

# Lecture 9

Markov models

# Topics

- Markov chains
- Hidden Markov models
- Markov Decision Processes
- Reinforcement learning in biology
- Reinforcement learning in computer science
- Q-learning

# Andrey Markov

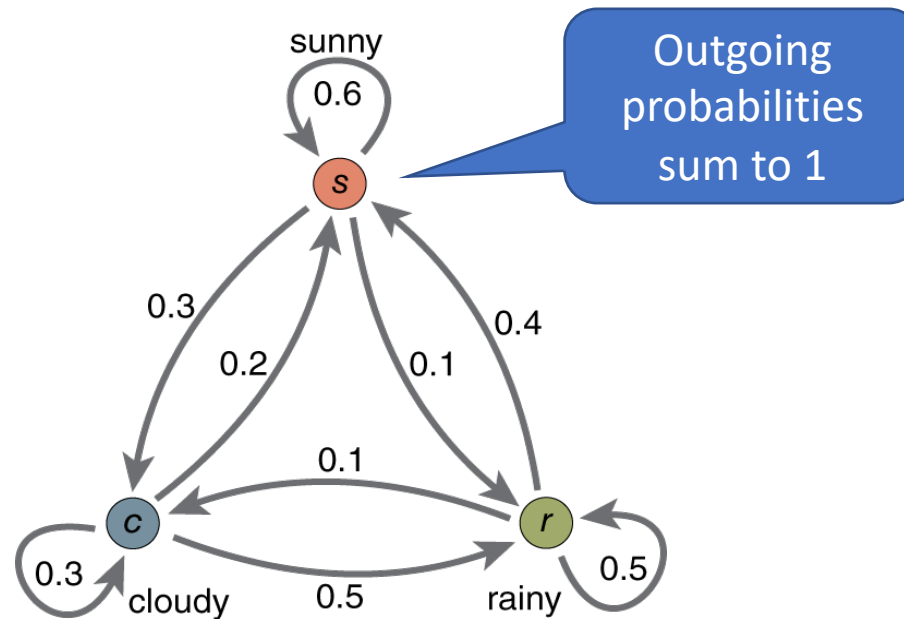
1856–1922



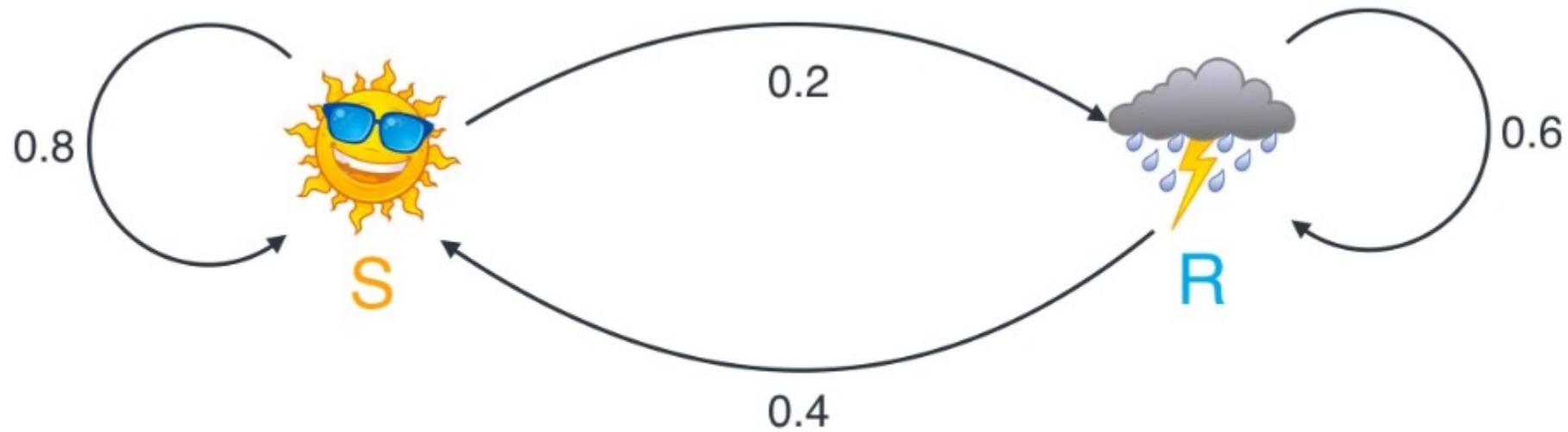
# Markov chains

# Markov chain

A *Markov chain* is a finite directed graph, whose edges are labeled with *transition probabilities*.



# Example of a Markov chain



Two states: sunny and rainy. Transition probabilities shown.


# Starting with data



Suppose we are given this weather data for 15 days. A sequence of two states: sunny and rainy.

# Finding state probabilities from data



	10	$\frac{2}{3}$
	5	$\frac{1}{3}$

Calculate the probability of sunshine and rain directly from the data



# Finding transition probabilities from data



How many sunny days  
were followed by a  
sunny day?



8

0.8



2

0.2



2

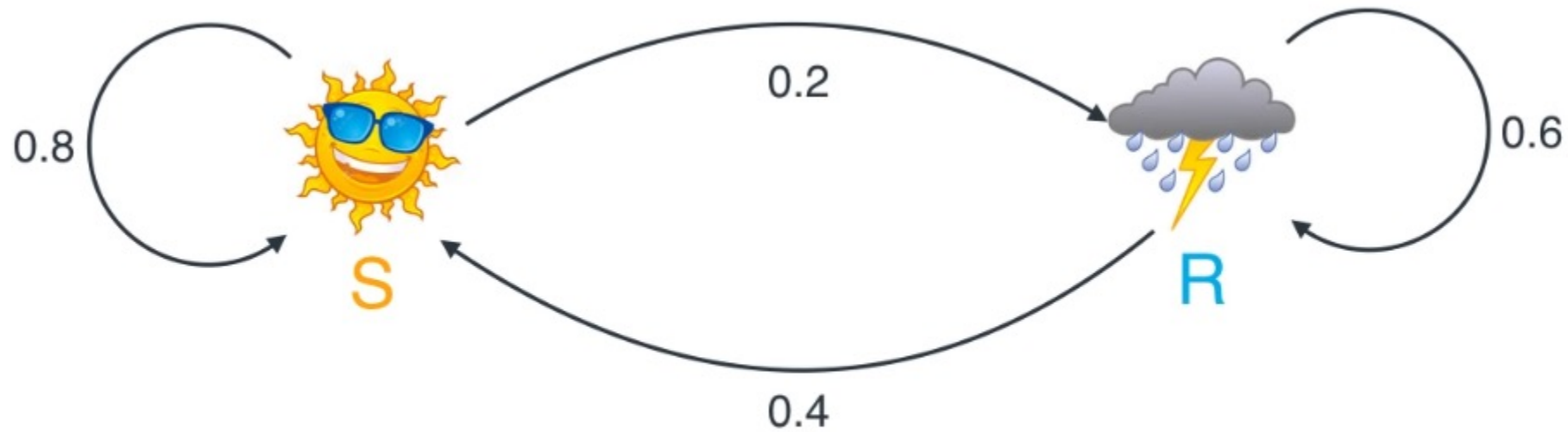
0.4



3

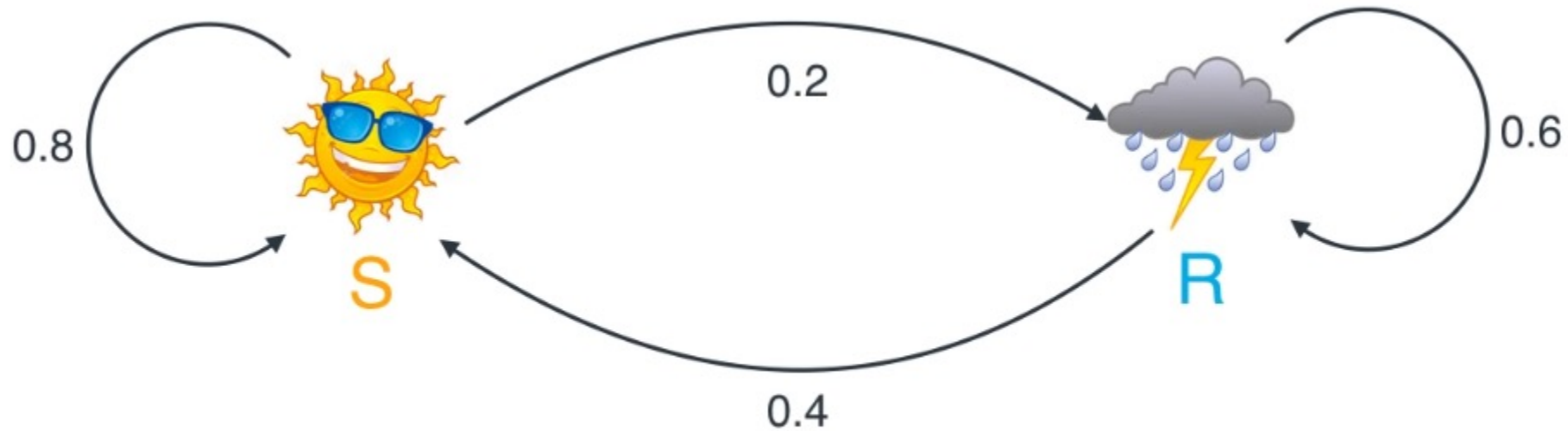
0.6

# Finding the Markov chain from the data



Now that we have the transition probabilities we have the entire Markov chain.

