SAPIENZA
UNIVERSITÀ DI ROMA

Autonomous Networking

Gaia Maselli
Dept. of Computer Science

SAPIENZA
UNIVERSITÀ DI ROMA

Where we are

What is an autonomous network?

SAPIENZA UNIVERSITÀ DI ROMA

No human intervention

Existing and functioning as an independent organism

Autonomy involves intelligence

Able to learn from the environment and react accordingly

Autonomous **networking**

We address autonomy at the networking level

(communication and routing)

How do we make a squad of drones or an IoT network autonomous?

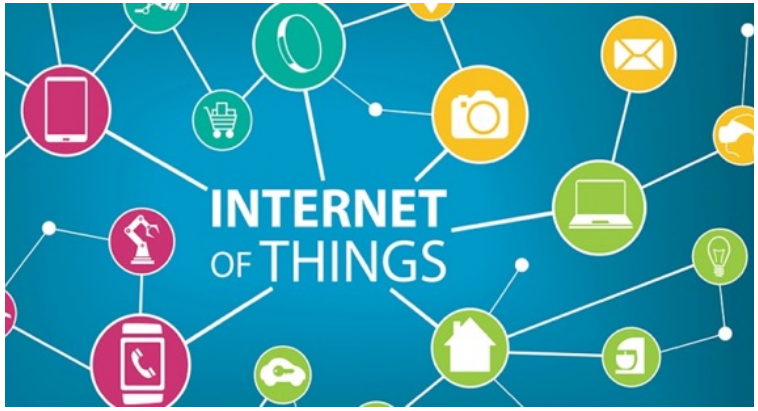
Autonomous networking

Which tecnologies?

Mainly wireless

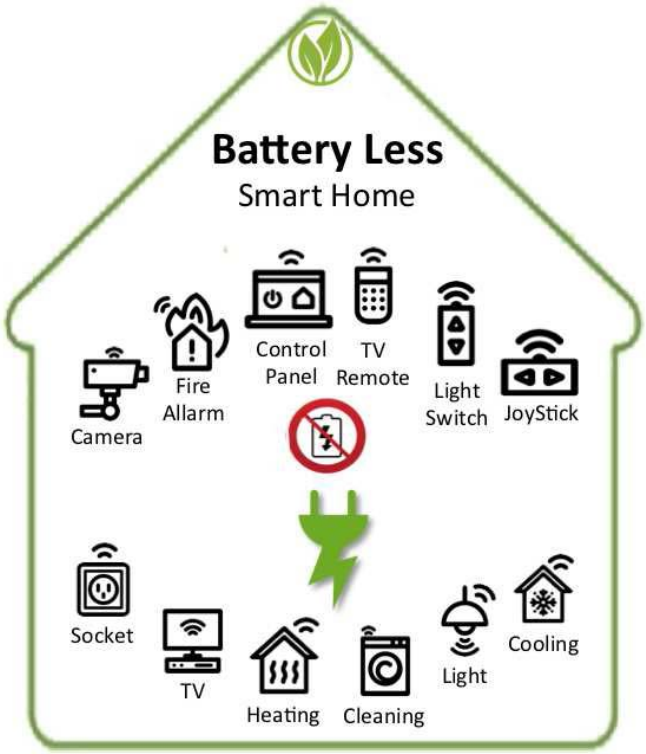
- Sensor & IoT
- Backscattering (RFID)
- Drone networks (dronet)

Sensor networks, RFID, IoT, Dronets



First case study: battery-free smart home

- Battery-free smart home
  - Cameras
  - presence sensors
  - smoke sensors
  - light sensors
  - Thermostats
  - smart meters
  - TV remotes
  - light switches
  - video game controllers



First case study: battery-free smart home

- Each device produces a new data sample with *a rate* that depends on the environment and the user
  - A joystick produces new (burst) data only when is used
  - Temperature depends on current state (normal condition, fire, etc.)
  - Presence sensor produces new data only when a person walks in front of it
  - Videocamera continuously produces new data
- The nature of the sensor data generation is spontaneous; it is highly depending on the events in the environment
- *A device should transmit only when it has new/changed data to send*
- In backscattering-based networks devices need to be queried by the reader
- Reader can query devices sequentially (one at each time step)

First case study: battery-free smart home

- Problem: efficient communication in a battery-free smart home
- In which order should the reader query tags?
- Assume prefixed timeslots
- TDMA with random access performs poorly (collisions)
- TDMA with fixed assignment also performs poorly (wasted queries, e.g., query a tag that does not have new data since the last query)
- We want to query devices that have new data samples and avoid
  - Loss of data (the device produces a new data sample before the previous has been sent to the reader)
  - Redundant queries (a device is queried multiple times for the same data sample, i.e., before it has produced a new data sample)

First case study: battery-free smart home

- Goal: **we want to design a MAC protocol that autonomously adapts to the environment**
  - Queries devices only when they have new data
  - Does not loose any data

- How can we design such a dynamic protocol?

