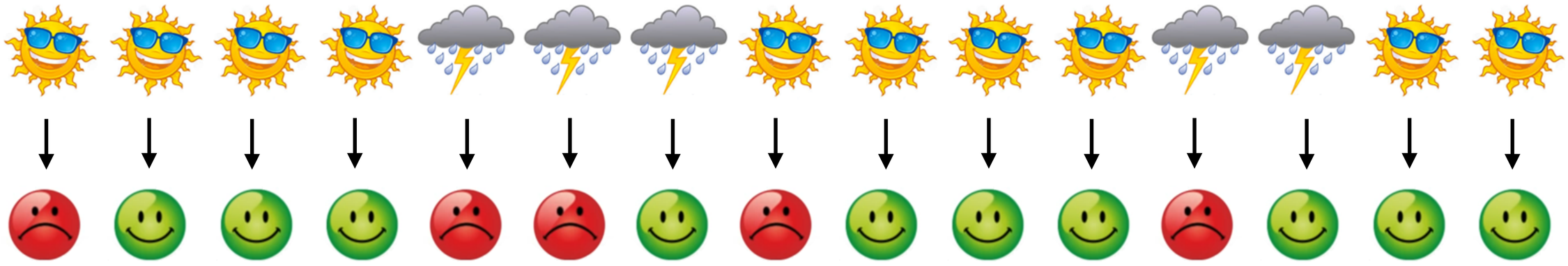


Hidden variable
Weather

Observed variable
Bob's mood

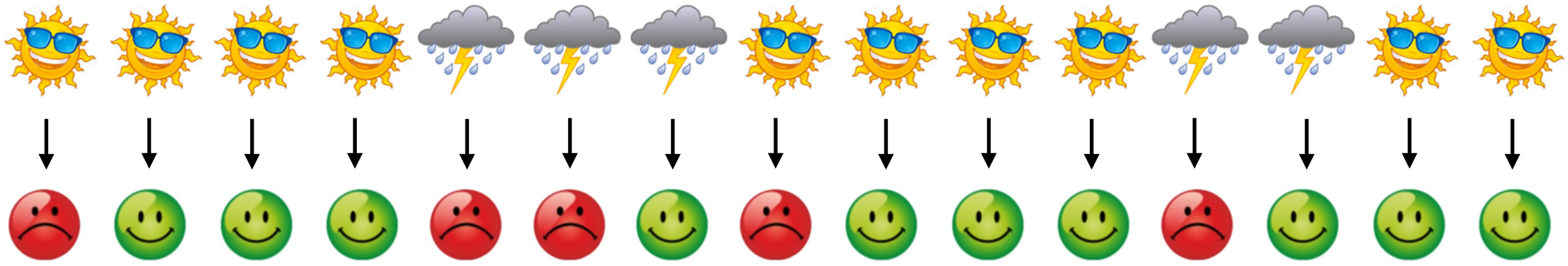
Hidden Markov models

Let’s keep track of Bob’s mood



Hidden Markov models

Let’s keep track of Bob’s mood

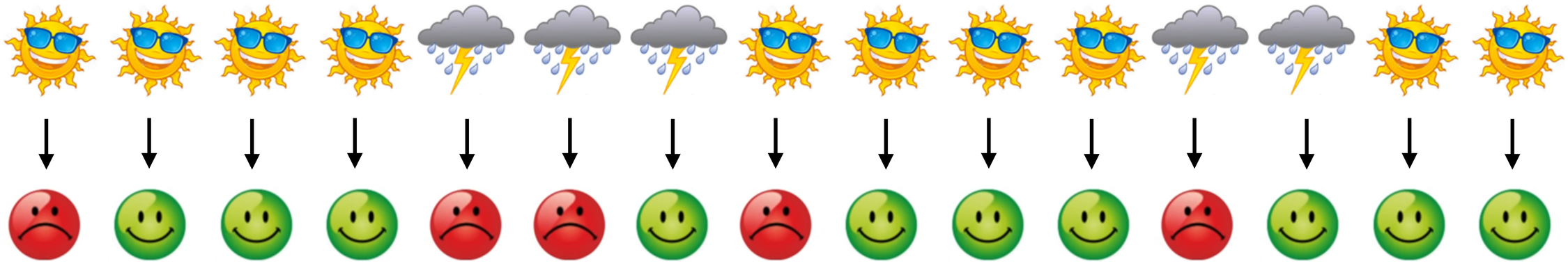


Now we can estimate the **emission probabilities!**
On how many rainy days is Bob Happy and Sad, respectively?
And on how many sunny days is Bob happy and sad, respectively?

Poll

Hidden Markov models

Let’s keep track of Bob’s mood

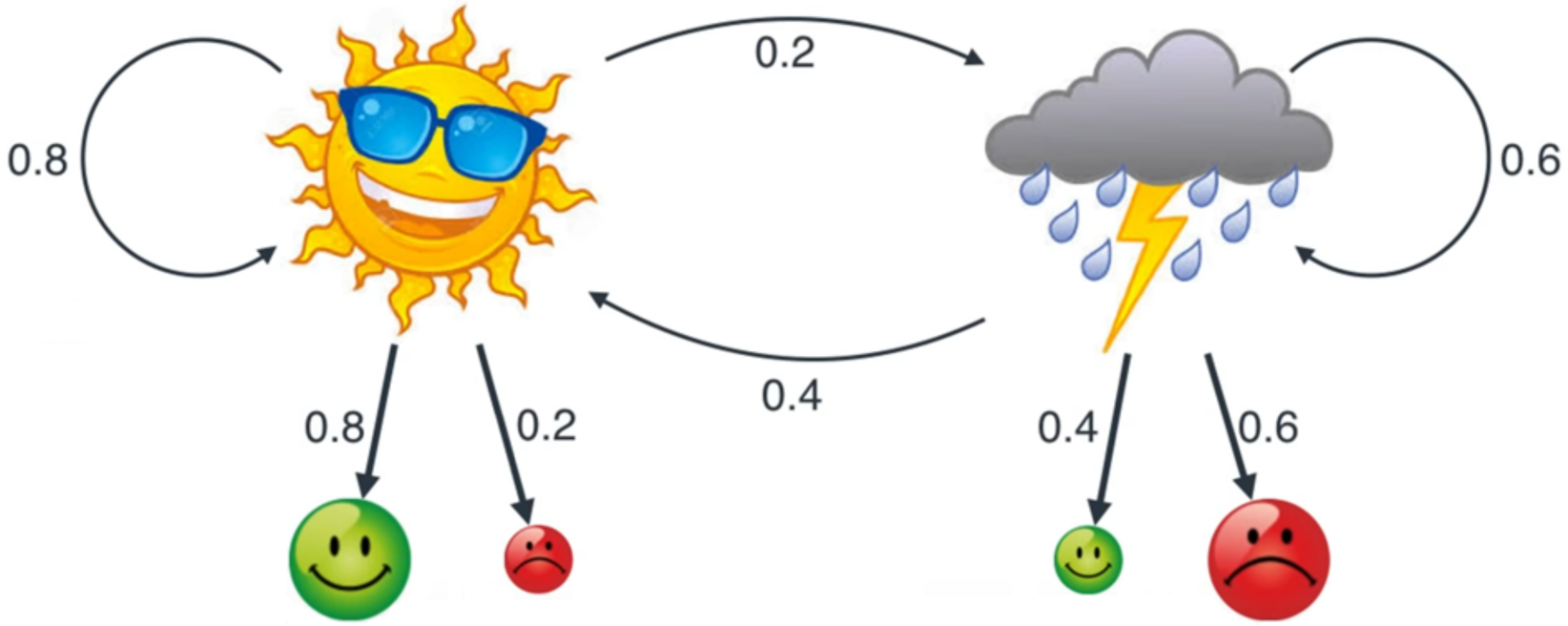


☀	→	☺	8 0.8
☀	→	☹	2 0.2

⛈	→	☺	2 0.4
⛈	→	☹	3 0.6

Only observing the mood

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

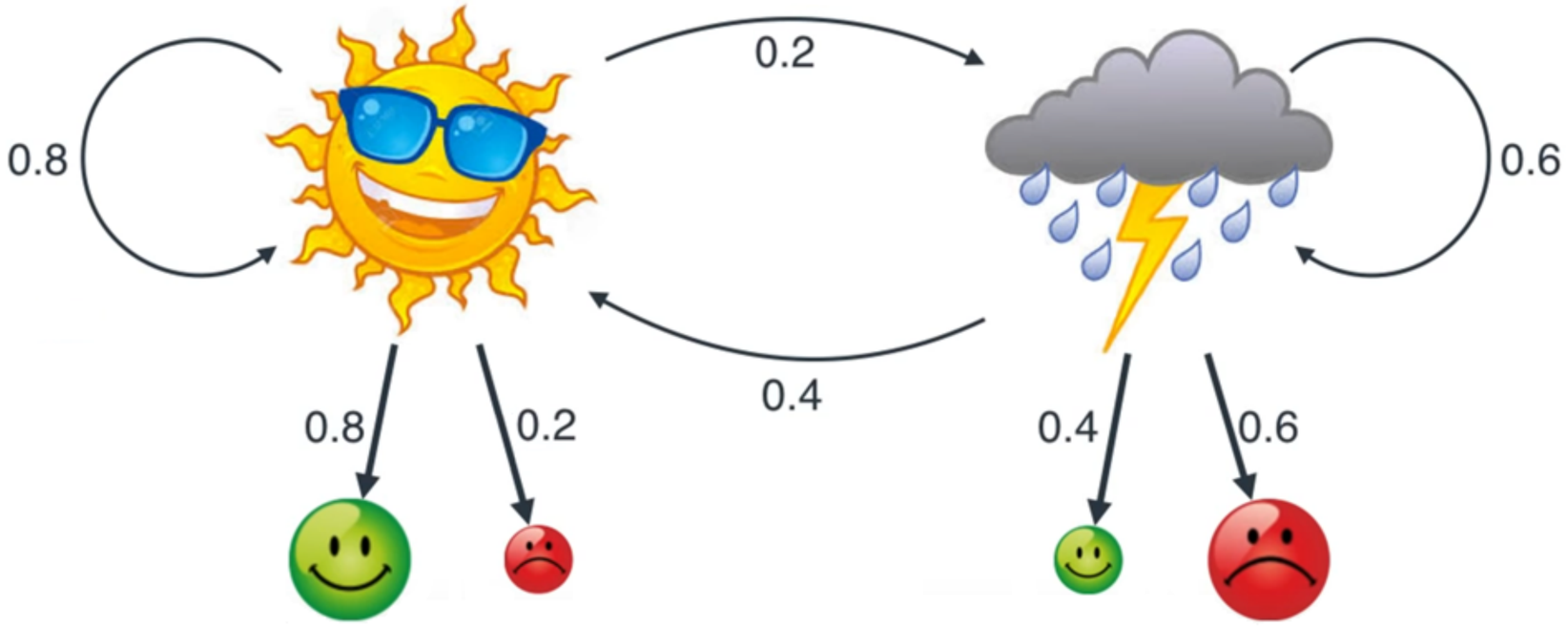


$P(\mathbf{H}) = 2/3$
$P(\mathbf{U}) = 1/3$

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

What is the probability of Bob being happy?