# Finding state probabilities from a Markov chain



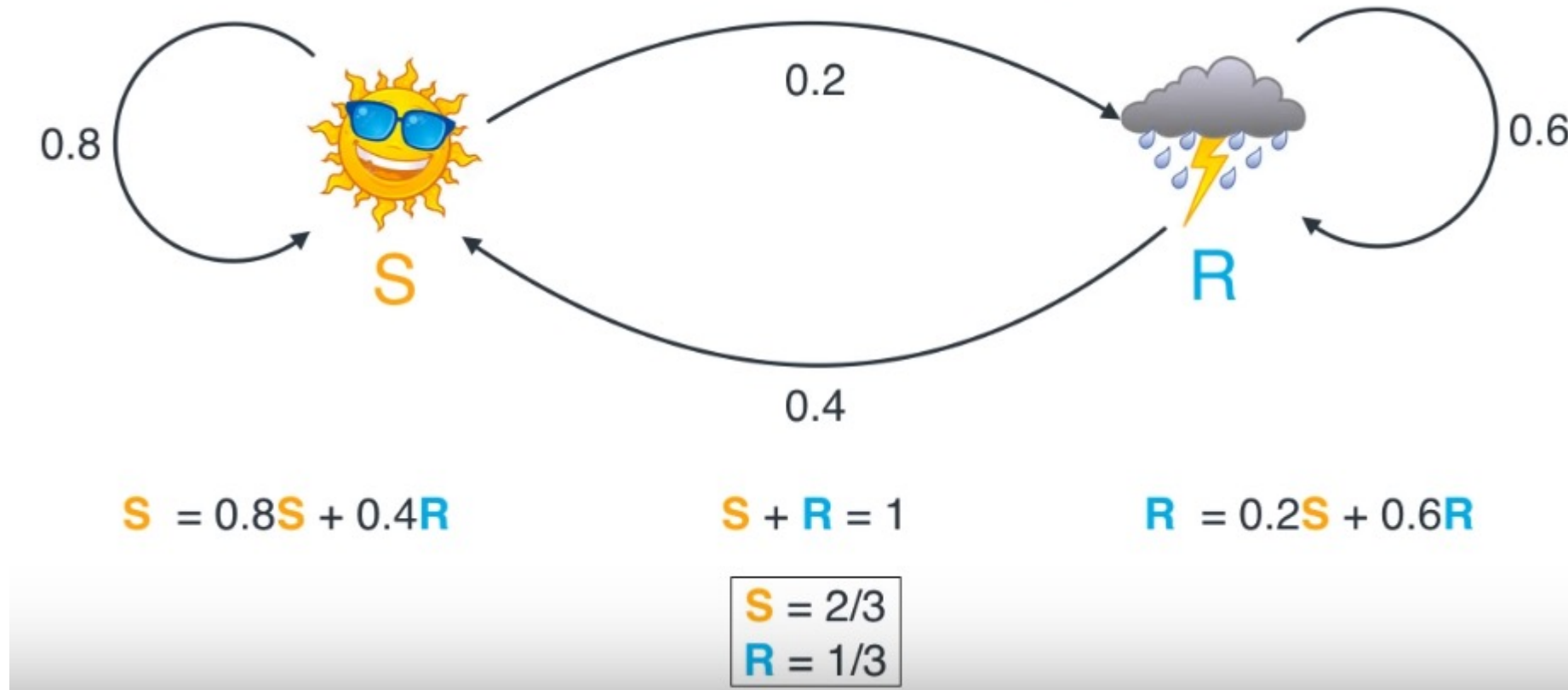
$$S = 0.8S + 0.4R$$

$$R = 0.2S + 0.6R$$

The probability of S tomorrow depends on the weather today

We can calculate the probability of sunshine/rain directly from the Markov chain.  
Here we have two equations, one for each state.

# Probability of sunshine



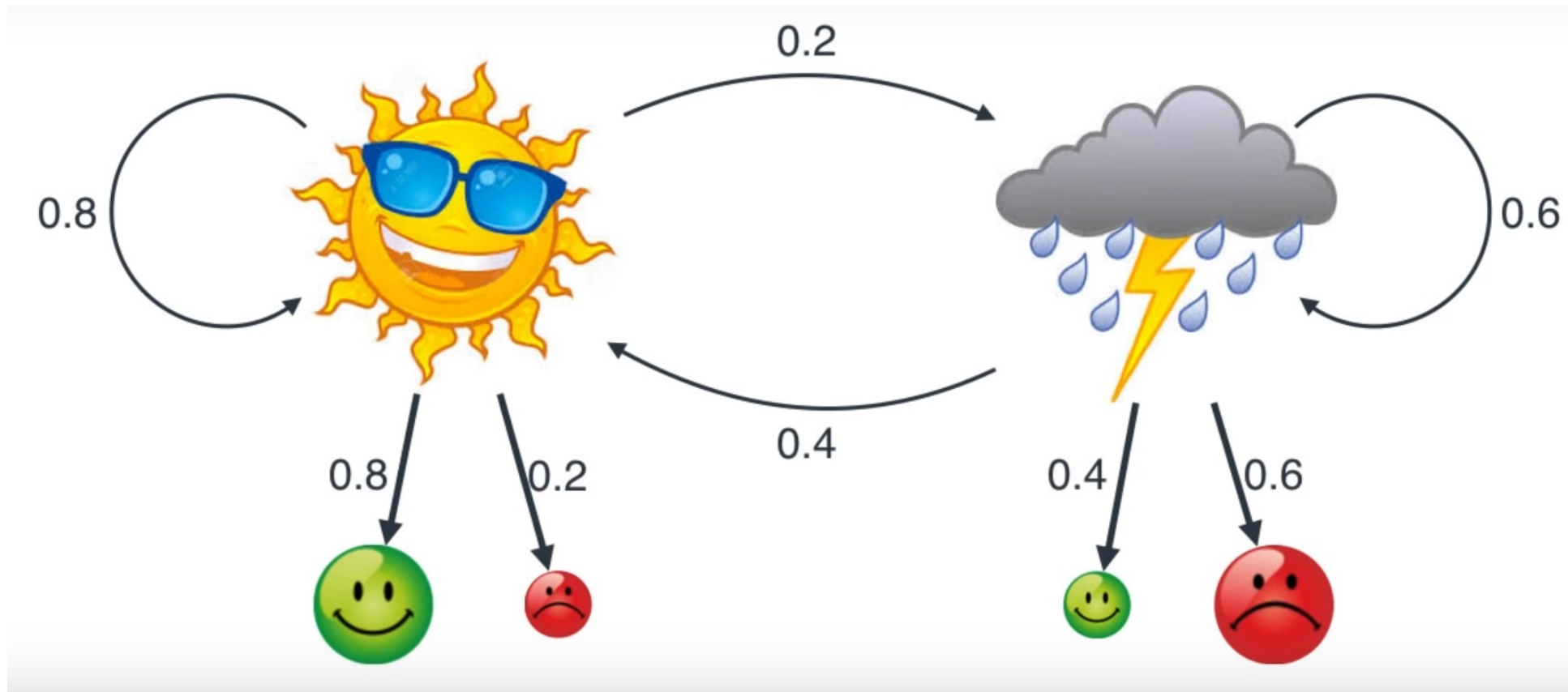
We have 2 equations with 2 unknowns. Unfortunately it is the same equation in disguise ( $S=2R$ ).

However, we also know that  $S+R=1$ , so we can use that and solve the equation system!

# Hidden Markov models

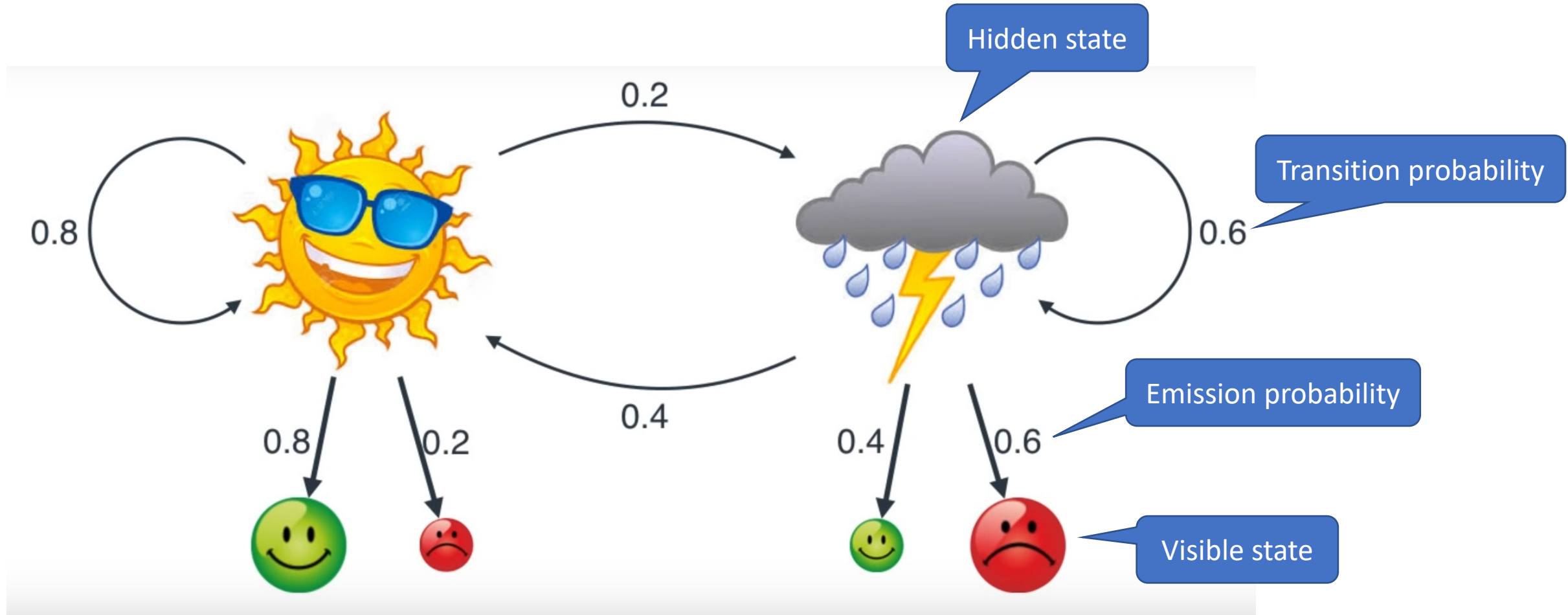
[Luis Cerrano](#)

# Example of a Hidden Markov model



This is a HMM. It represents the weather and how Bob's mood depends on it.

# Terminology



# More data

Now suppose our data includes Bob's moods





# Finding emission probabilities from data



On how many sunny days is Bob happy?



8

0.8



2

0.2



2

0.4

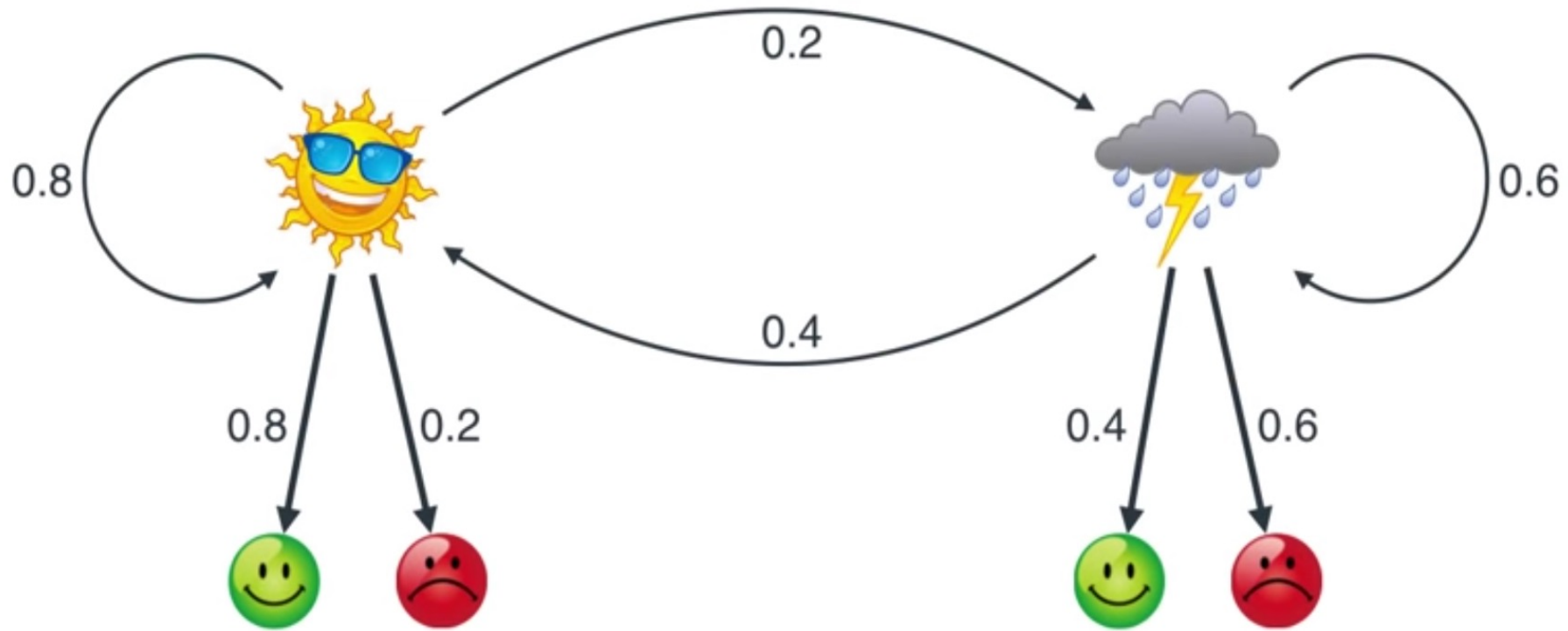


3

0.6

Before we saw how to find the transition probabilities from data. Now we know how to generate the entire HMM from data!