The need for a protocol able to learn from the environment

The answer is

Reinforcement Learning

Reinforcent learning (RL)

Reinforcement learning is concerned with the really foundational issue of

How can an intelligent agent **learn** to make a **good sequence of decisions**

Learning to make a good sequence of decisions (under uncertainty)

1. Learn to make good **sequence of decisions**

- How can an intelligent agent make not just one decision but a whole sequence of decisions

2. Learn to make **good** sequence of decisions

- Goodness: we have some utility measure over the decisions that are being made

3. Learn to make good sequence of decisions

- Learning
- The agent does not know in advance how its decisions are going to affect the world
- What decisions might be associated with good outcomes
- The agent has to acquire this information through experience

Learning by interaction with the environment

- A learning agent can figure out how the world works by simply trying things and see what happens
- That's what we think people and animals do
  - When an infant plays or waves its arms, it has no explicit teacher but it has a sensorimotor connection to its environment
  - Exercising this connection produces a wealth of information about cause and effect, about the consequences of actions, and about what to do in order to achieve goals
  - When we are learning to drive a car we are acutely aware of how our environment responds to what we do, and we seek to influence what happens through our behavior
- We explore a **computational approach to learning from interaction**
- The approach we explore is focused on **goal-directed learning from interaction**

Reinforcement Learning: characteristics

- RL is learning what to do - how to map situations to actions – so as to maximize a numerical reward signal
- RL presents **two main characterstics**:
- **Trial-and-error search**: The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them
- **Delayed reward**: In many cases actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards

Reinforcement learning: method

- Sensation, action, goal are the 3 main aspects of a reinforcement learning method
- A learning agent must be able to
  - **sense** the state of its environment to some extent
  - take **actions** that affect that state
- The agent also must have a **goal** or goals relating to the state of the environment
- Any method that is well suited to solving such problems we consider to be a reinforcement learning method

Difference from other machine learning paradigms

- There is no supervisor (no training), only a reward signal
- Feedback may be delayed (a catastrophic negative reward can arrive after some positvie rewards)
- Time really matters (sequential decision making)
- Agent's actions affect the subsequent data it receives
- A sequence of successful decisions will result in the process being reinforced, because it best solves the problem at hand
- **RL learns online** rather than just learning from data

Learning online

- In RL, we focus on the problem of **learning while interacting with an ever changing world**.
- We don't expect our agents to simply compute a good behaviour and then execute that behaviour in an open loop fashion.
- We expect our agents to get things wrong, to refine their understanding as they go.
- **The world is not a static place**. We get injured, the weather changes, and we encounter new situations and our goals change.
- An agent that immediately integrates its most recent experience should do well especially compared with ones that attempt to perfectly memorize how the world works.
- The idea of **learning online** is extremely powerful and is a defining feature of RL.

Real-world applications of RL

Self-driving cars

- Some of the autonomous driving tasks where reinforcement learning could be applied include trajectory optimization, motion planning, dynamic pathing, controller optimization, and scenario-based learning policies for highways.

Industry automation

- reinforcement learning-based **robots** are used to perform various tasks. Apart from the fact that these robots are more efficient than human beings, they can also perform tasks that would be dangerous for people.

Real-world applications of RL

Healthcare

- In healthcare, patients can receive treatment from policies learned from RL systems. RL is able to find optimal policies using previous experiences without the need for previous information on the mathematical model of biological systems. It makes this approach more applicable than other control-based systems in healthcare.
- The use of RL in healthcare also enables improvement of long-term outcomes by factoring the delayed effects of treatments.