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



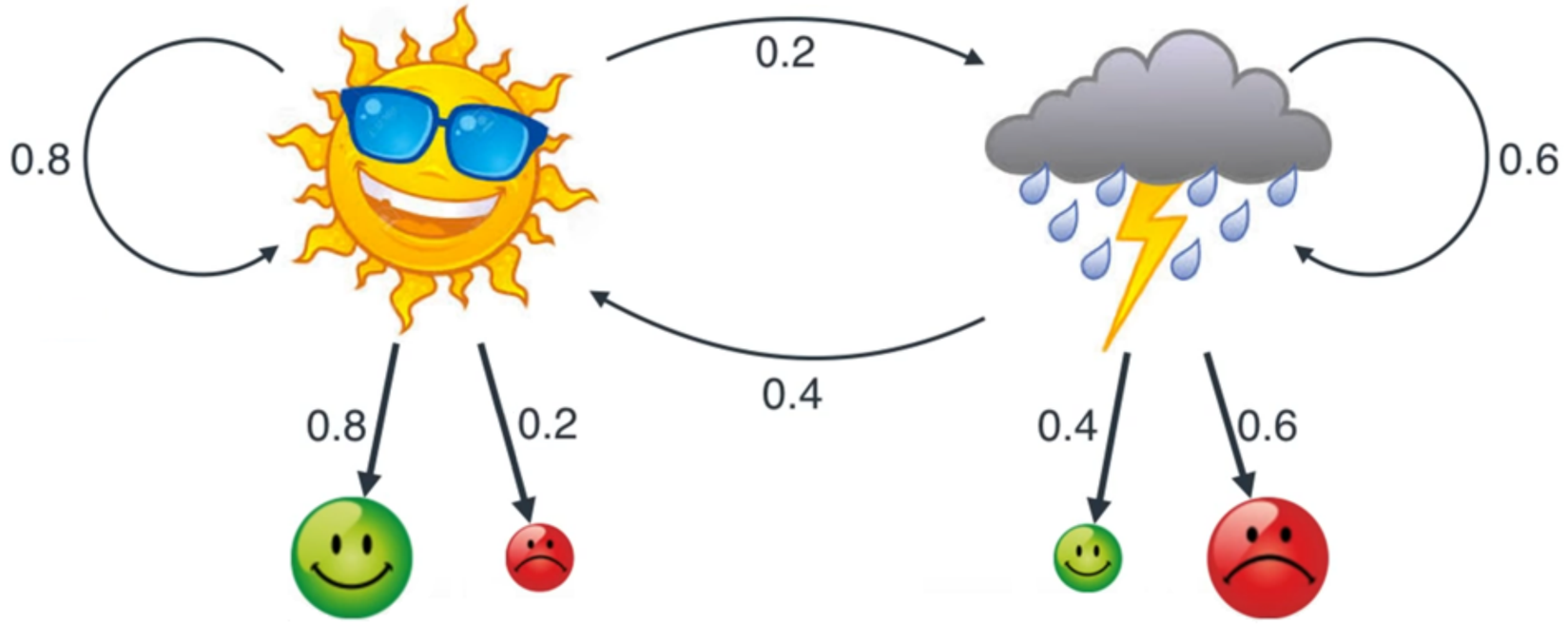
$P(\mathbf{H}) = 2/3$
$P(\mathbf{U}) = 1/3$

The law of total probability:

$$p(H) = p(H \mid S)p(S) + p(H \mid R)p(R) = 0.8 \cdot 2/3 + 0.4 \cdot 1/3 = 2/3$$

What is the probability of sun if Bob's happy?

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



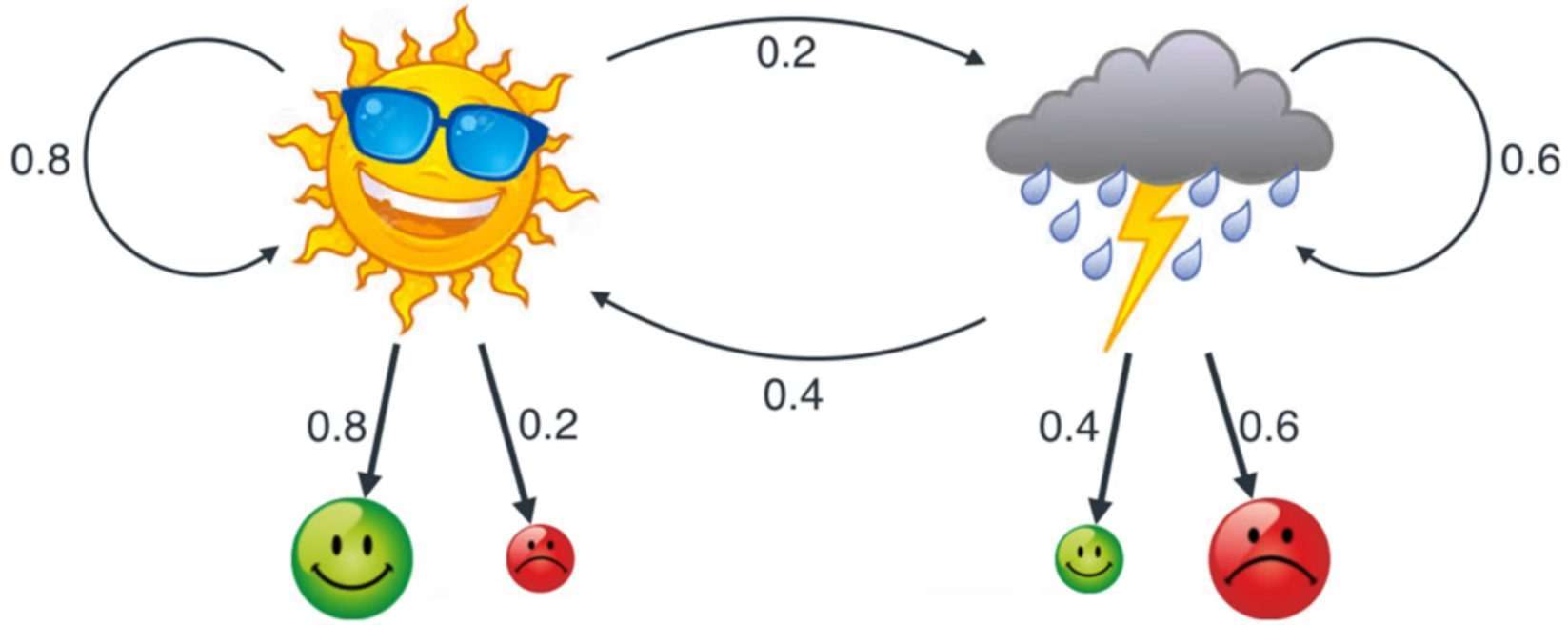
$P(\mathbf{H}) = 2/3$
$P(\mathbf{U}) = 1/3$

How can we compute this?

Poll

What is the probability of sun if Bob's happy?

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

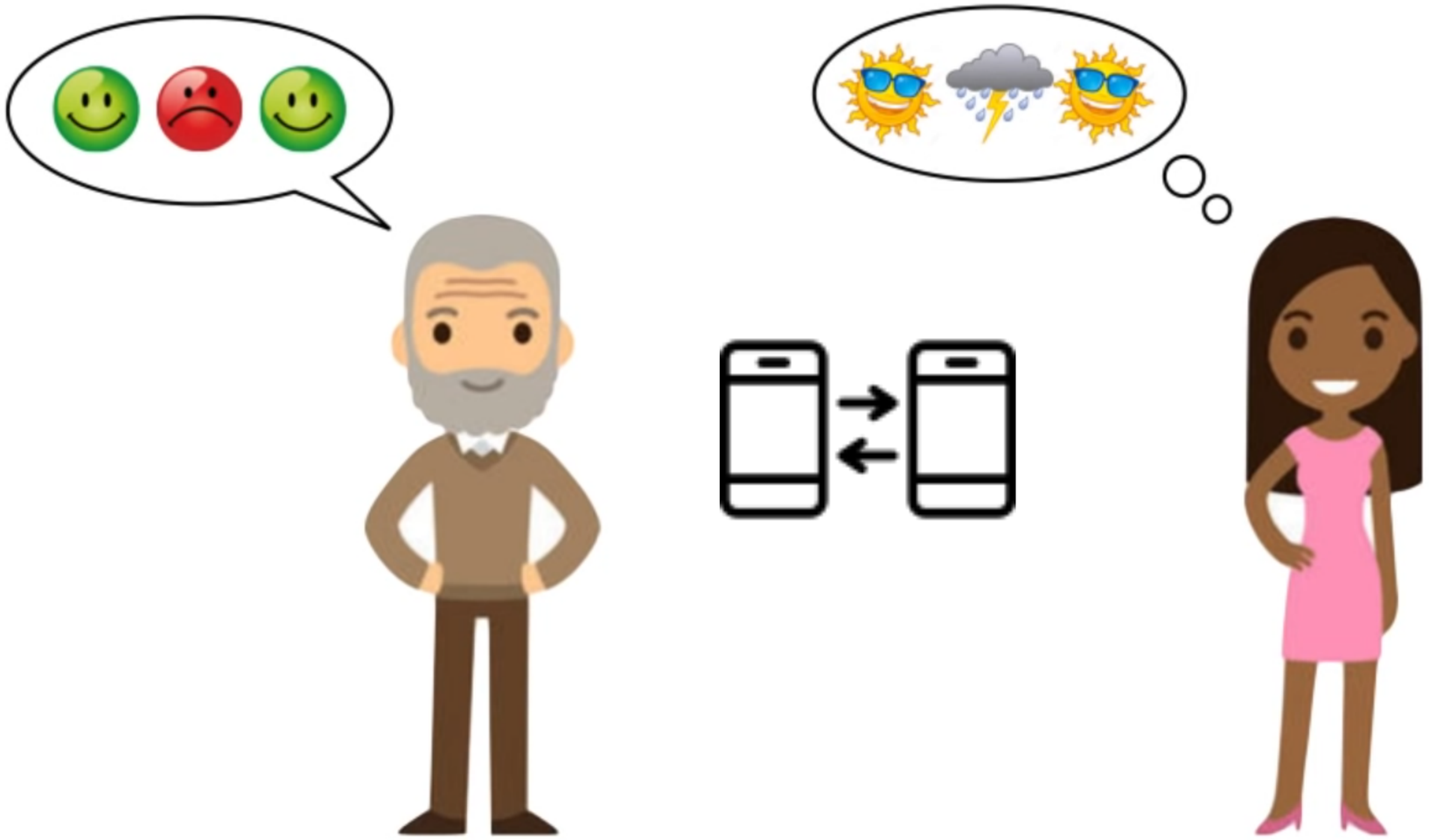


$P(\mathbf{H}) = 2/3$
$P(\mathbf{U}) = 1/3$

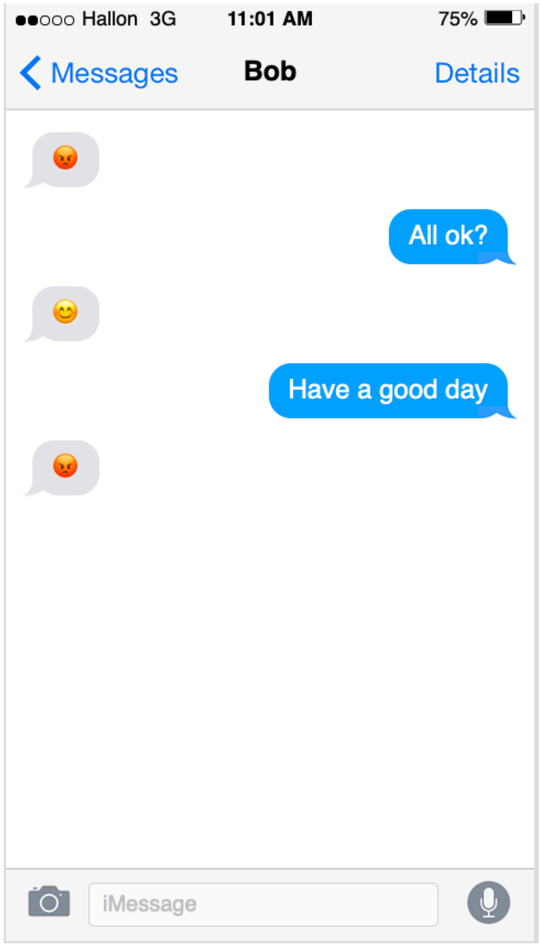
Bayes' rule:

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

Introducing Alice



Alice only observes Bobs mood



Bob *is* sensitive to the change in weather.