# Finding the HMM from data

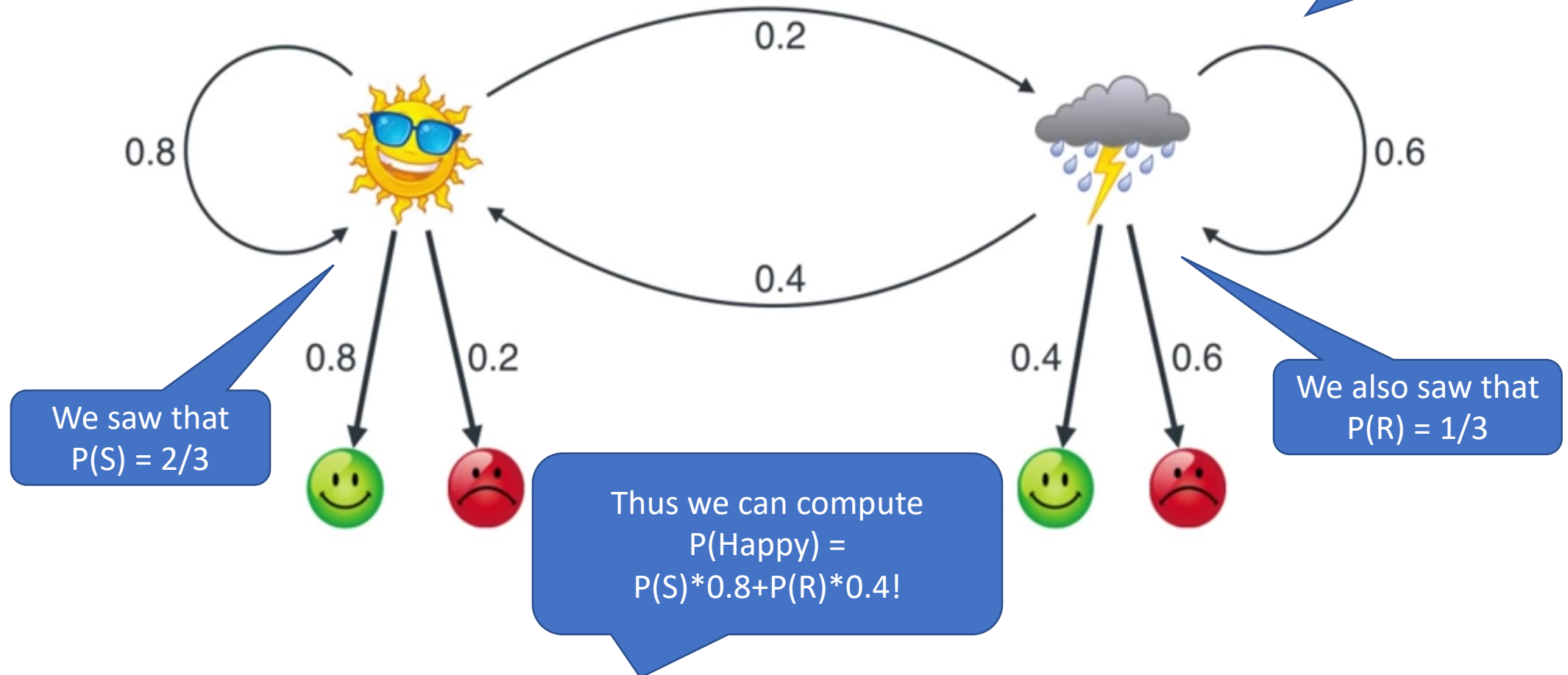


This HMM was generated from the data.

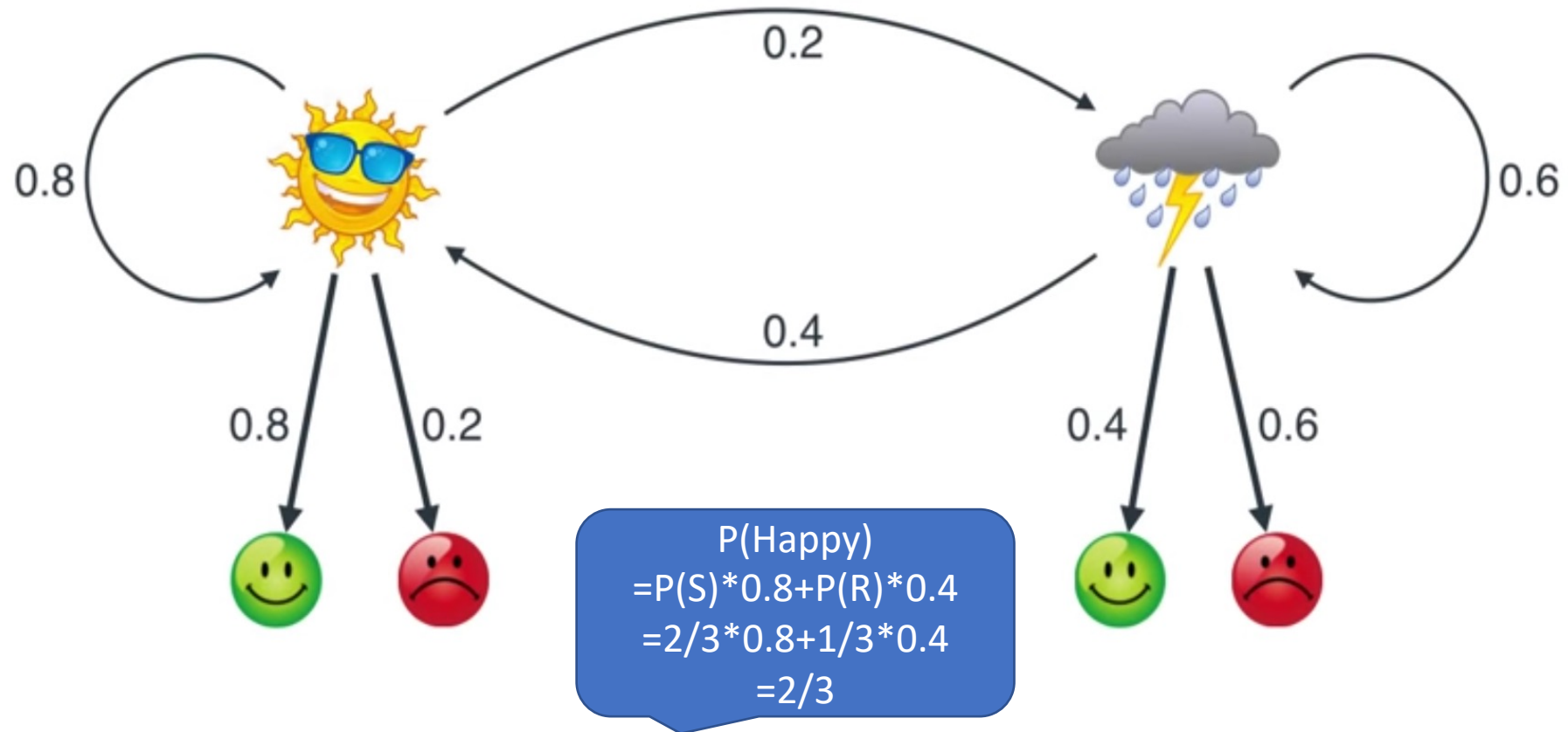
# Probability of Bob being happy

Now we can answer several questions by analyzing the HMM.

What is the probability that Bob is happy?

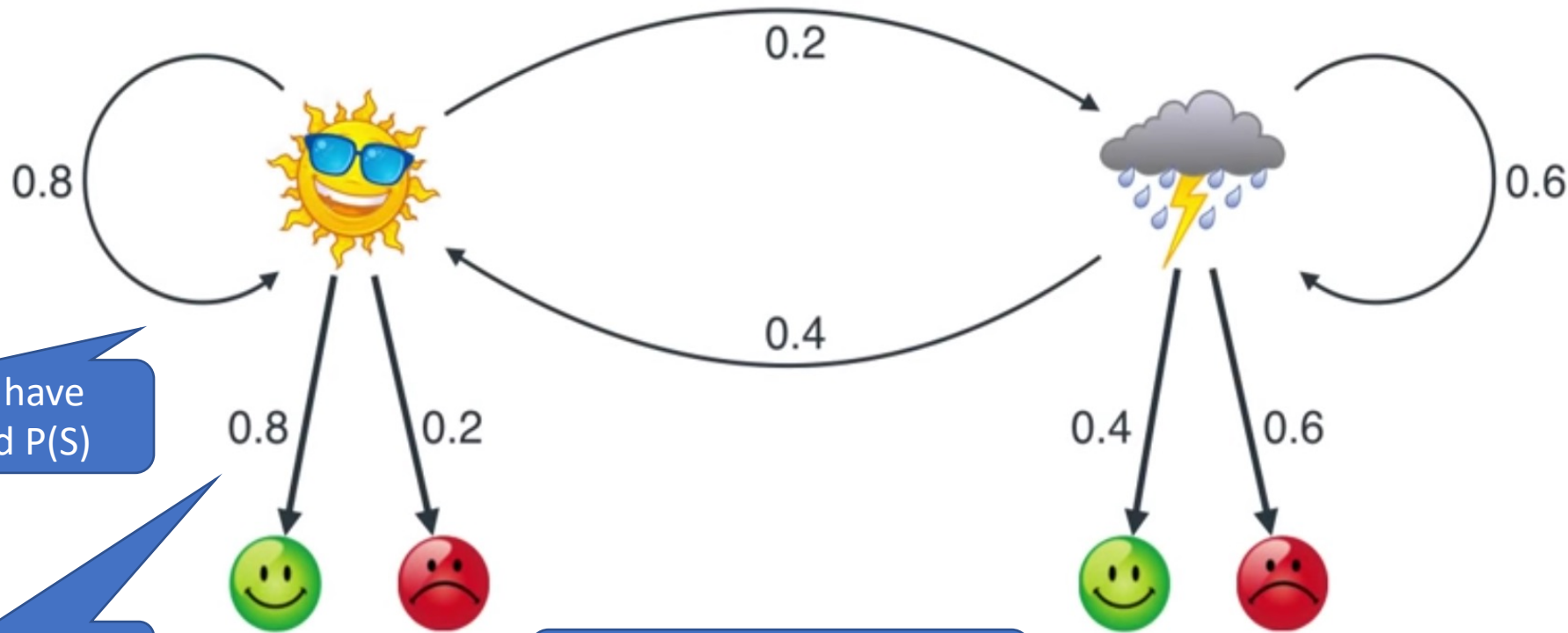


# Probability of Bob being happy



# Probability of sunshine given happy

Suppose we know that Bob is happy. Then what is the probability of sunshine,  $P(S|H)$ ?