Real-world applications of RL

Engineering

- Facebook has developed an **open-source reinforcement learning platform**—Horizon. The platform uses reinforcement learning to optimize large-scale production systems. Facebook has used Horizon internally:
  - to personalize suggestions
  - deliver more meaningful notifications to users
  - optimize video streaming quality

News recommendation

- User preferences can change frequently, therefore recommending news to users based on reviews and likes could become obsolete quickly. With reinforcement learning, the RL system can track the reader's return behavior.

Real-world applications of RL

Applications in trading and finance

Natural Language Processing

Gaming

Real-time bidding

Robotics

And …

Computer networks !

Rewards

- A reward R_t is a scalar feedback signal
- Indicates how well the agent is doing at step t
- The agent's job is to maximize cumulative reward

RL is based on the reward hypothesis:

All goals can be described by the maximisation of expected cumulative reward

Examples of rewards

 Make a humanoid robot walk	+ve reward for forward motion −ve reward for falling over
 Fly stunt manoeuvres in a helicopter	+ve reward for following desired trajectory −ve reward for crashing
 Communication in battery-free environments	+ve reward for a device with new data −ve reward for a device that has not produced new data

Sequential decision making

- Goal: *select actions to maximise total future reward*
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward

Examples:

- A financial investment (may take months to mature)
- Refuelling a helicopter (might prevent a crash in several hours)
- Blocking opponent moves (might help winning chances many moves from now)

Exploration-exploitation dilemma

- One of the **challenges** that arises in RL is the trade-off between exploration and exploitation
- To obtain a lot of reward a RL agent must prefer actions that it has tried in the past and found to be effective in producing reward (**exploit known information**)
- But to discover such actions it has to try actions that is has not selected before (**find more information about the environment**)
- The agent has to **exploit** what it has already experienced in order to obtain reward, but it has to **explore** in oder to make better action selections in the future

Exploration vs exploitation dilemma: example

- The exploration vs exploitation dilemma exists in many aspects of our life.
- A real-life example : where to eat?
- Your favorite restaurant is right around the corner. If you go there every day, you would be confident of what you will get, but miss the chances of discovering an even better option.
- If you try new places all the time, very likely you are going to eat unpleasant food from time to time.