The weather is a hidden process.

But Alice is more concerned about the weather

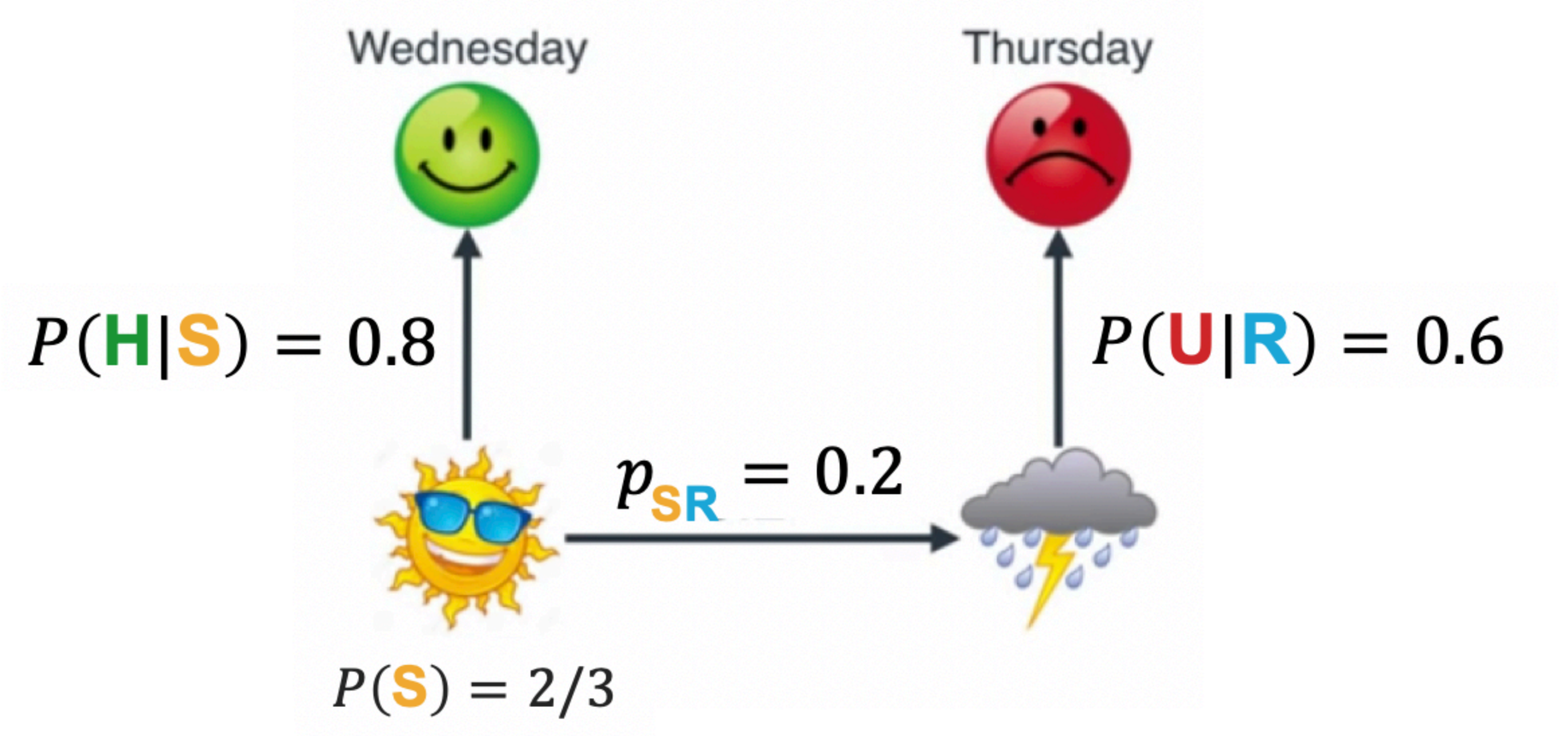
How can Alice predict the most likely sequence of weather using the observations

Wednesday

Thursday

But Alice is more concerned about the weather

How can Alice predict the most likely sequence of weather using the observations



But Alice is more concerned about the weather

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

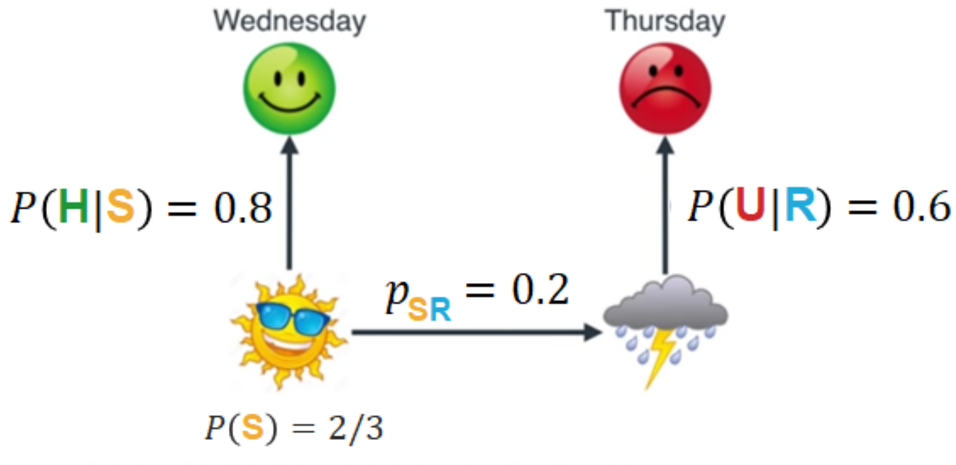
$$= \frac{P(X_1 = \mathbf{S}, X_2 = \mathbf{R}, Y_1 = \mathbf{H}, Y_2 = \mathbf{U})}{P(Y_1 = \mathbf{H}, Y_2 = \mathbf{U})}$$

But Alice is more concerned about the weather

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

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

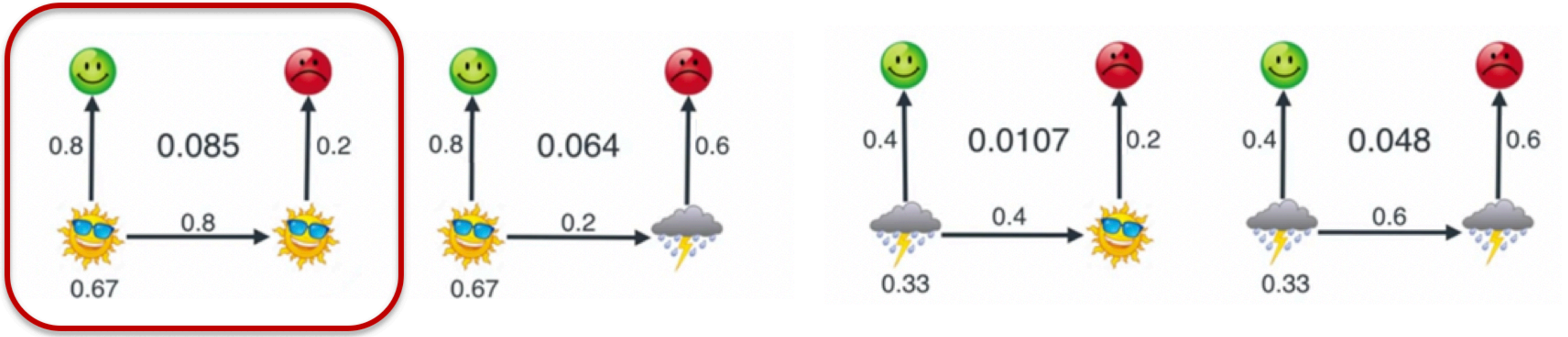
But Alice is more concerned about the weather



$$P(X_1, X_2, Y_1, Y_2) = P(Y_2|X_1, X_2, Y_1)P(X_2|X_1, Y_1)P(Y_1|X_1)P(X_1) =$$

Maximum Likelihood

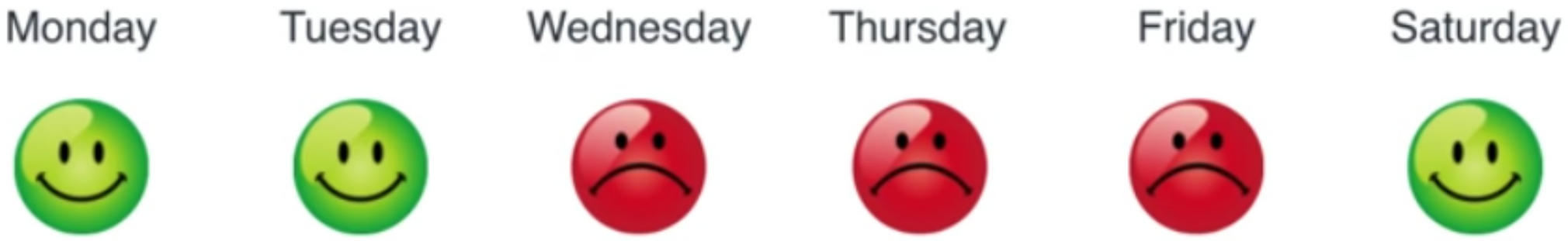
Do the same to all possible weather sequences



The winner is ”sunny” followed by ”sunny”

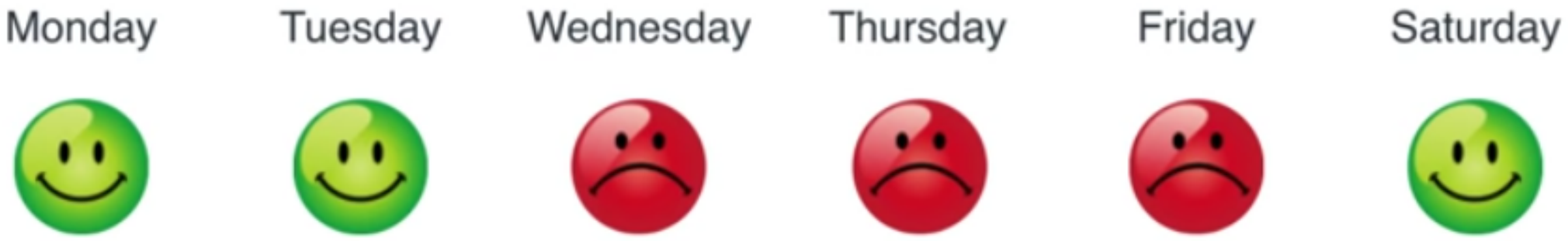
Maximum Likelihood

General problem: estimate the most likely hidden sequence given a set of observations.