Well, we have computed  $P(S)$

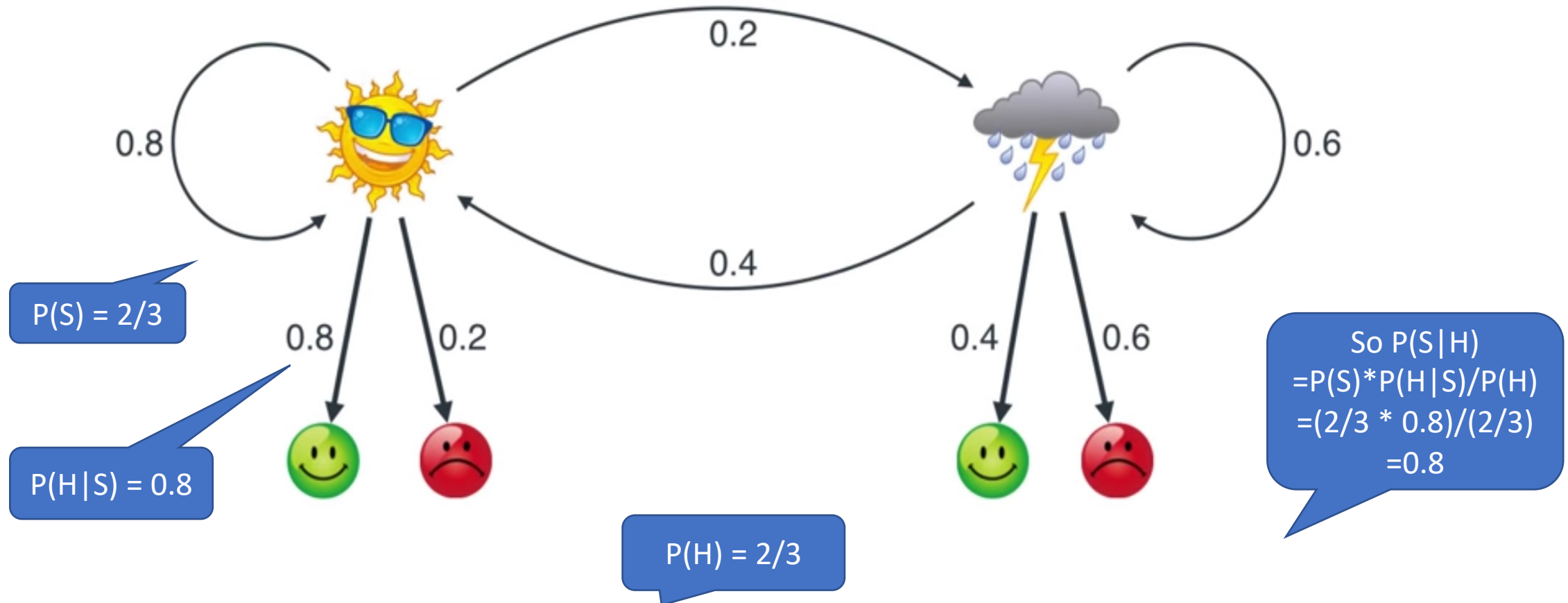
And here is  $P(H|S)$

And we just computed  $P(H)$

So we can use Bayes rule to compute  $P(S|H)$ !

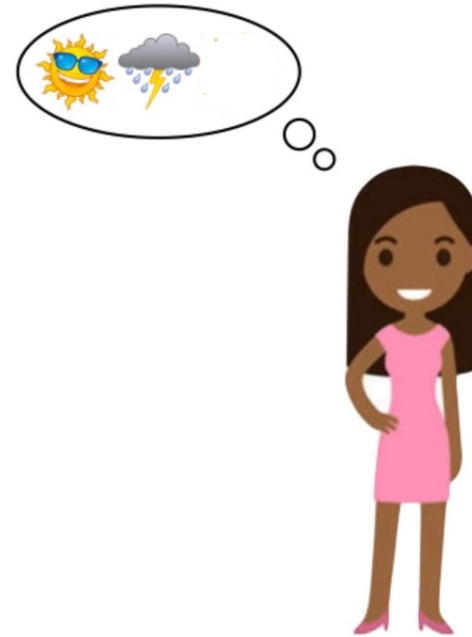
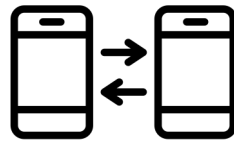
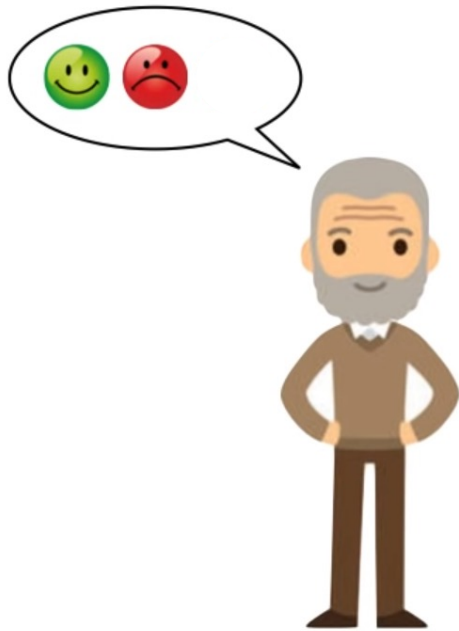
# Probability of sunshine given happy

Suppose we know that Bob is happy. Then what is the probability of sunshine,  $P(S|H)$ ?

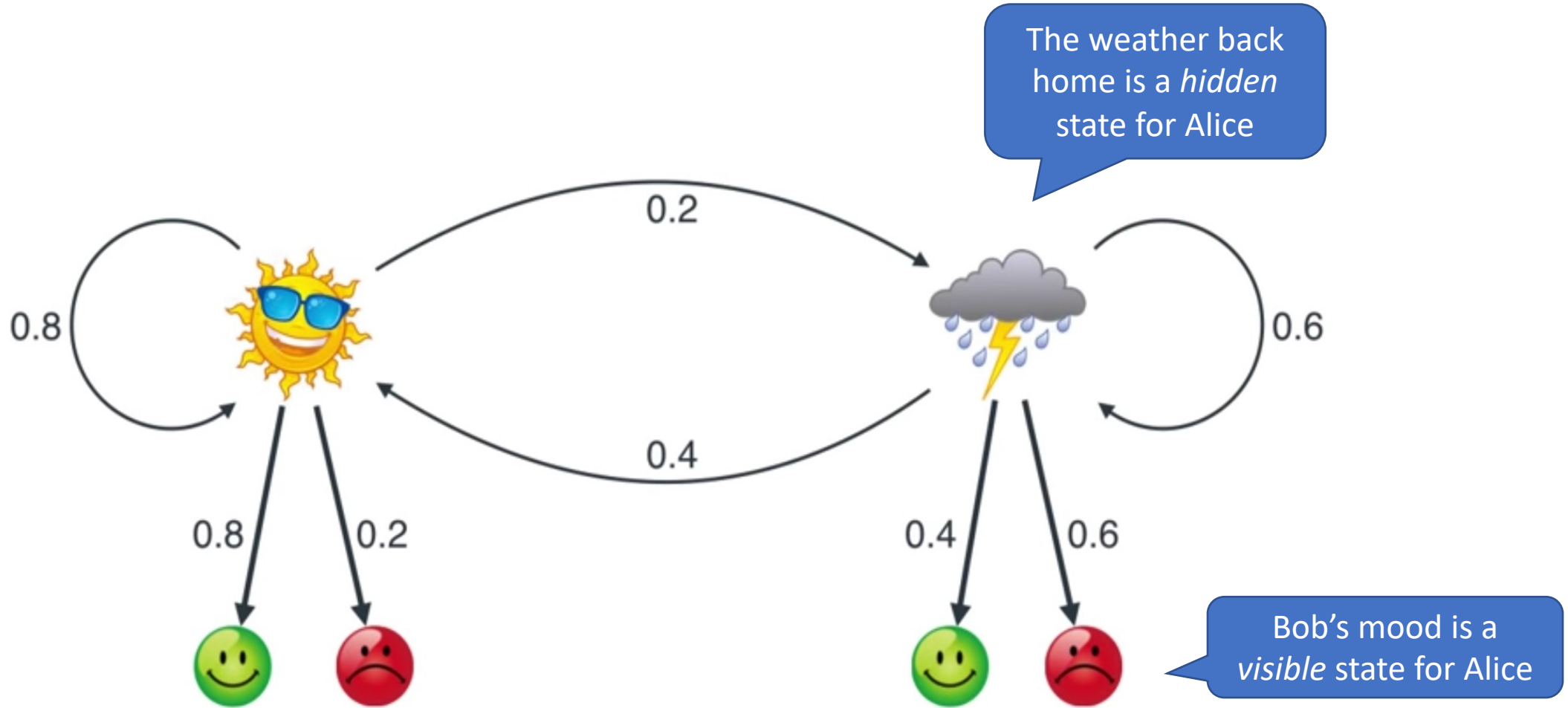


# Introducing Alice

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.



# Hidden and visible states



For Alice, Bob's mood is a visible state (observation), but the weather at his end is a hidden state.