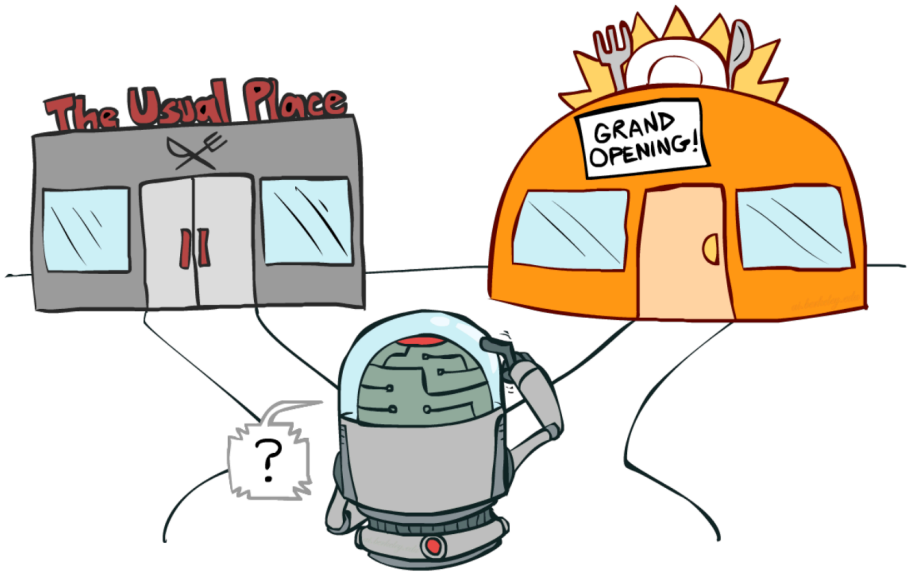


Image source: UC Berkeley AI course

Exploration vs exploitation dilemma: example

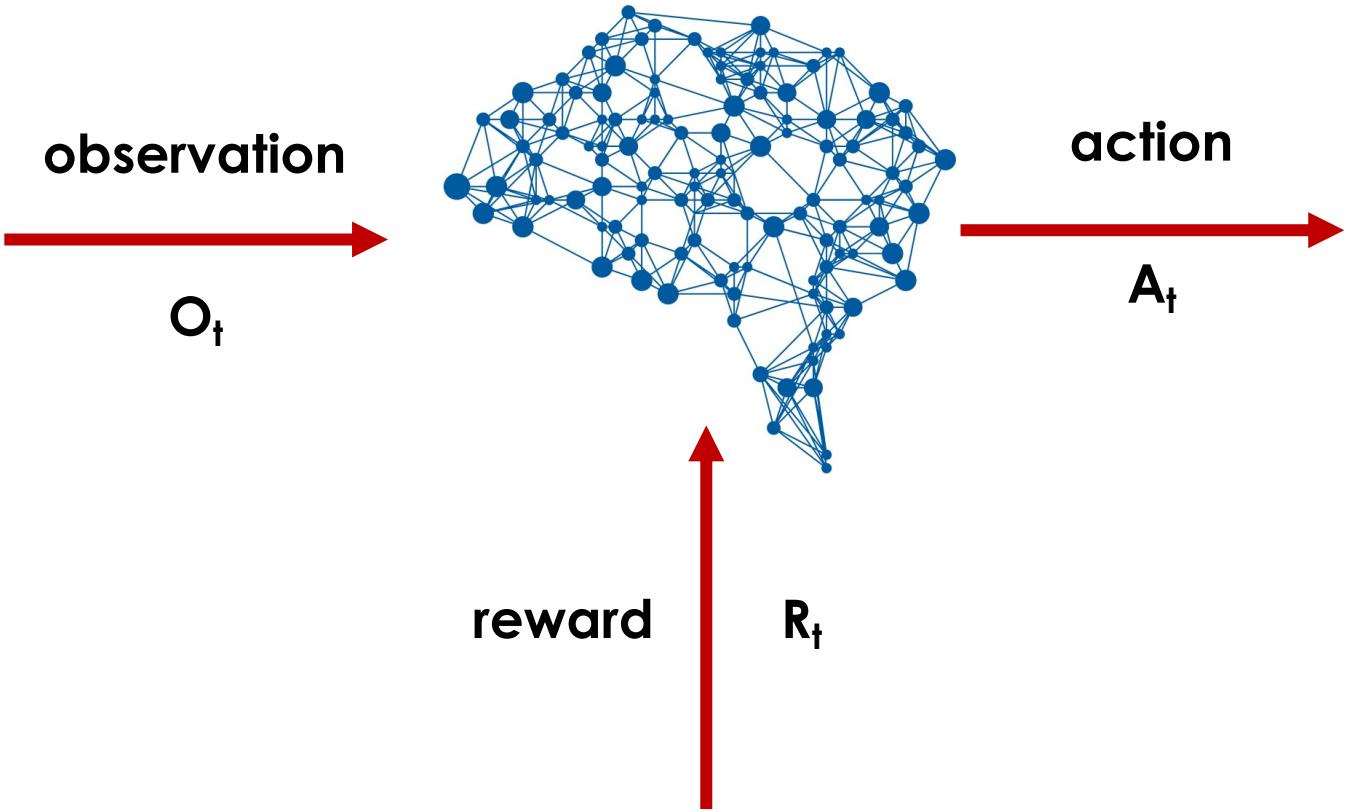
- The dilemma comes from the *incomplete* information: we need to gather enough information to make best overall decisions while keeping the risk under control.
- With exploitation, we take advantage of the best option we know.
- With exploration, we take some risk to collect information about unknown options.
- The best long-term strategy may involve short-term sacrifices. For example, one exploration trial could be a total failure, but it warns us of not taking that action too often in the future.