Maximum Likelihood

General problem: estimate the most likely hidden sequence given a set of observations.



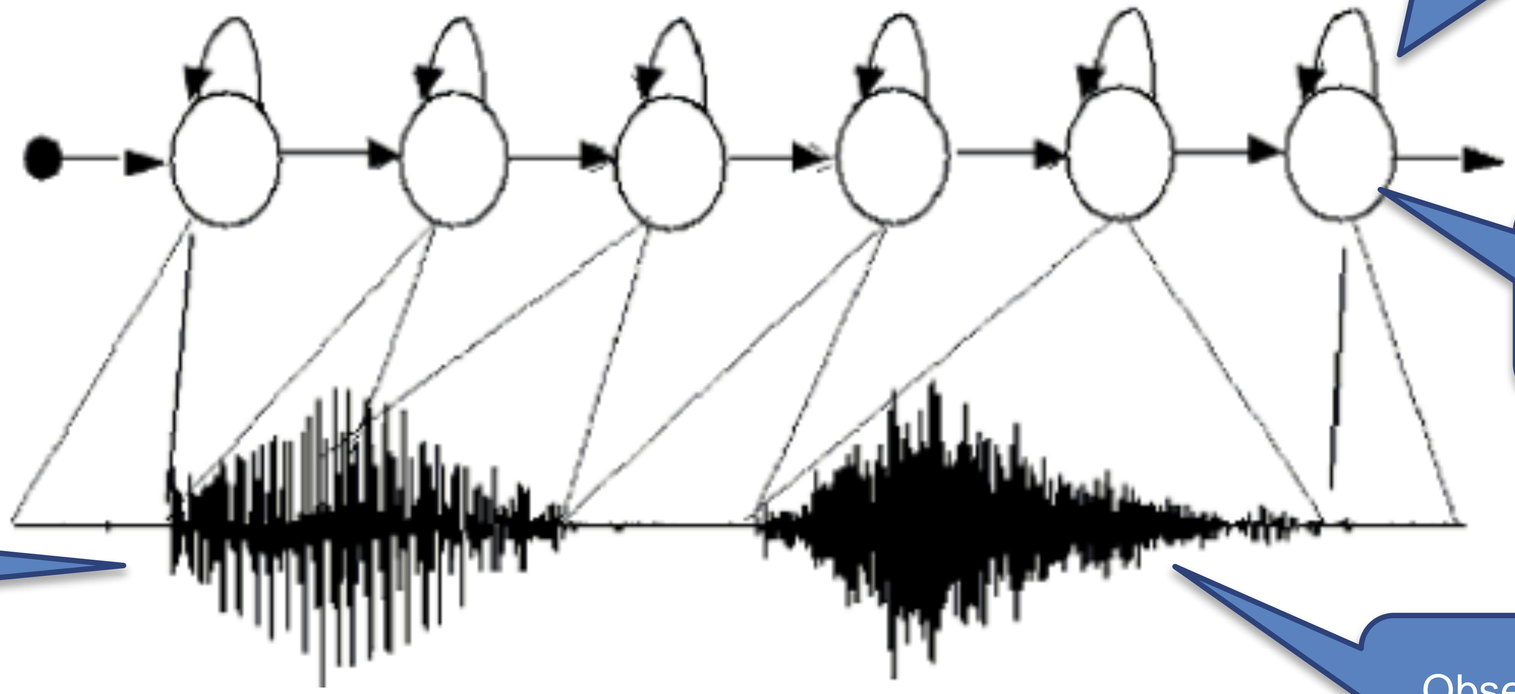
Baum-Welch algorithm — A special case of the *expectation-maximisation* algorithm.

HMM applications

Speech recognition

HMM applications

Speech recognition



Look for the most likely sequence of phonemes = word

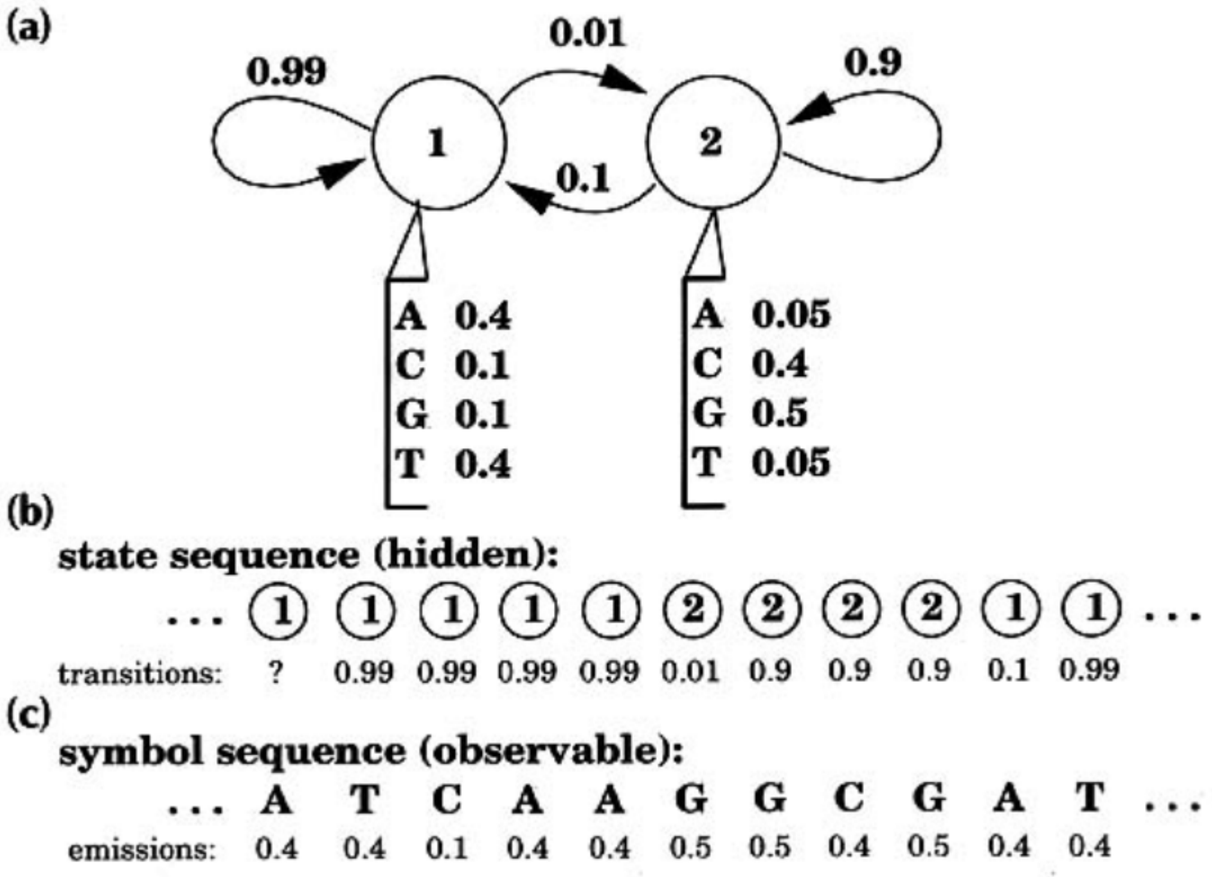
Hidden states = phonemes

Soundwaves from speech

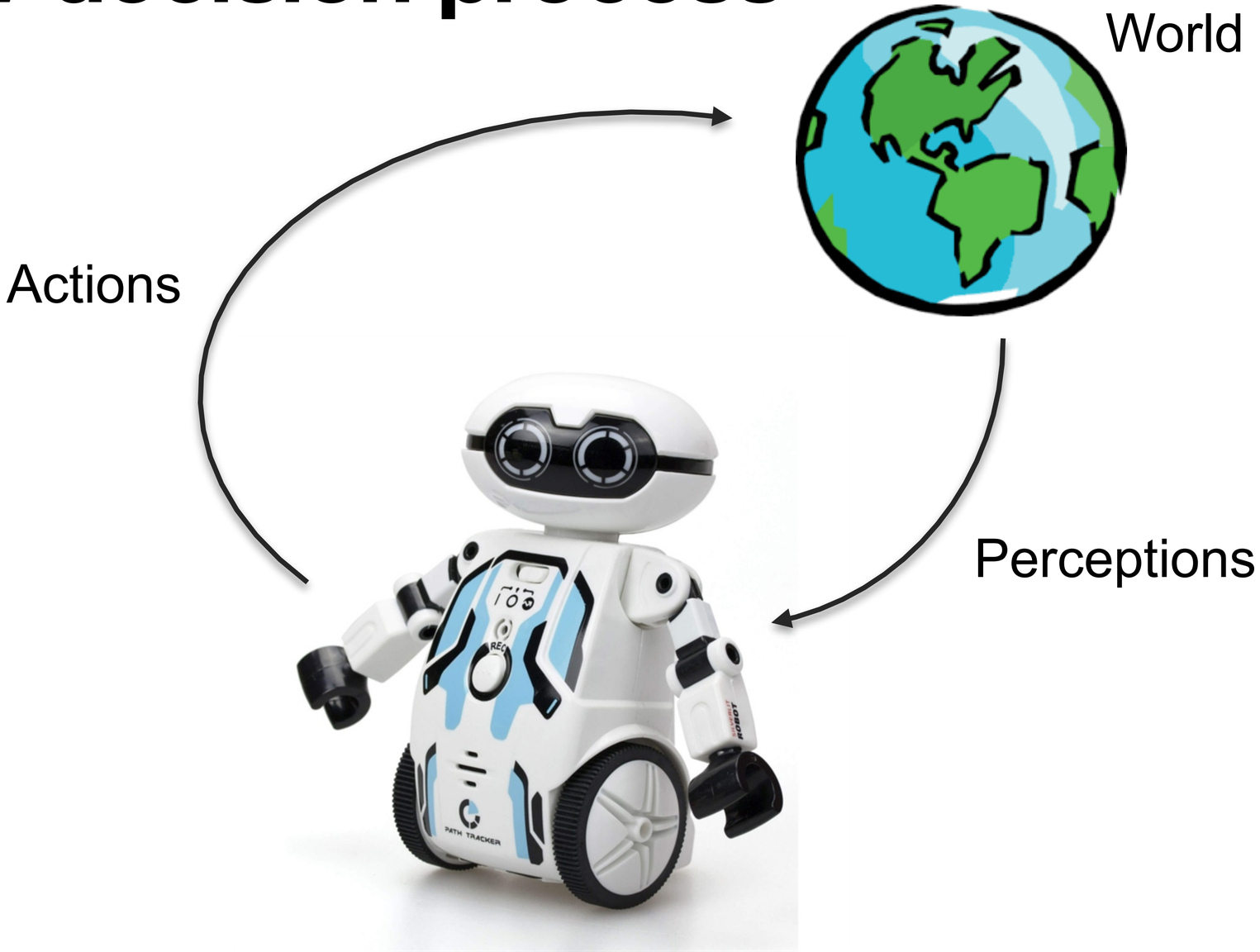
Observed sequence = sound wave properties