# Alice computes the most likely weather sequence

Wednesday

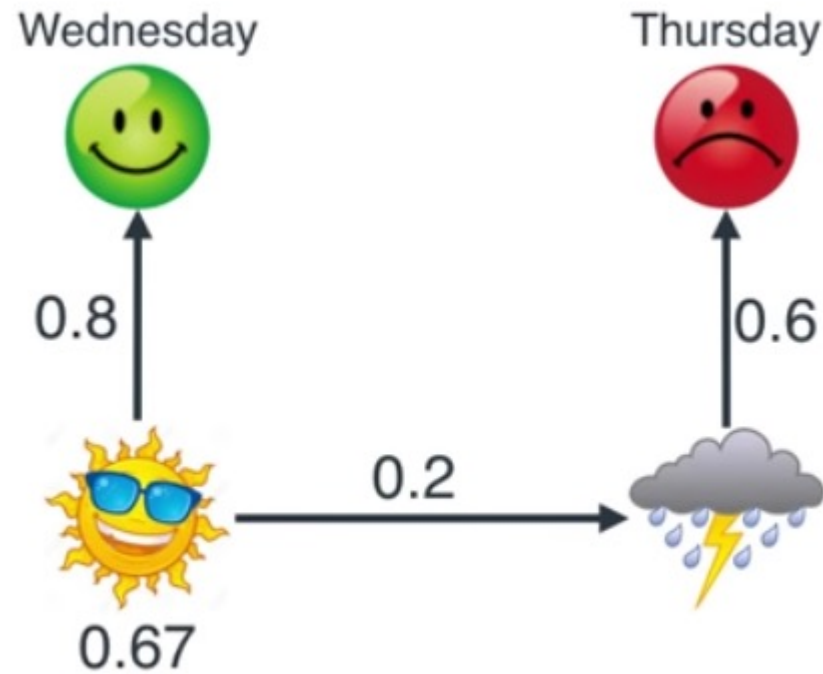


Thursday



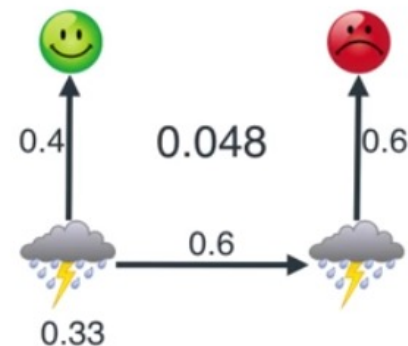
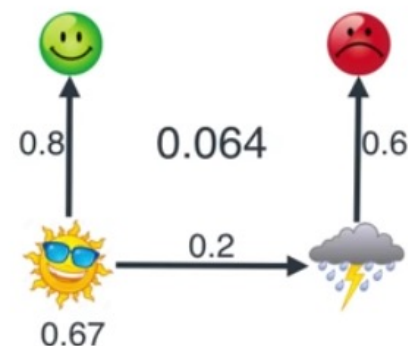
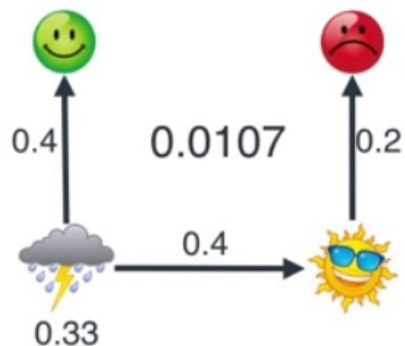
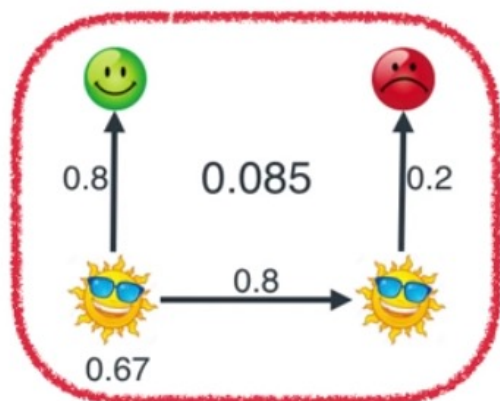
What is the most likely weather sequence if Bob is happy followed by sad? Could it be sunny followed by rainy? Or sunny followed by sunny?

# Likelihood of a sequence



What is the probability that it was sunny followed by rainy? Multiply the numbers together for the probability of everything shown happening:  $0.67 \times 0.8 \times 0.2 \times 0.6 = 0.064$

# Maximum likelihood



Do the same with all 4 weather sequences of two days. Pick the one that is most likely to happen.  
The winner is sunny followed by sunny!

# Maximum likelihood

Monday



Tuesday



Wednesday



Thursday



Friday

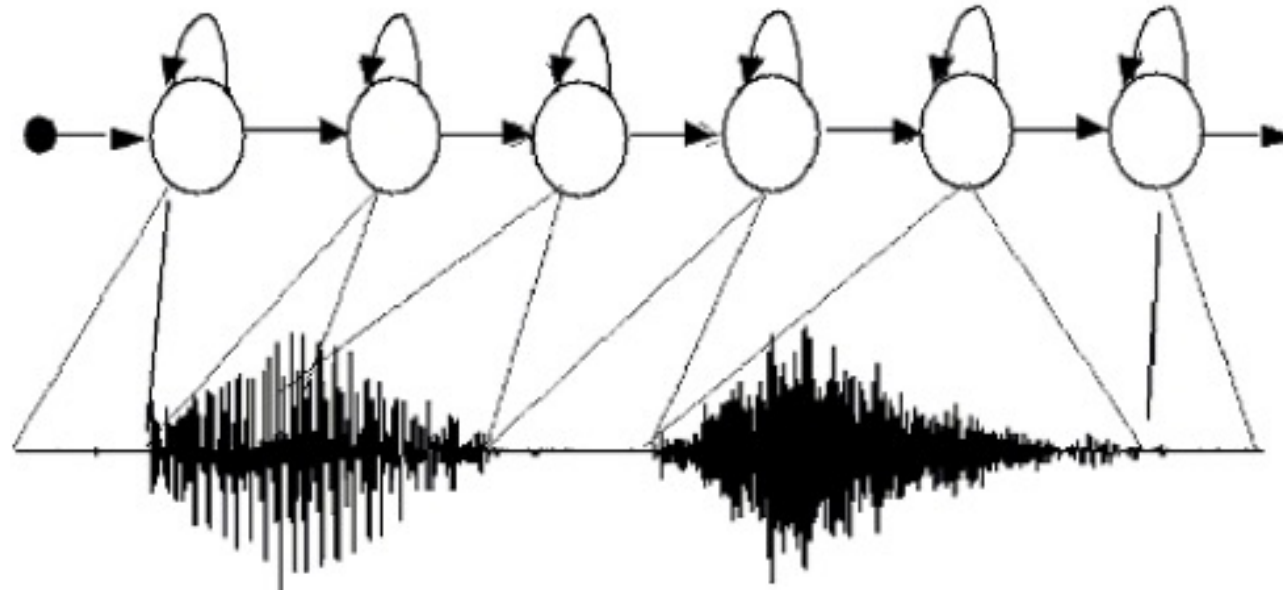


Saturday



General problem: Given a sequence of observations, find the most likely sequence of states.

# Speech recognition



1. Soundwaves  
when someone  
says a word

2. Look for the most likely  
sequence of phonemes = word

Phonemes are the  
hidden states

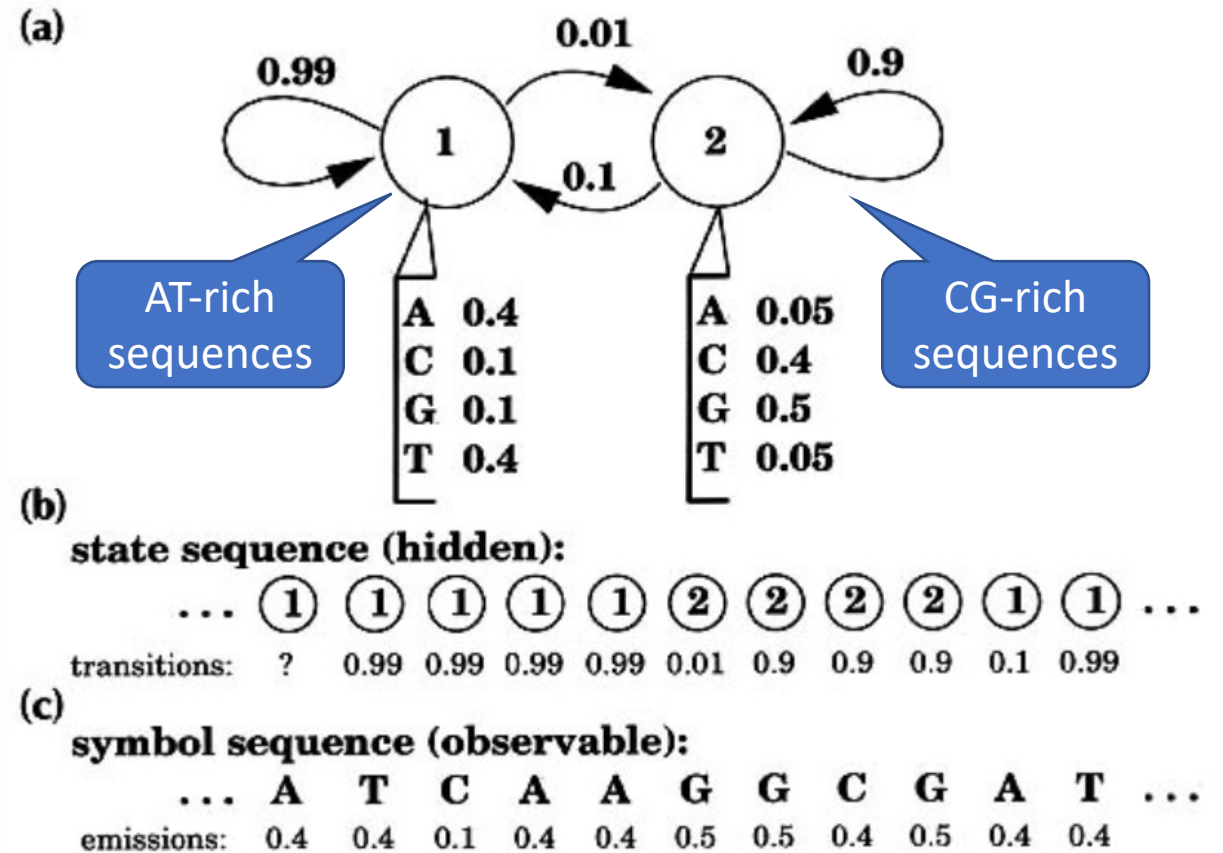
Properties of sound  
waves are the  
observed states

[Source](#)

# DNA analysis



[Source](#)



[Source](#)