Exploration-exploitation dilemma

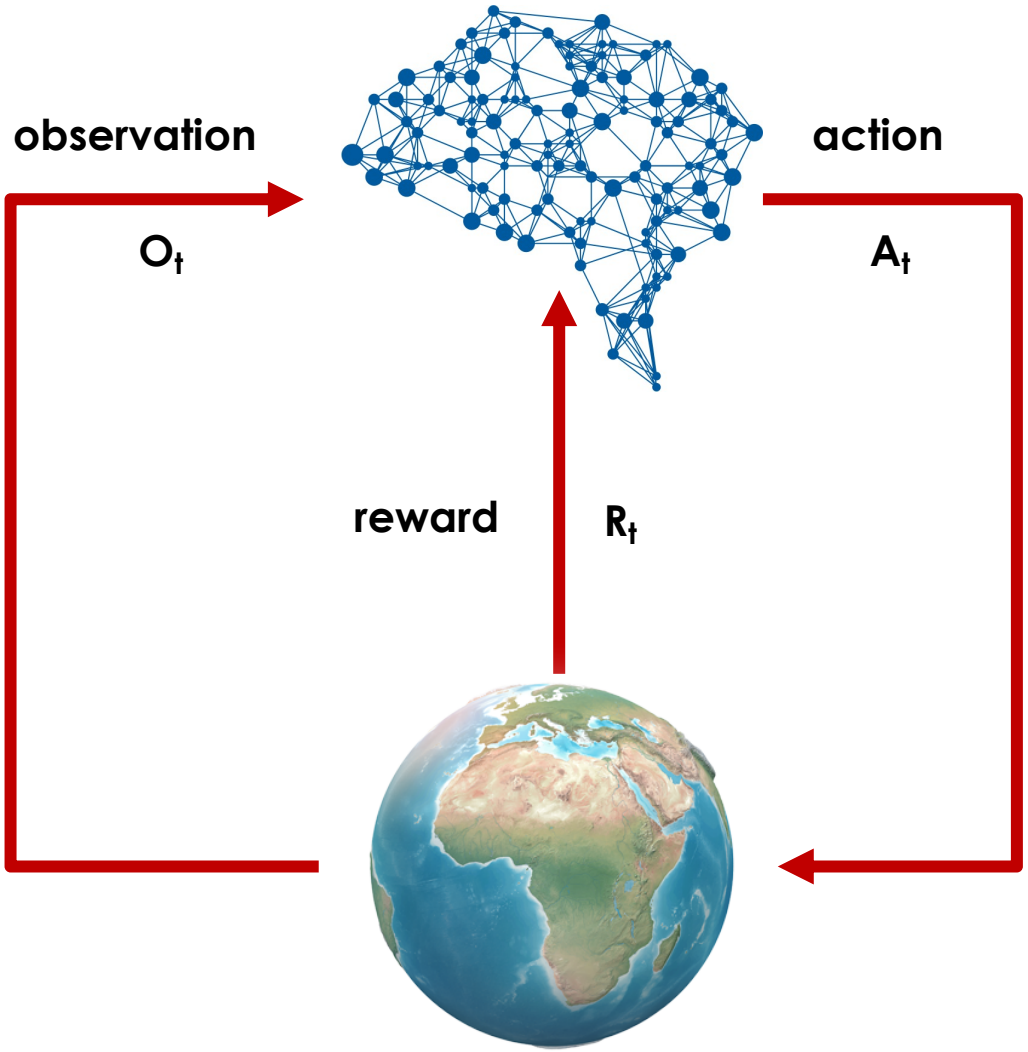
- The dilemma is that neither exploration nor exploitantion can be pursued exclusively without failing at the task
- The agent must try a variety of actions and progressively favor those that appear to be best
- On a stocastic task, each action must be tried many times to gain a reliable estimate of its expected reward
- The exploration-exploitation dilemma has been studied for many decades and yet remains unresolved.