HMM applications

DNA sequence



Markov decision process

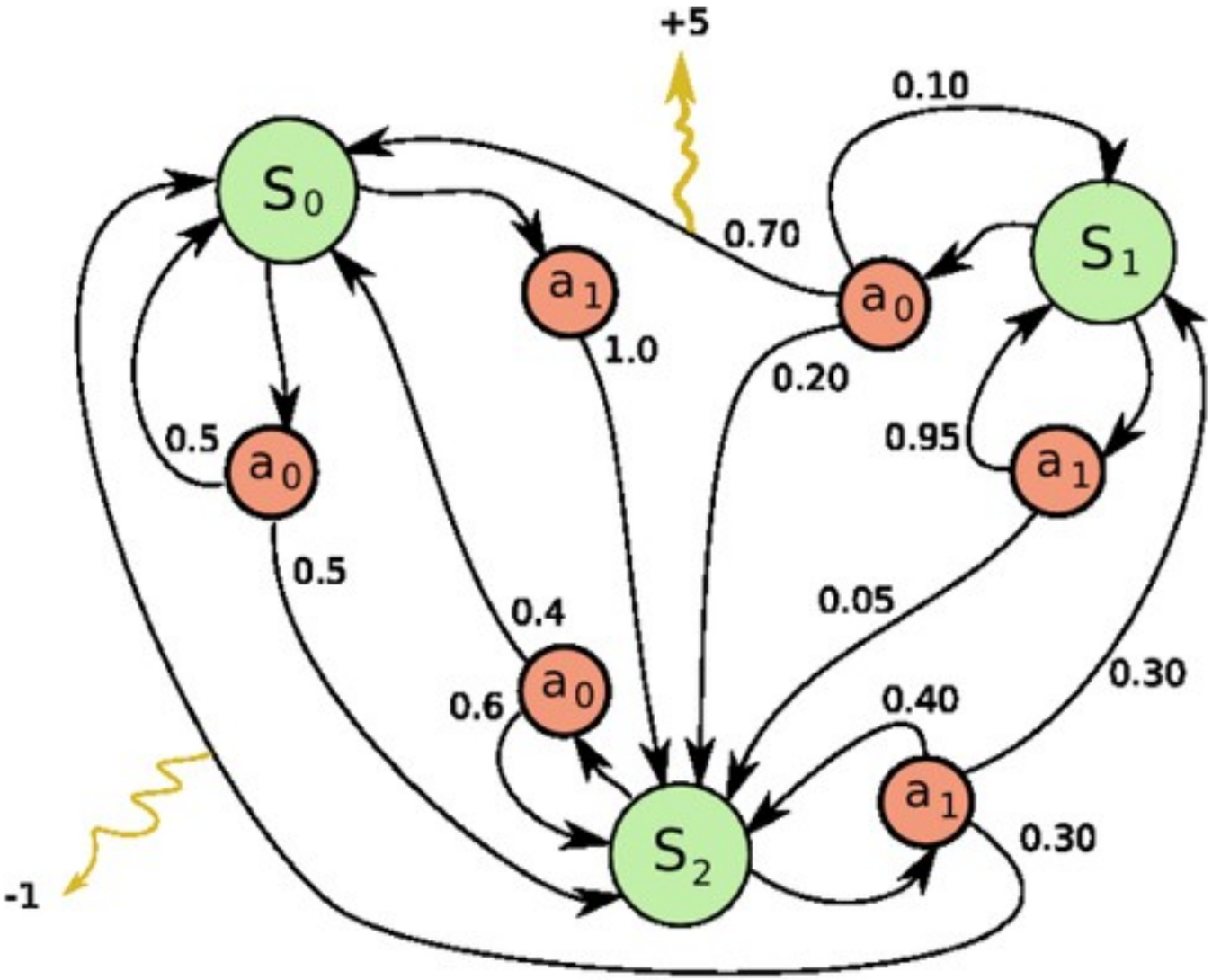


Markov decision process

A **Markov Decision Process (MDP)** is a Markov Chain *augmented* 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

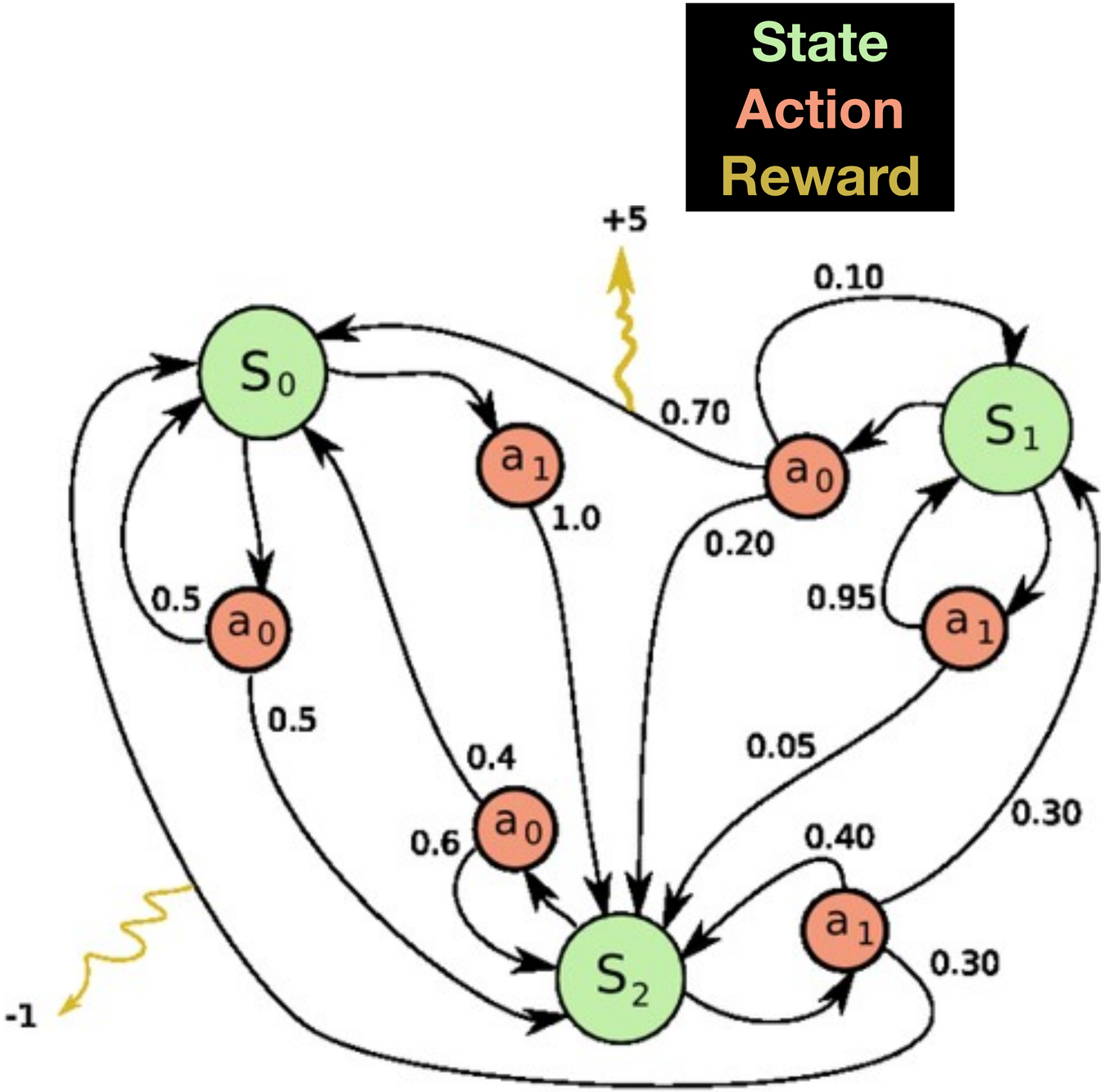
Solving MDPs



State
Action
Reward

Solving MDPs

1. Find a **policy** that gives the maximum reward in the long run
2. A **policy** is a set of *rules* for which action to take in each state. E.g.
   1. In state s_0: do action a_1
   2. In state s_1: do action a_0
   3. In state s_2: do action a_1



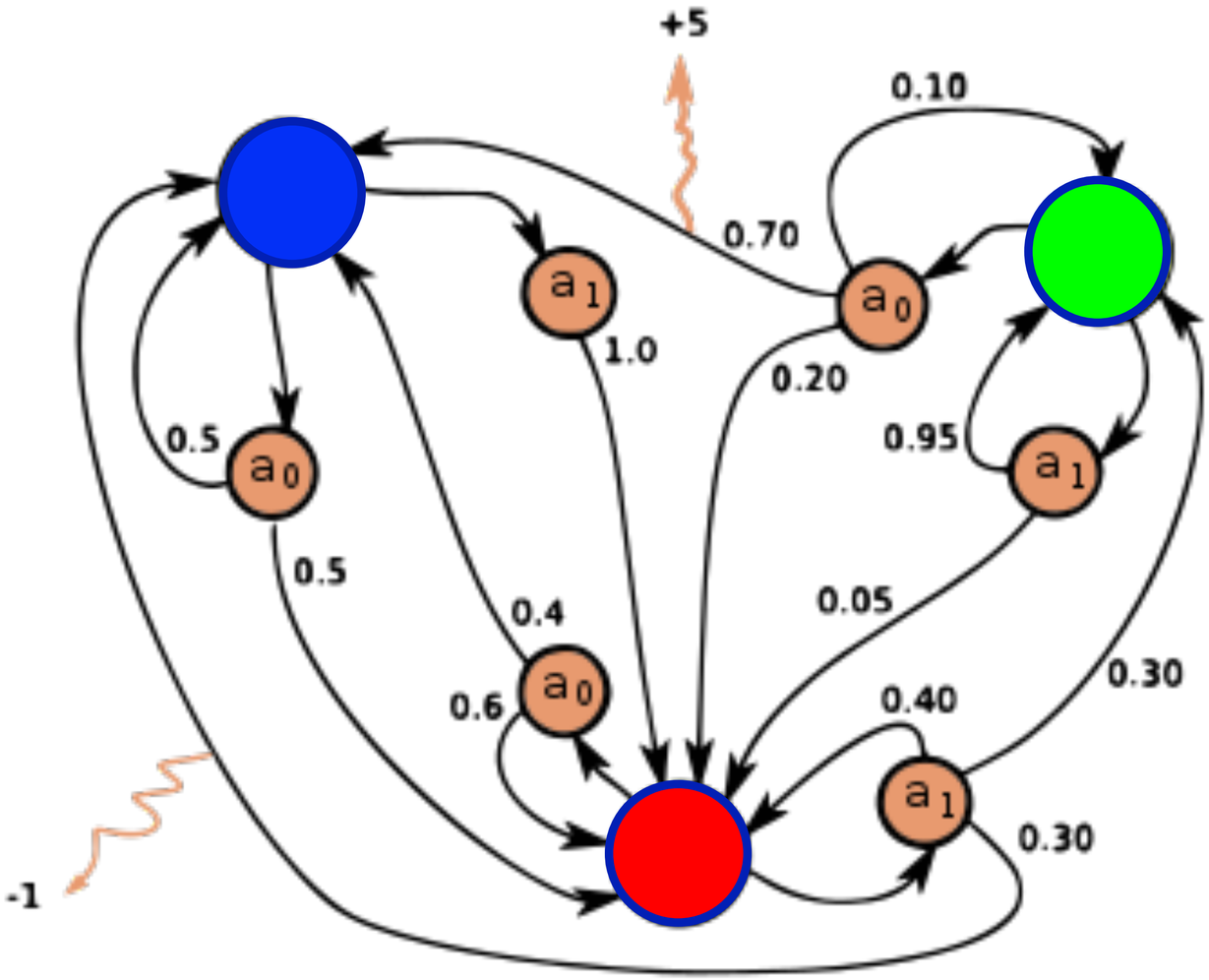
MDPs a case of Reinforcement Learning

1. An **MDP** describes the *interaction* between the **agent** and **the environment (world)**
2. Usually the **agent** does <u>not</u> have *full knowledge* of the **environment**
3. The **agent** *interacts* with the **environment** to learn from it.