# Robot localization



Hidden states: locations  
Observations: distances from objects

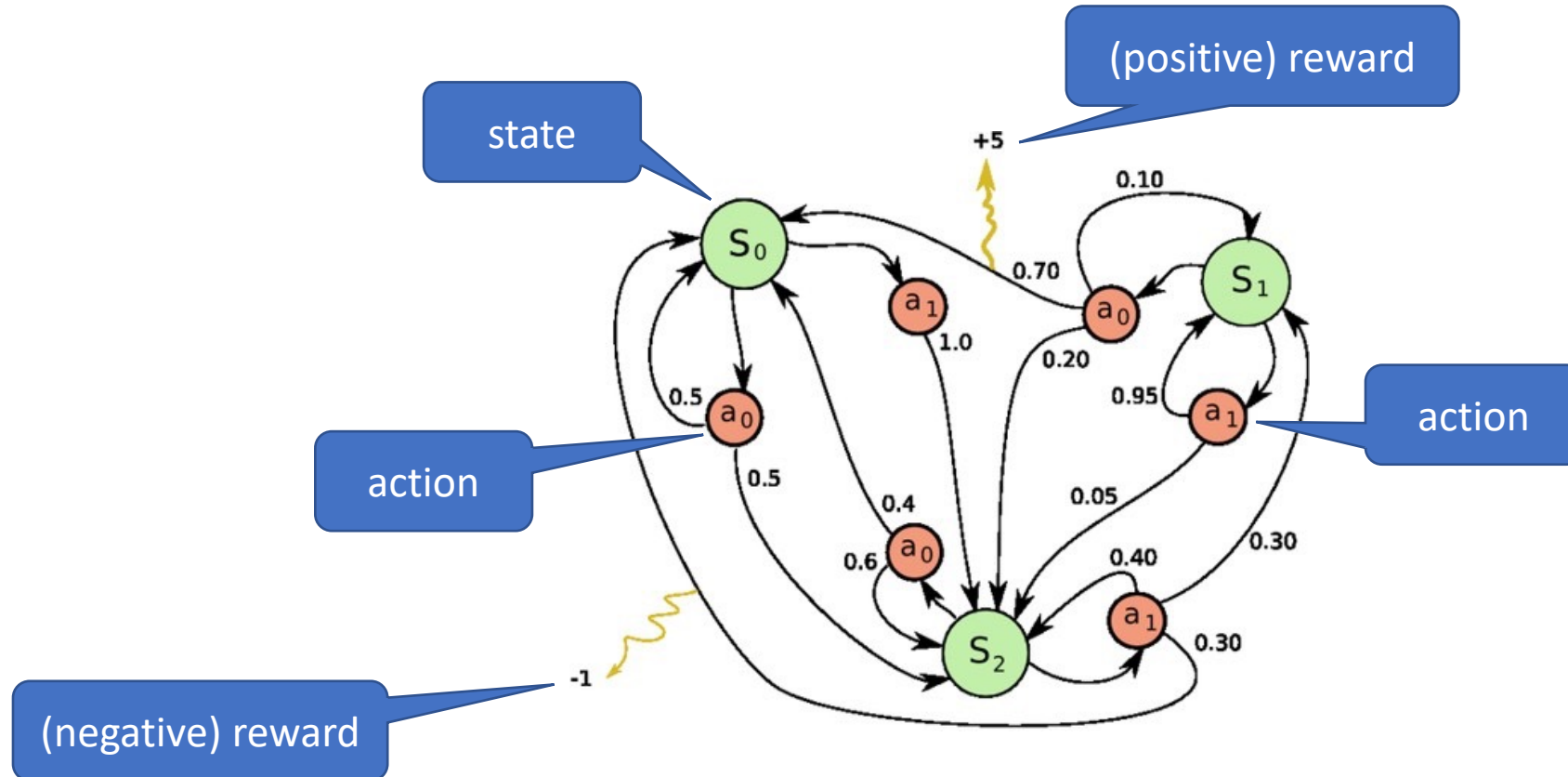
Given a robot percept, find the most likely location.

# Markov Decision Processes



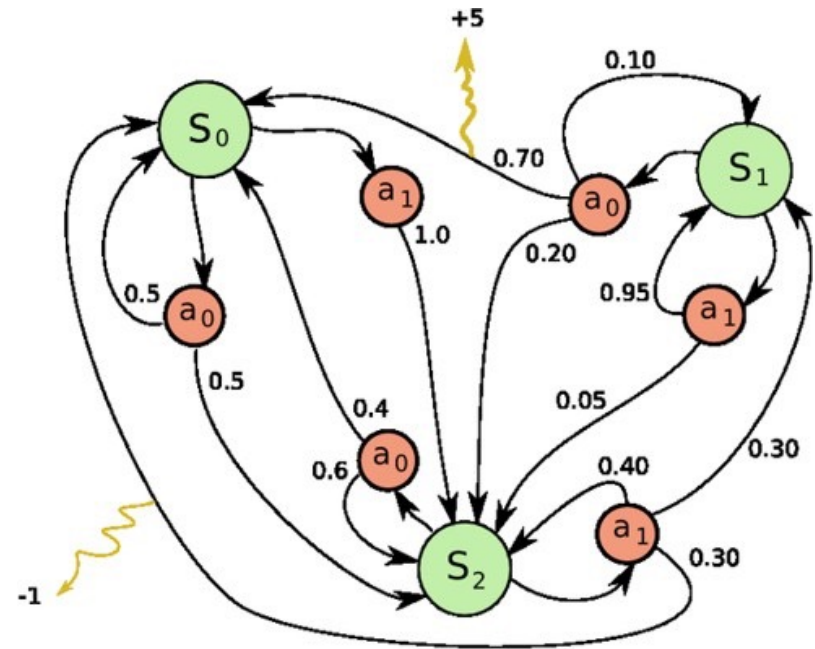
# Markov Decision Processes

- A *Markov Decision Process (MDP)* is Markov chain extended with actions and rewards.



# Challenge

- Find a policy that gives maximum expected reward in the long run
- A *policy* is a mapping from the set of states to the set of actions
- Example of a policy:
  - In state  $S_0$  do action  $a_1$
  - In state  $S_1$  do action  $a_0$
  - In state  $S_2$  do action  $a_1$



# Interaction

- An MDP can be viewed as a problem. A solution to an MDP is a policy that maximizes reward over time (in some precisely defined way).
- An MDP describes the interaction between an agent and an environment from a bird's eye's perspective.
- In general, the agent does not have the full picture of its environment (MDP). The more it interacts with it, the more it can learn about it and the more reward it can collect.

# Psychological test

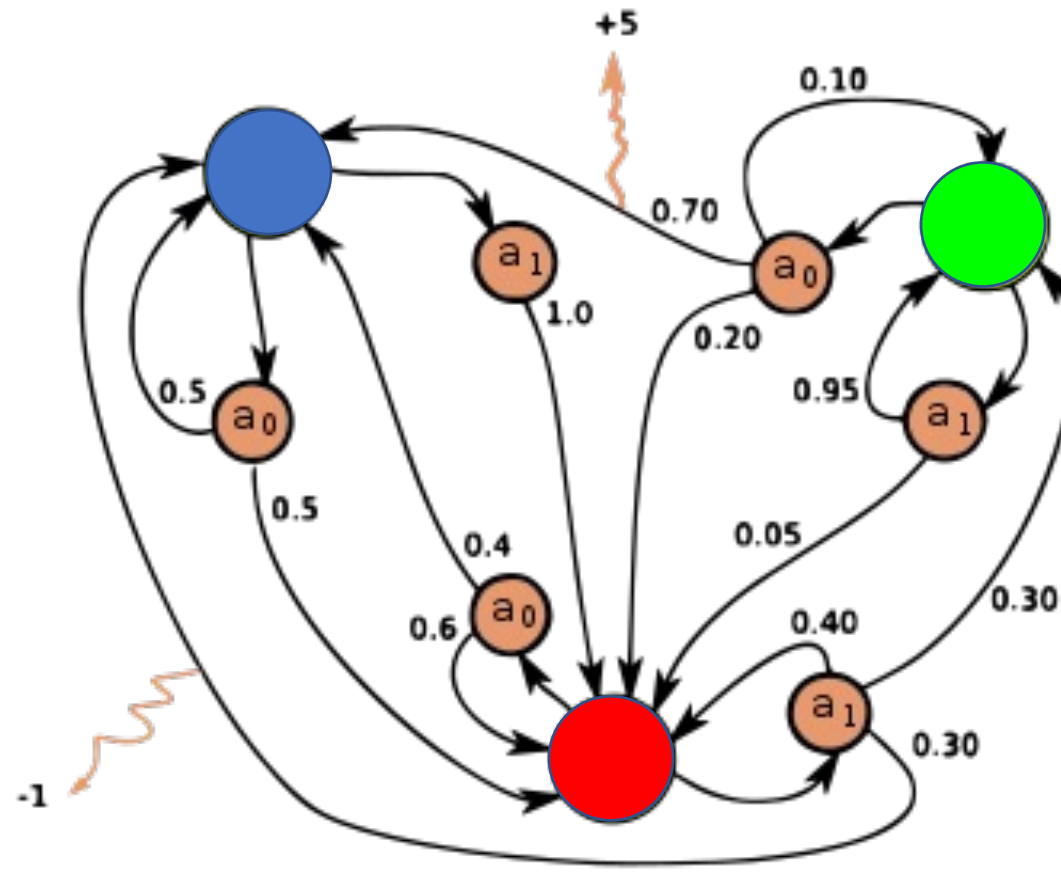
- A color is shown on the screen (the present state)
- Select between two or more actions (buttons)
- Candies (rewards) are given through a dispenser
- How much candy can you collect in 5 minutes?



These tests can be adapted to certain non-human animals!

# Psychological test

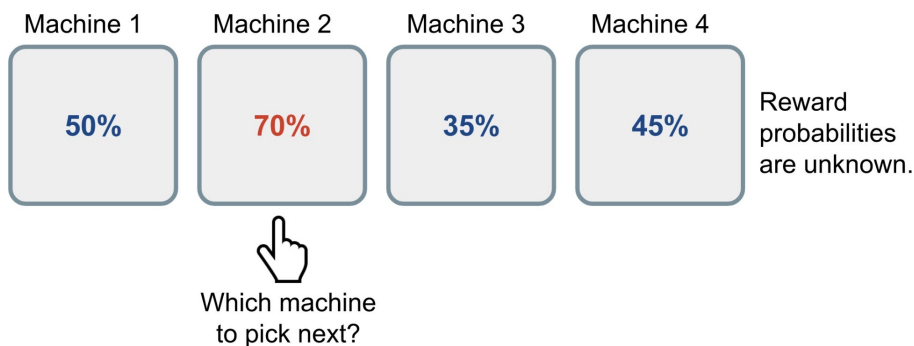
Each MDP defines a psychological test!



Use one color per state (in this case three colors)

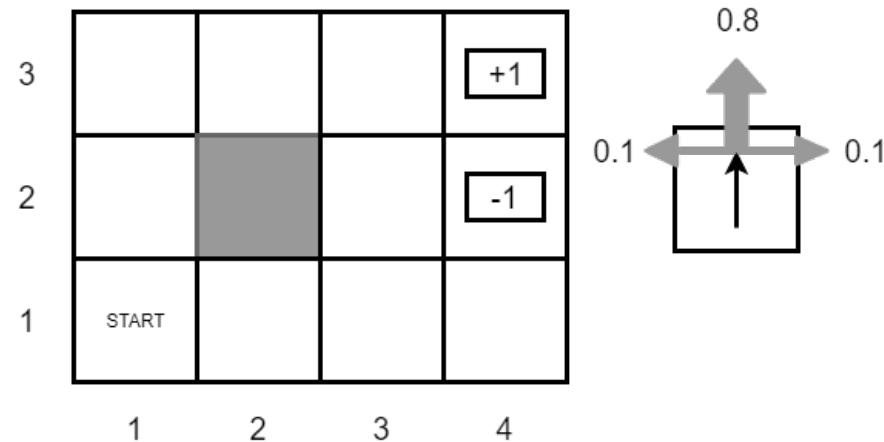
# Bandit problems

A *multi-armed bandit problem* is an MDP with just one state. Intuitively it has one action per slot machine. For each action/slot machine, there are rewards with different probabilities.