Example: Psychological test

1. A **color** is shown on the screen (**current state**)
2. Select between two or more **actions** (**buttons**)
3. Candies are given through a **dispenser** (**reward**)
4. How much **candy** can you collect in 5 minutes? (*maximize reward*)

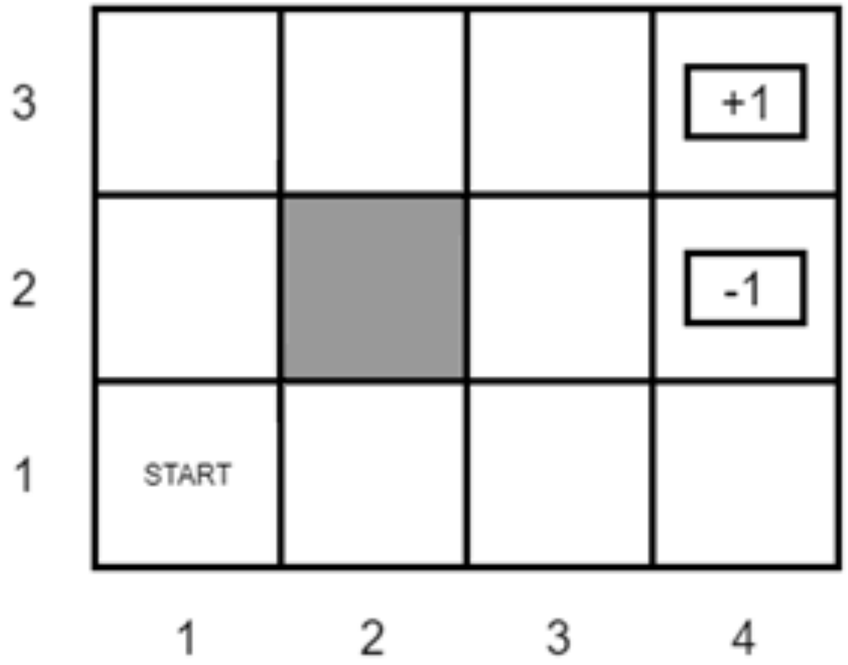
Example: Psychological test

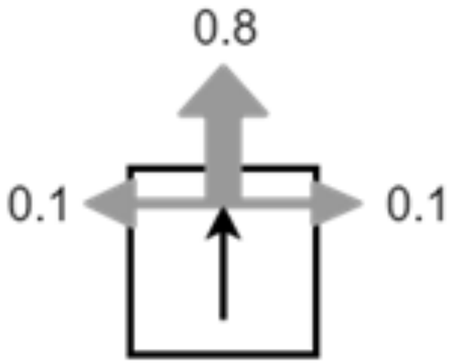


Example: grid-world

This robot world can be viewed as an MDP.

- **States**: 11 squares
- **Actions**: 4 moves
  - $p(\text{correct move} \,|\, \text{action}) = 0.8$
  - $p(\text{wrong move} \,|\, \text{action}) = 0.1 + 0.1$
  - Rewards: 0, -1, or +1, as shown

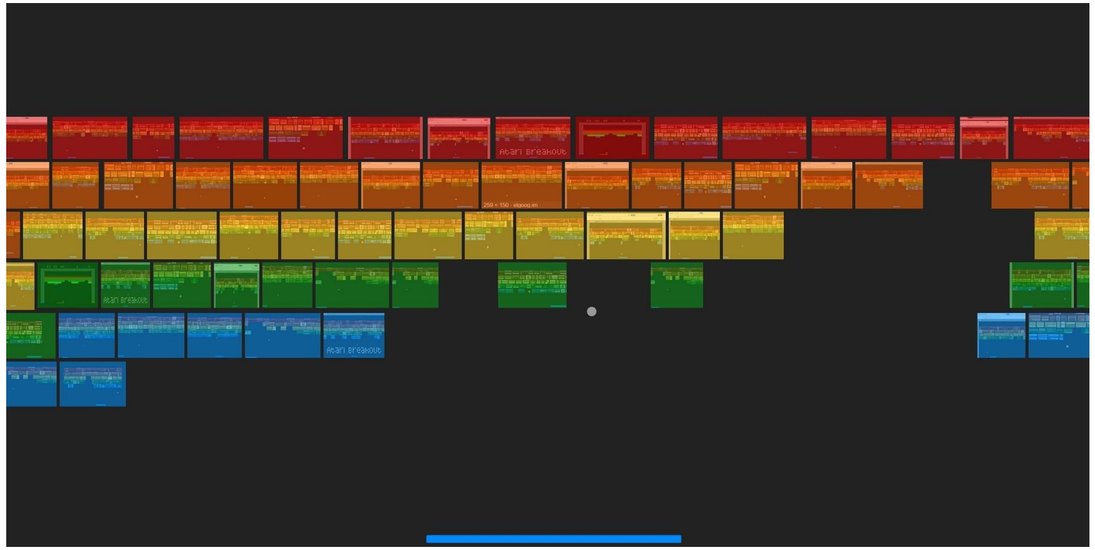




Example: break-out

Breakout can be viewed as an MDP.

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

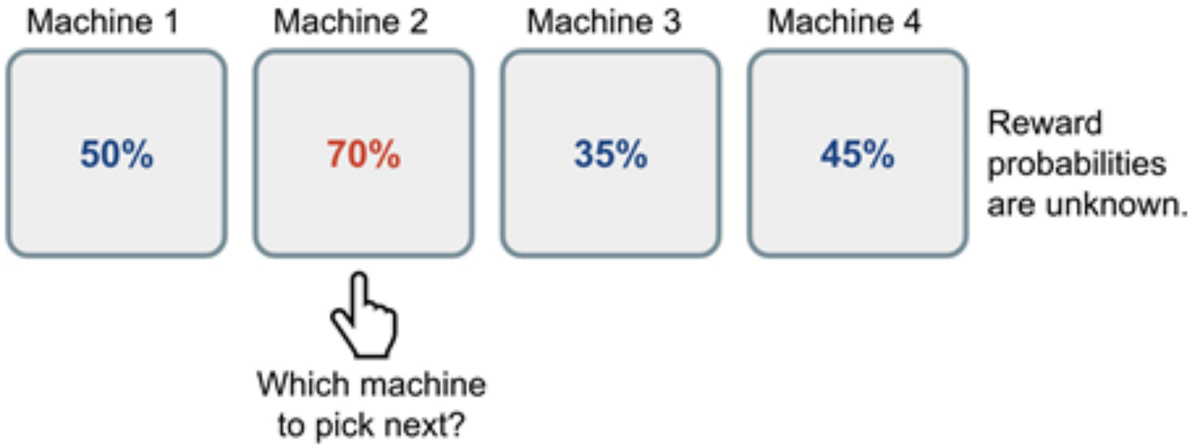


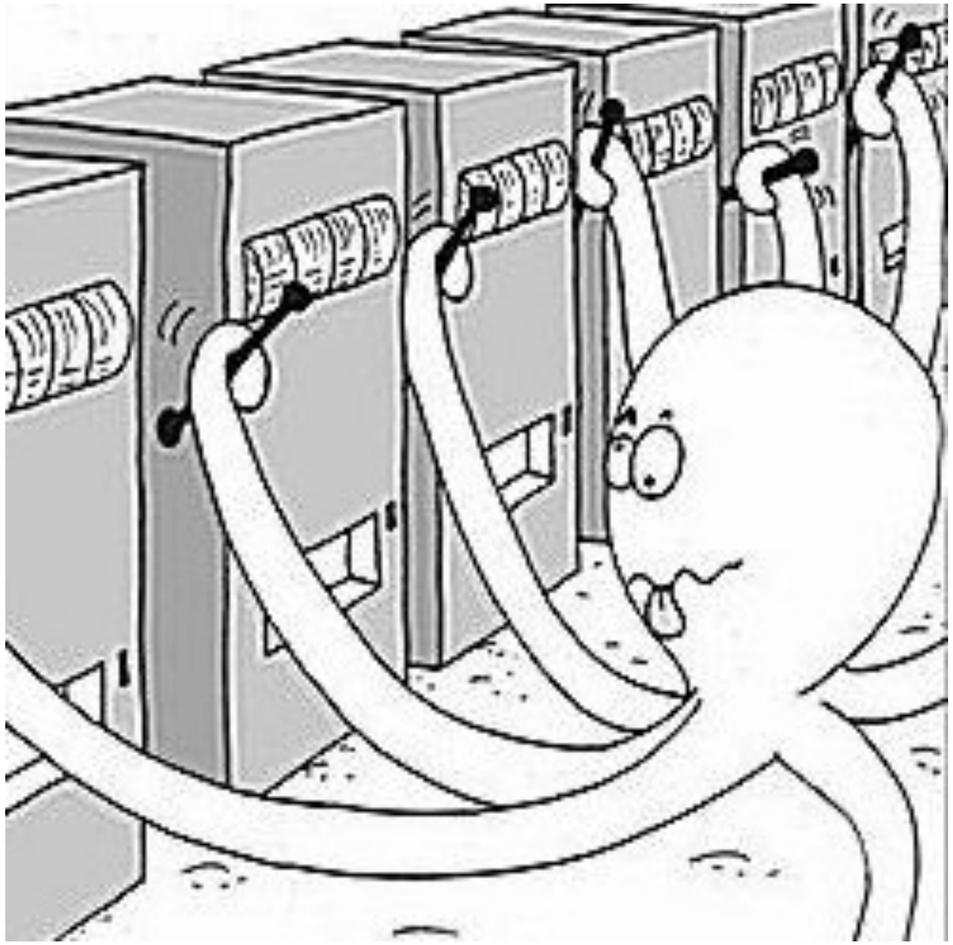
Example: Bandit problems

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

Example: Bandit problems

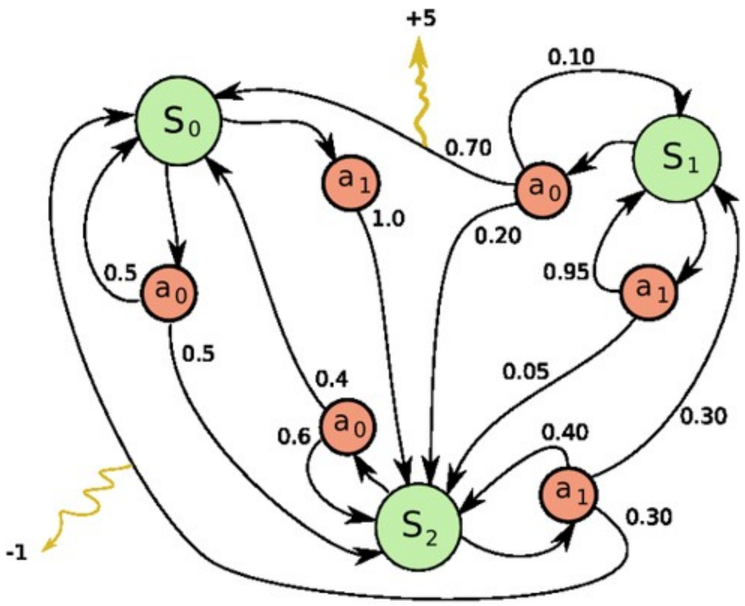
- *A multi-armed bandit problem is an MDP*
  - *Just one* ***state***
  - ***Action****: which slot to pick next*
  - ***Reward****: the winnings*
- ***Goal****: to maximize the reward*



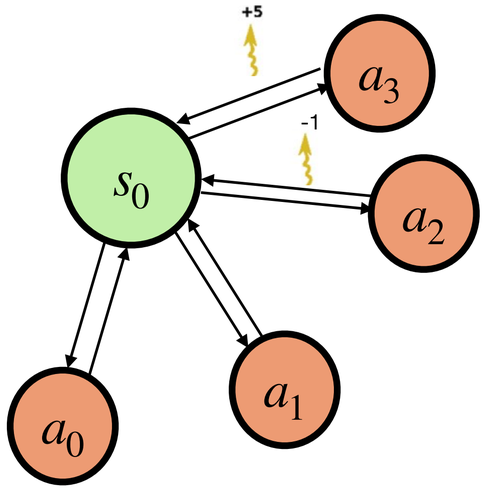


Poll: which MDPs represent a Multi-armed bandit problem?

A



B



Reinforcement learning in Biology

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

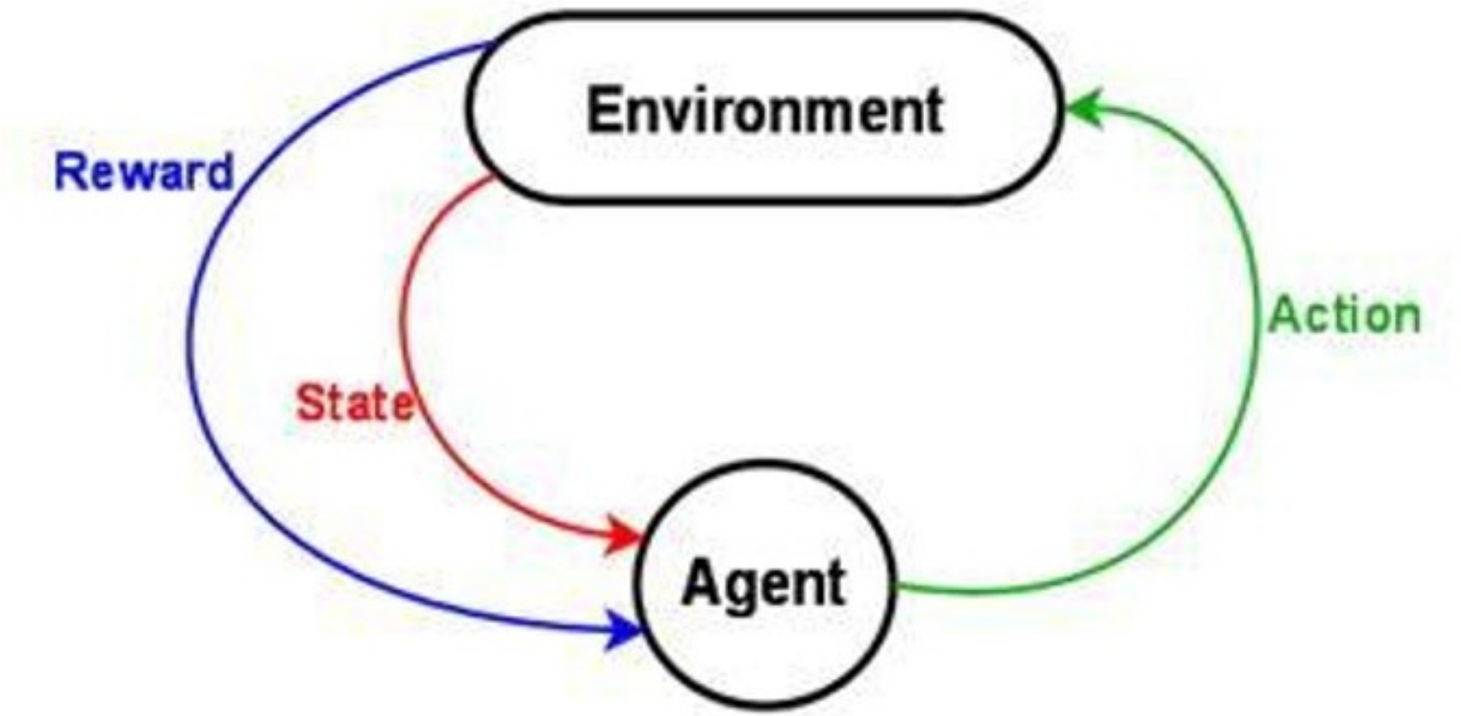
Reinforcement learning in Biology

Dogs can learn advanced tricks from treats

Reinforcement learning in Biology

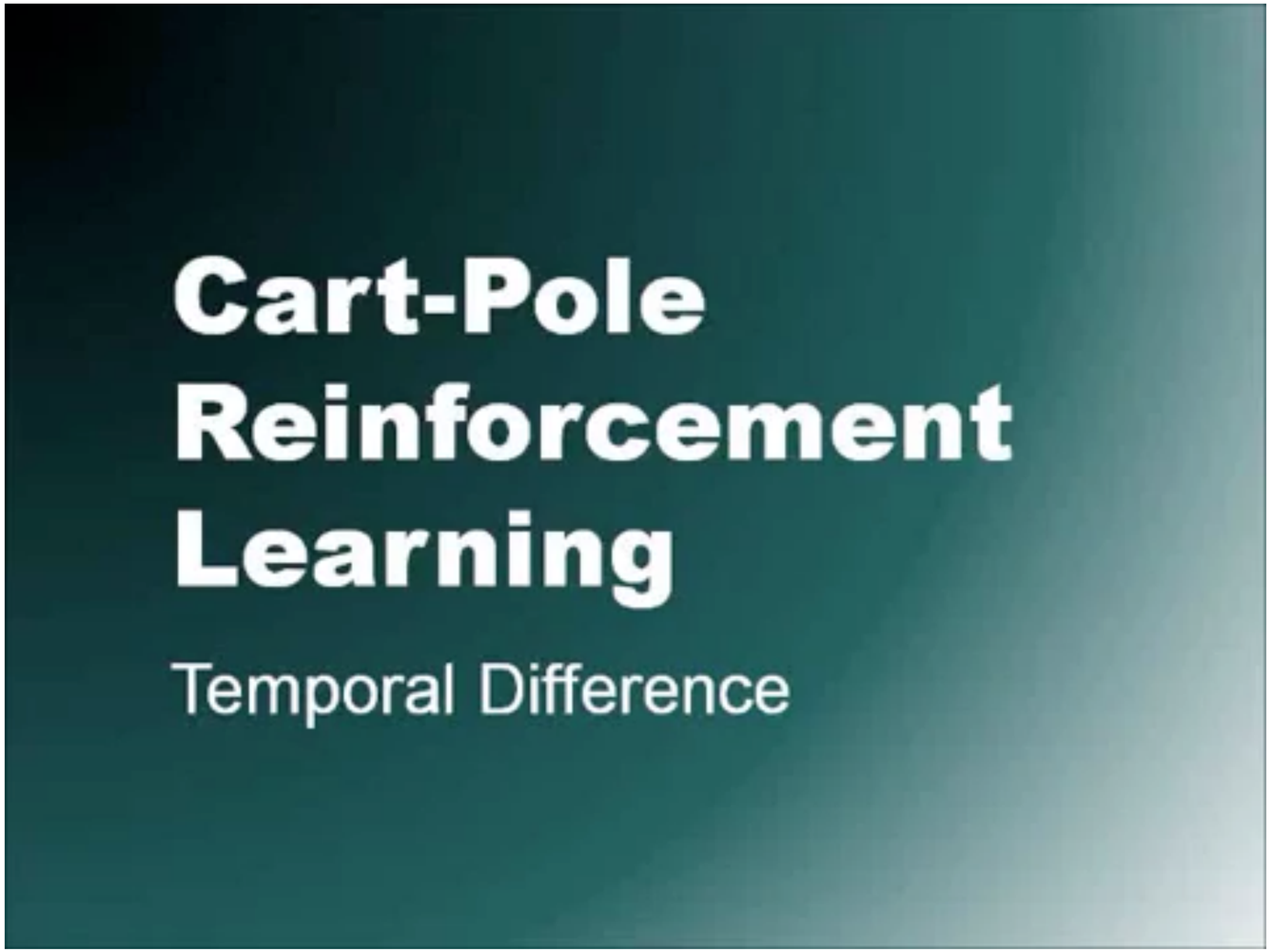
… and so can chicken.

Reinforcement learning in Computer Science



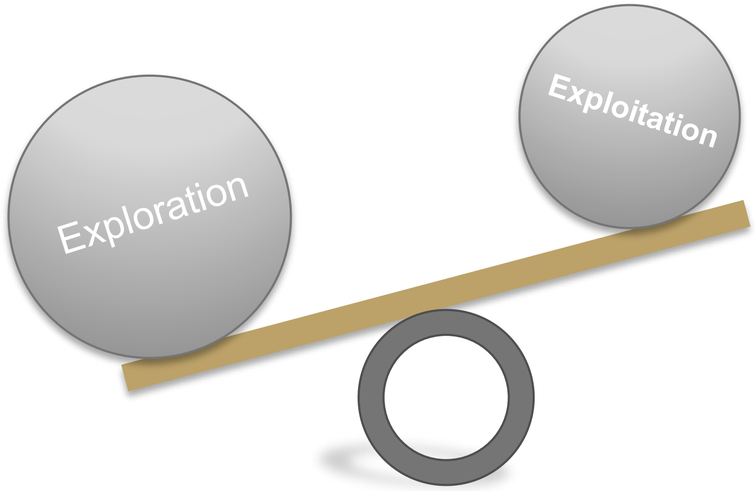
Agents take *actions* in the **environment** based on the current *state* in order to maximise *reward*

Pole balancing



Exploration-exploitation

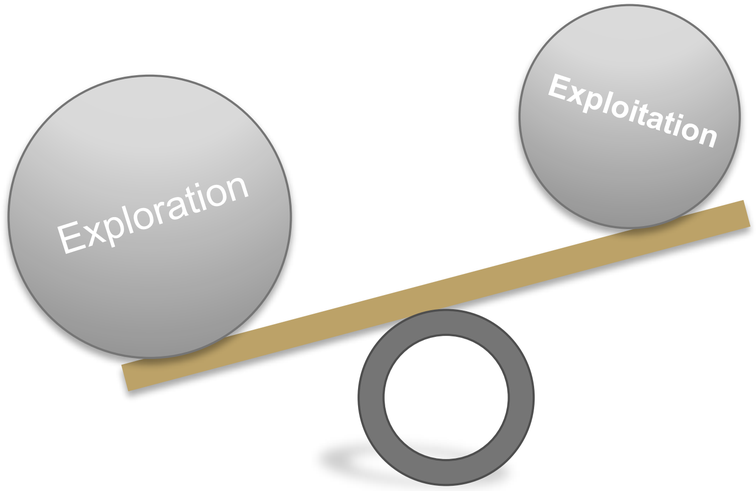
- **Explore**: taking a random action
- **Exploit**: taking whatever action seems most rewarding at the present moment while forgetting about future consequences (the greedy action).