# Robot world

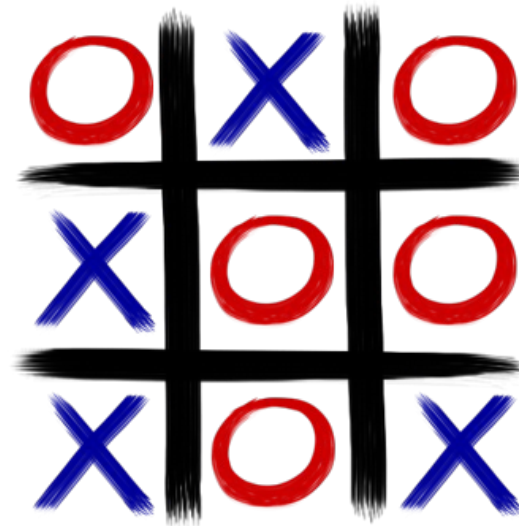
Example of an MDP



This robot world can be viewed as an MDP. The robot can be in one of 11 squares (11 states) and move in 4 directions (4 actions). If it moves up, it ends up in the square above (or bounces back) with probability 0.8. It may also slip and end up in one of the side squares (or bounce back) with probability 0.2. The rewards are 0, -1, or +1, as shown in the figure.

# Tic-tac-toe

Example of an MDP



Tic-tac-toe can be viewed as an MDP with states as nodes and moves as actions. You play X, get a position and have to make a decision (move). When you move into a final position you get the reward 0 or 1. All transition probabilities are 1.



# Breakout

Example of an MDP



Breakout can be viewed as an MDP with states = images and moves (Left, Right) as actions. Rewards are additions to the score.

# Reinforcement learning in biology

# Reinforcement learning in biology



Reinforcement learning is learning from trial and error via reward and punishment.  
Almost all animals use it, including [humans](#).

# Reinforcement learning in biology



It is known that dogs learn more efficiently from reward alone than from punishment alone or from reward and punishment in combination

Dogs can learn advanced tricks from treats

# Reinforcement learning in biology



Animals learn from the past and apply that experience to decision-making in the present. This is only helpful to the extent that the present resembles the past. Radical behaviorism: all behavior (including thoughts and emotions) is determined by the history of rewards and punishments.

# Reinforcement learning in biology

- Reward and punishment can be defined in terms of homeostatic variables, e.g. oxygen, energy, water, temperature, pleasure, and pain. Each variable has a desired value (sweet spot). Moving closer to the sweet spot means reward. Moving away from the sweet spot means punishment.
- Animals develop policies (=mappings from states to actions) that enable them to find resources, e.g. food, and avoid dangers, e.g. cold.
- Animals with sufficiently good (adaptive) policies + luck will be able to survive and reproduce. The others will not.

# Reinforcement learning in computer science

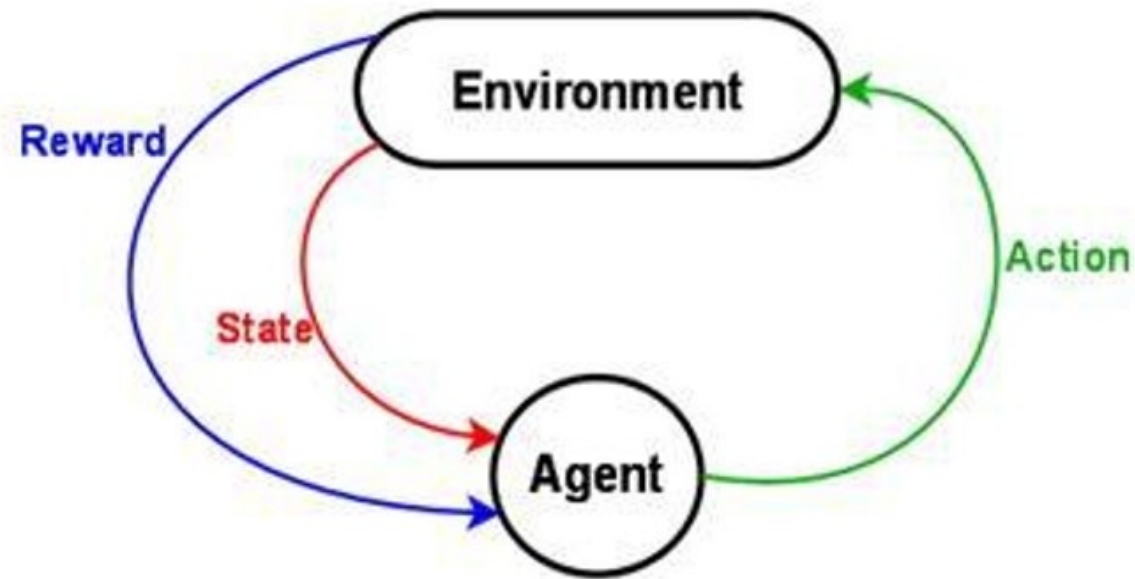
# Reinforcement learning in computer science

Computer science has borrowed the term reinforcement learning from biology/neuroscience/psychology:

*The class of machine learning algorithms concerned with maximizing some form of reward for an agent in an environment by learning how to best select among a set of actions is called reinforcement learning. (Wikipedia)*

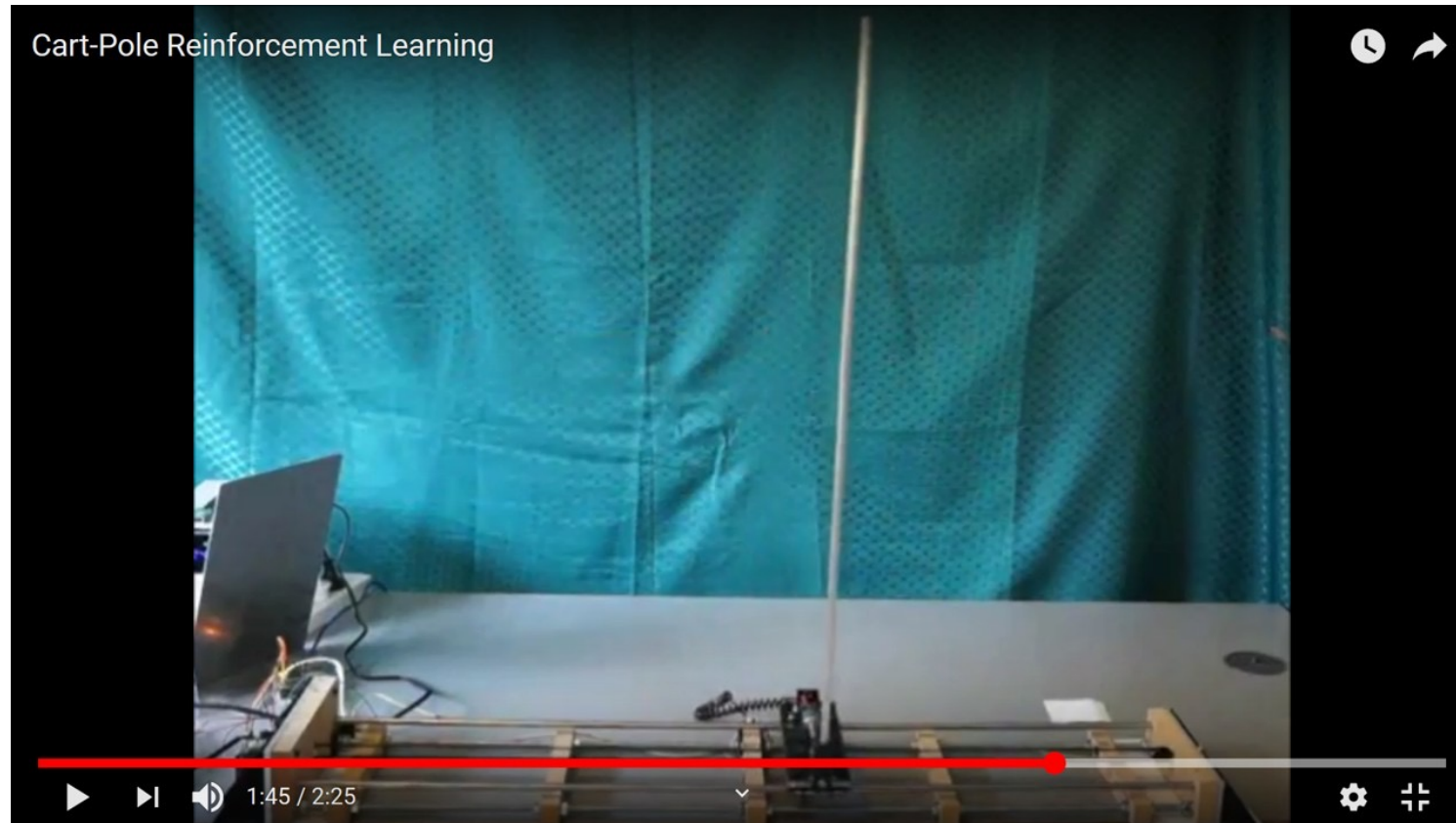


# Reinforcement learning



Agents try to act so as to maximize reward in the short or long run.

# Pole balancing



Pole balancing. States: 32 states of the form (angle, angle speed).  
Actions: move cart left, move cart right. Reward:  $-\text{abs}(\text{angle})$ .

# Great generality

What is this amazingly general mechanism that can both learn to move like a snake, control a power plant, and play board games?  
Let's take a closer look!

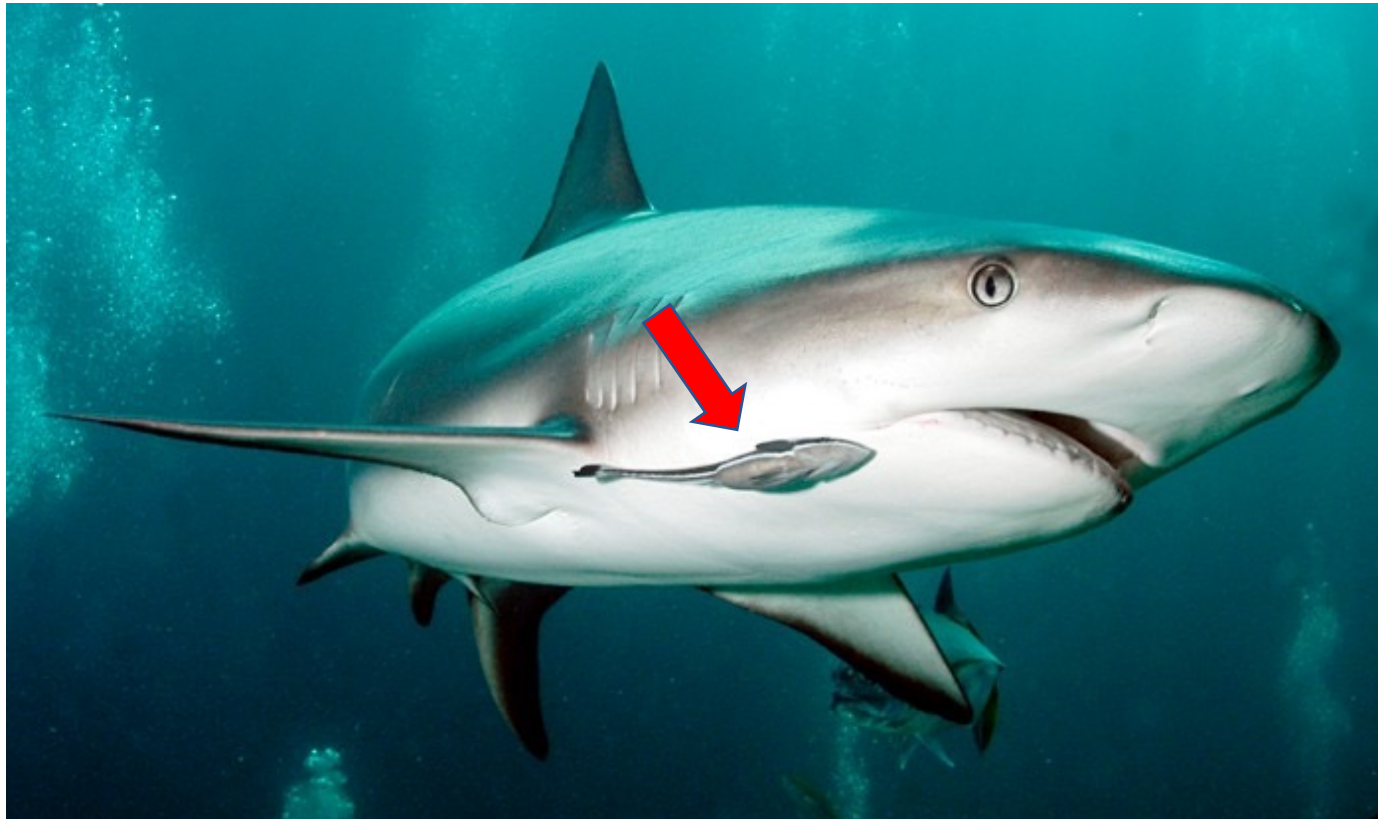
# Epsilon greedy algorithms

- A common class of reinforcement learning algorithms are called  $\epsilon$ -greedy. Here  $\epsilon$  is a real number, e.g.  $\epsilon = 0.1$ .
- At each iteration these algorithms explore with probability  $\epsilon$  and exploit with probability  $1-\epsilon$ .
- To *explore* means taking a random action and to *exploit* means taking whatever action seems most rewarding at the present moment while forgetting about future consequences (the greedy action).

# What's the point of exploring?

- If we exploit all the time (and never make untried moves), many actions in many states will remain untried. Thus we might miss actions that could be much better than anything we have tried before. Hence exploration is critical to finding an optimal solution.
- On the other hand, if we explore all the time, then we do not use the knowledge that we have accumulated in order to increase our performance.
- Hence a mix of exploitation and exploration is needed for optimal performance in the long run!

# Exploit or explore?



The remora fish eats scraps of prey dropped by the shark and parasites on the shark's skin. Should it stay or swim away looking for a better shark?

# Exploit or explore?

- Examples of the exploit or explore dilemma in everyday life:
  - 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?

Q-learning



# Accumulated reward

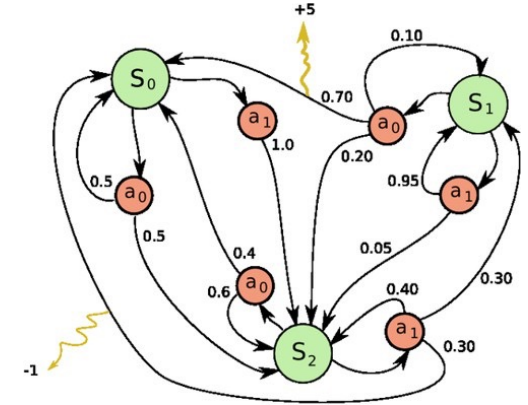
Any policy defines an interaction sequence:

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

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

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

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



# Accumulated discounted reward

Reward now is better than reward later in many cases...

We can include that aspect into our model by using a discount factor  $\gamma \in [0, 1]$  and define the *accumulated discounted reward* of the above sequence as

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

If  $\gamma < 1$  and the MDP is finite (so that all rewards are bounded), then this sum converges even if the sequence is infinite.

# Accumulated discounted reward

Important observation:

$$\begin{aligned} R_t &= r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \dots \\ &= r_t + \gamma \cdot [r_{t+1} + \gamma \cdot r_{t+2} + \dots] \\ &= r_t + \gamma \cdot R_{t+1} \end{aligned}$$

Here  $R_t$  is the accumulated discounted reward from  $t$  on (given a certain policy).

In words: the accumulated discounted reward from  $t$  on is the present reward plus the discounted accumulated reward from  $t + 1$  on

# Q-learning

We will use  $Q$ -values  $Q(s, a)$ , for each action  $a$  and state  $s$ .  $Q$  stands for “quality”.  $Q(s, a)$  is an estimate of the maximum accumulated discounted reward when we perform the action  $a$  in state  $s$ , and continue optimally from there.

Thus it makes sense to define the policy as:

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

In words: in  $s$  we select an action  $a$  such that  $Q(s, a)$  is maximal.

# Q-learning

- Q-learning is an algorithm capable of learning the optimal policy for any finite MDP!
- It was introduced by Watkins in 1989.
- Its exploit part is as follows:

```
initialize  $Q[num\_states, num\_actions]$  arbitrarily  
observe initial state  $s$   
repeat  
    select and carry out an action  $a$   
    observe reward  $r$  and new state  $s'$   
     $Q[s, a] = Q[s, a] + \alpha(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$   
     $s = s'$   
until terminated
```

Update formula for the Q-values.  
See next slide!

## Update formula for the Q-values

The learning rate, i.e. that extent to which new information overrides old information. This is a number between 0 and 1.

The Q function we are updating, based on state  $s$  and action  $a$  at time  $t$

The reward earned when transitioning from time  $t$  to the next next turn, time  $t+1$ .

The value of the action that is estimated to return the largest (i.e. maximum) total future reward, based on the all possible actions that can be made in the next state.

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

The arrow operator means update the Q function to the left. This is saying, add the stuff to the right (i.e. the difference between the old and the new estimated future reward) to the existing Q value. This is equivalent in programming to  $A = A+B$ .

The discount rate. Determines how much future rewards are worth, compared to the value of immediate rewards. This is a number between 0 and 1

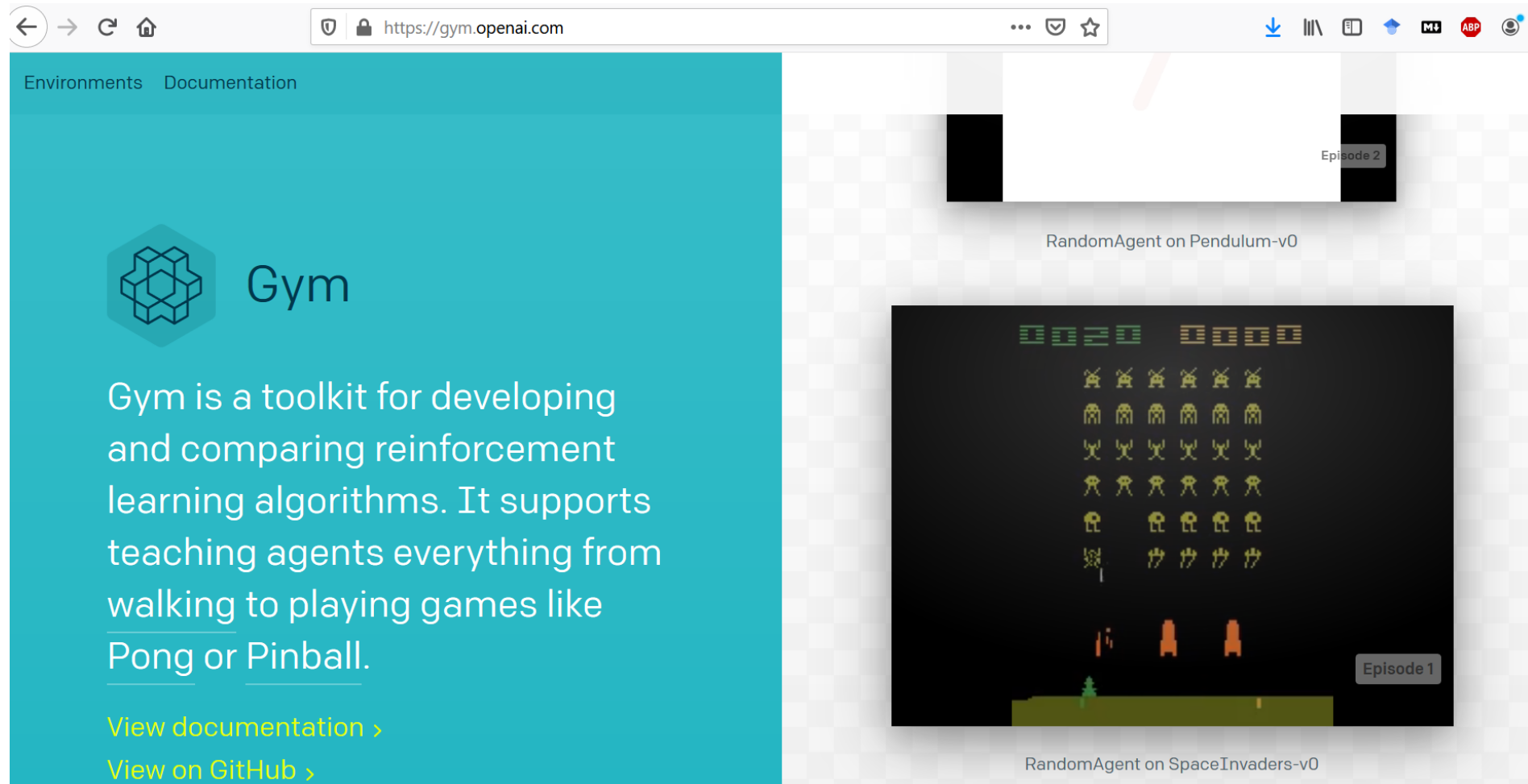
The existing estimate of the Q function, (a.k.a. current the action-value).

Greek letters are hyperparameters (that take values in  $[0, 1]$ ).

# Q-learning


- Q-learning eventually finds a policy that maximizes the accumulated discounted reward for any finite MDP. Very general!
- It may take extremely long though.
- Moreover, Q-learning based on tables has limited scalability, since one variable is needed for each (state,action)-pair.

# OpenAI gym



The image shows a browser window at <https://gym.openai.com>. The left sidebar is teal and contains the Gym logo, a description of the toolkit, and links to documentation and GitHub. The main content area displays two game environments: Pendulum-v0 and SpaceInvaders-v0, both running with a RandomAgent.

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