ϵ-greedy algorithms

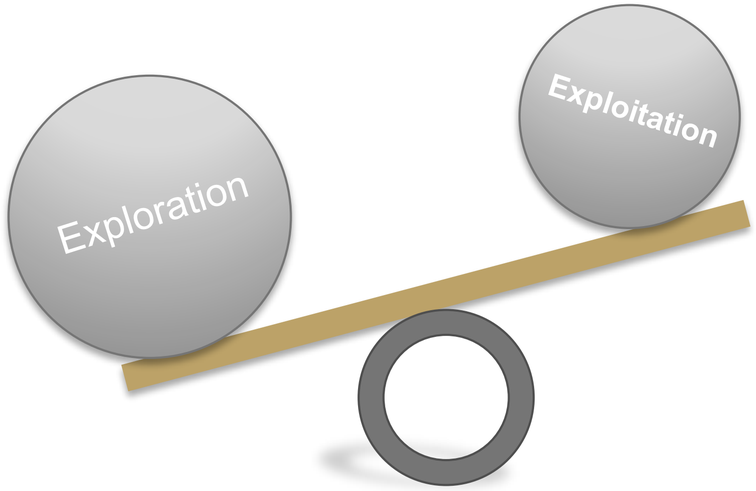
At each interaction:

- **Explore** with probability $1 - \epsilon$
- **Exploit** with probability ϵ
- The **threshold**, $0 \leq \epsilon \leq 1$: choice controls the exploration-exploitation balance.



Exploration-Exploitation trade-off

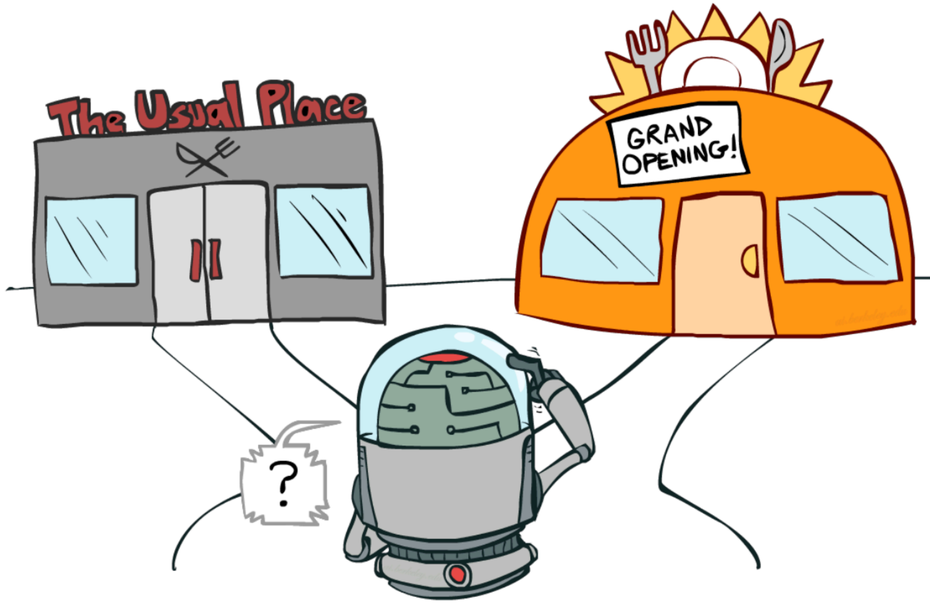
- If we **exploit** all the time we might miss large parts of the state space (end up in a *local optimum*)
- If we **explore** all the time, we don't learn from experience => no improvement
- Striking a *balance* of **exploitation** and **exploration** is needed for optimal performance



Exploration-Exploitation trade-off

Examples:

- 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?

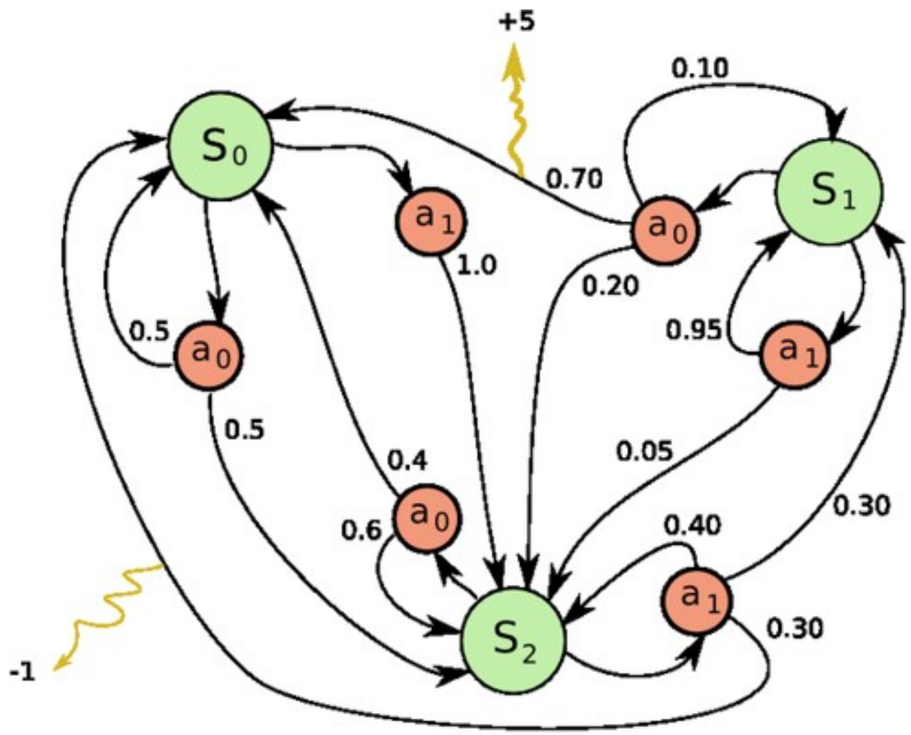


Accumulated discounted reward

- Early **reward** may be better than later.
- This can be controlled via a <u>discount factor</u> $\gamma \in [0,1]$
- The accumulated discounted reward becomes

$$R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots$$

- If $\gamma < 1$ the sum of rewards <u>converges</u> even if the decision horizon is infinite.



Q-learning

- In **Q-learning** (**Q** = ”*quality*”) we assign a value $Q(s, a)$ to each initial state-action pair (s, a).
- We want to find a **policy** (a set of state-action decision rules) $\pi(s) = a$ such that

$$\pi(s) = \text{argmax}_a Q(s, a)$$

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

Q-learning

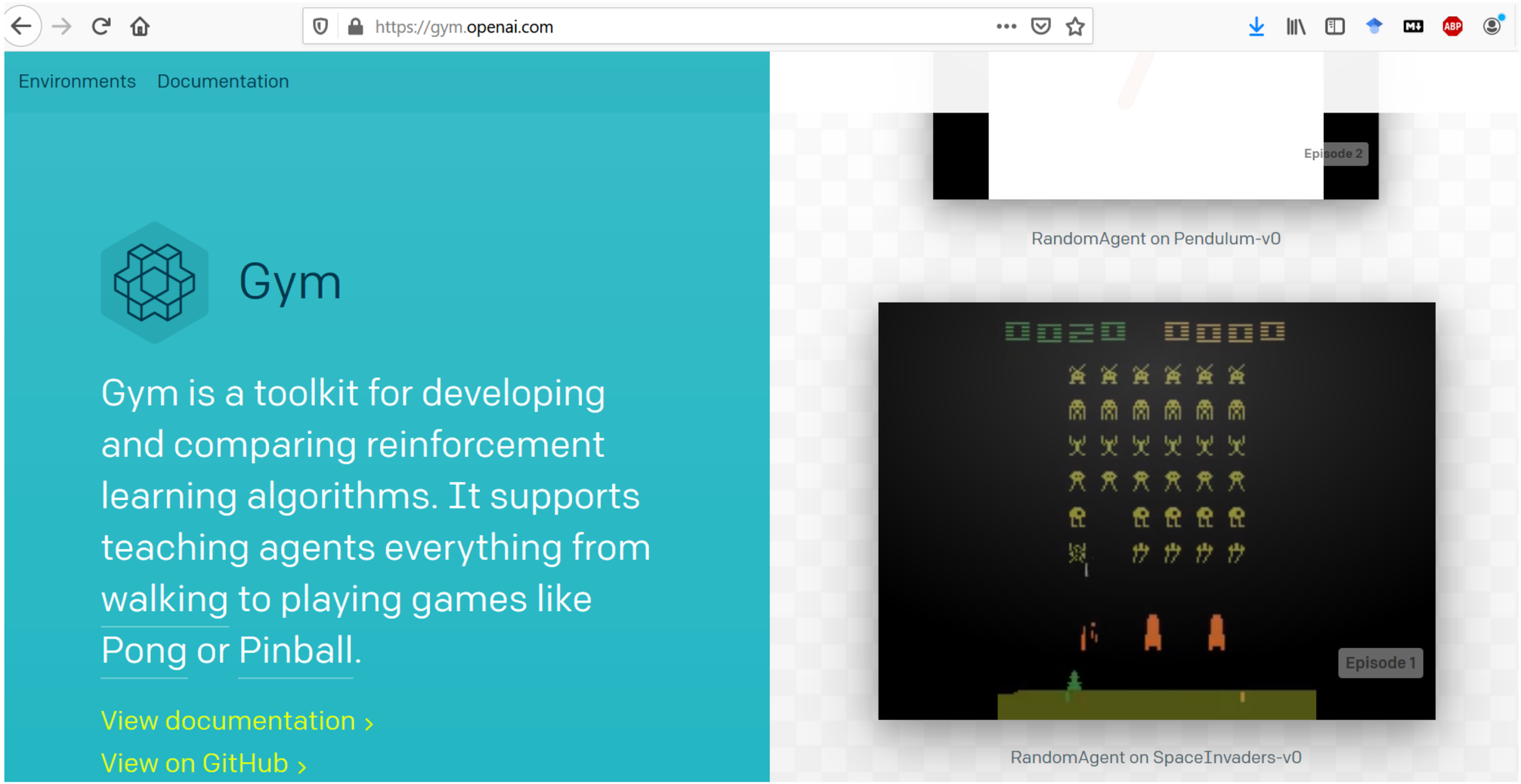
- **Q-learning** is an iterative procedure where an update at time t is computed using the Bellman equation:

$$Q^{new}(s_t, a_t) = \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$

Q-learning

- Q-learning may eventually find the **optimal policy** for any finite MDP.
- Convergence <u>may be very slow</u>.
- Moreover, table-based Q-learning has limited scalability.

OpenAI gym


https://gym.openai.com
Environments
Documentation
Gym
Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.
View documentation ›
View on GitHub ›
Episode 2
RandomAgent on Pendulum-v0
Episode 1
RandomAgent on SpaceInvaders-v0