A general RL framework

Agent and Environment



Agent and Environment



- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

History and State

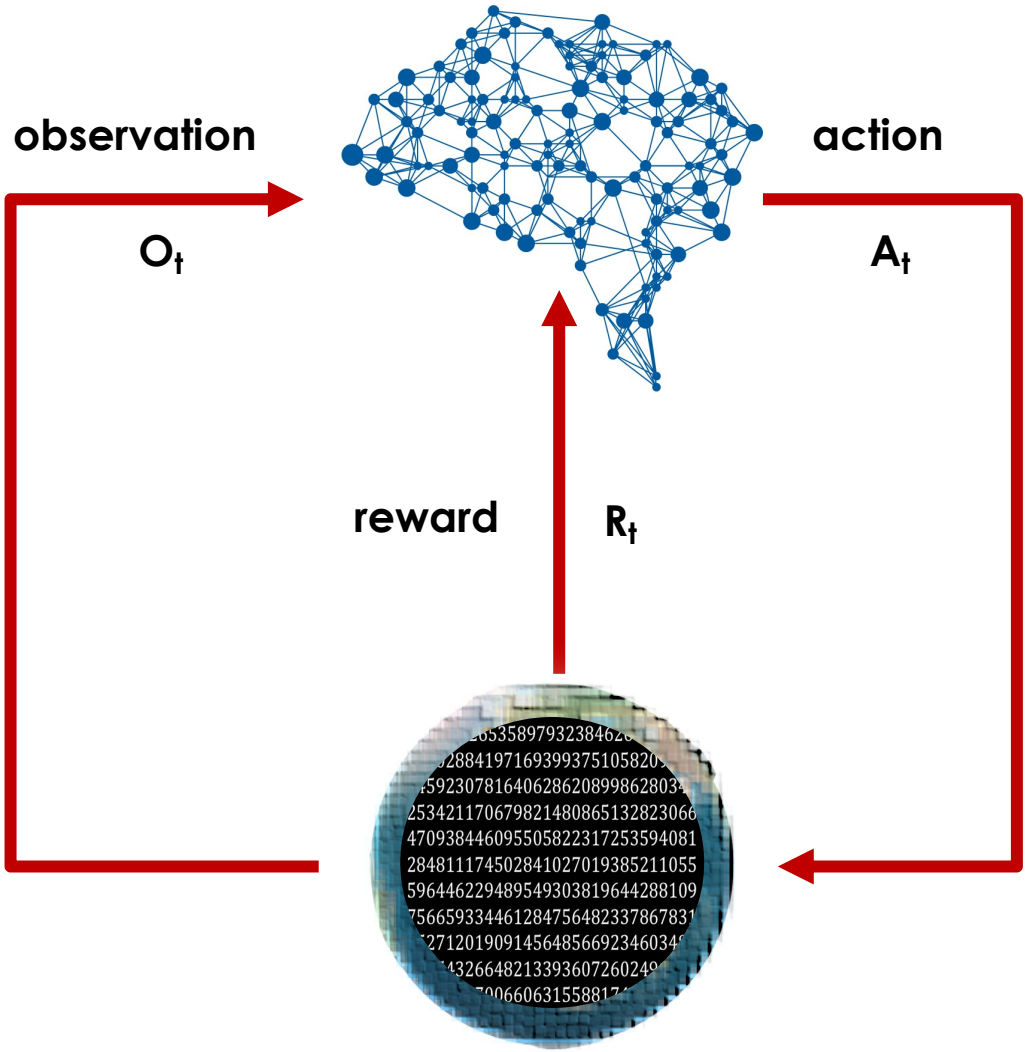
- History: The sequence of observations, actions, rewards

$$H_t = A_1, O_1, R_1, \ldots, A_t, O_t, R_t$$

- i.e. all observable variables up to time t
- i.e. the sensorimotor stream of a robot or embodied agent
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- State is the information used to determine what happens next
- Formally, state is a function of the history:

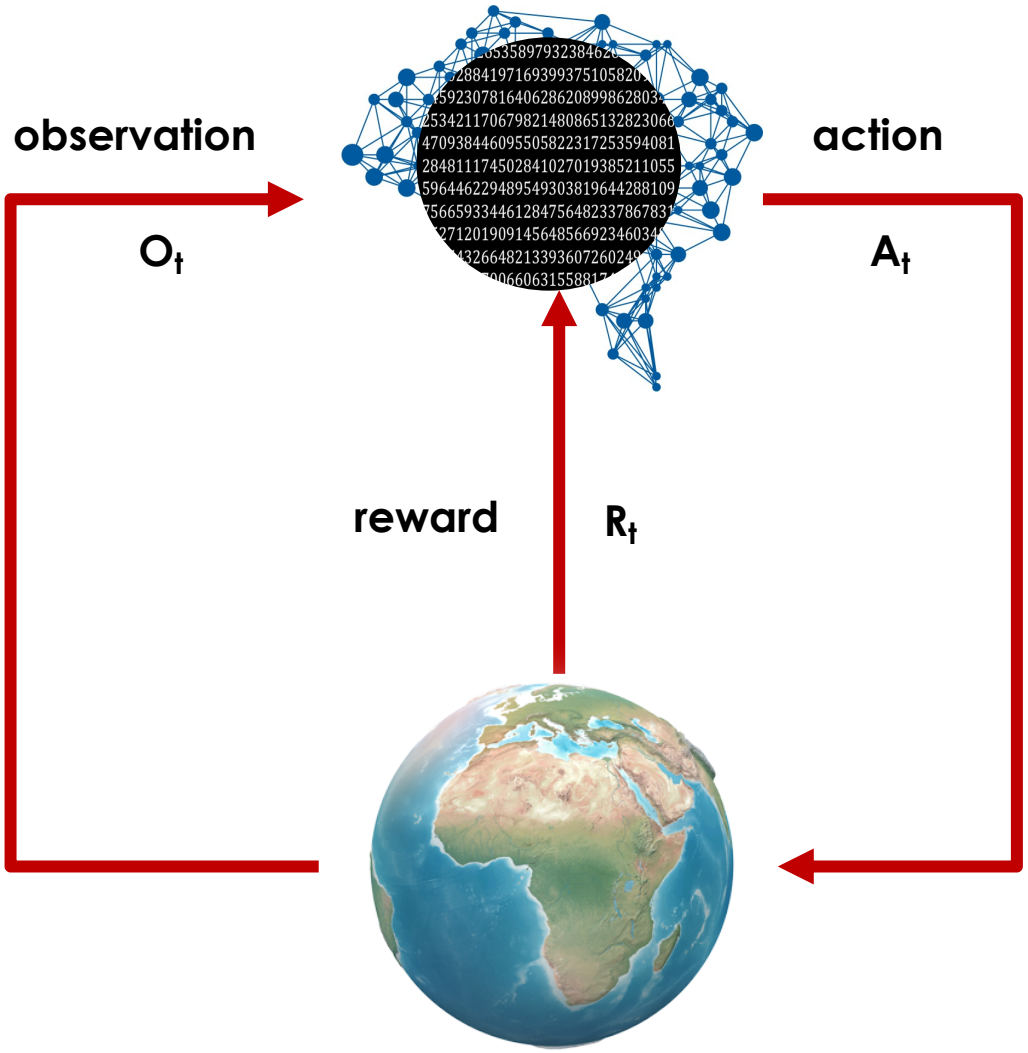
$$S_t = f(H_t)$$

Environment state



- The environment state S_t^e is the environment's private representation
- i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent

Agent state



- The agent state S_a^t is the agent's internal representation
- i.e. whatever information the agent uses to pick the next action
- i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history:

$$S_a^t = f(H_t)$$

Agent state

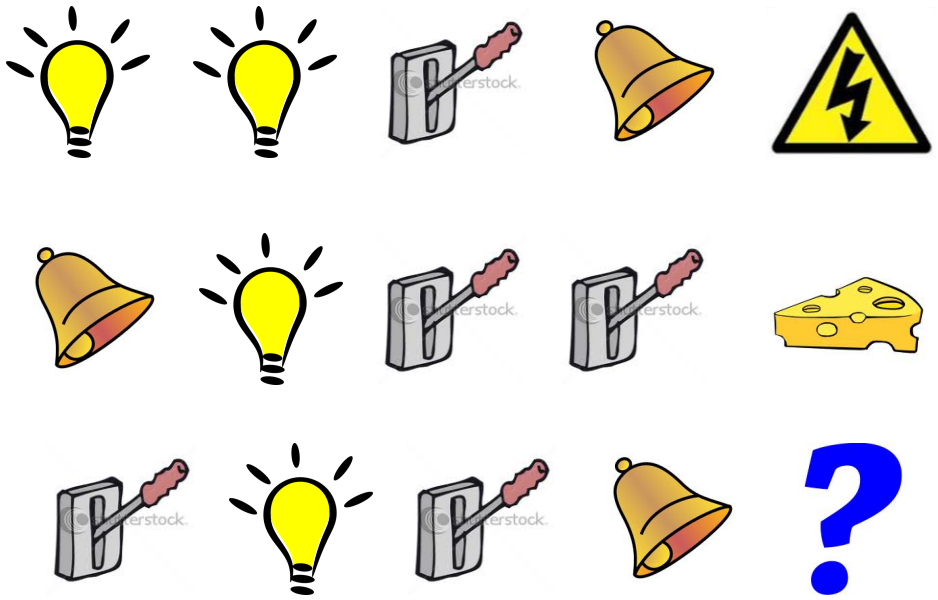
Fully observable elements

- Agent state = environment state

Partially observable elements

- Partial observability: agent indirectly observes environment
  - A robot with camera vision isn't told its absolute location
  - A trading agent only observes current prices
- Now agent state $\neq$ environment state

Rat Example



- What if agent state = last 3 items in sequence?
- What if agent state = counts for lights, bells and levers?
- What if agent state = complete sequence?

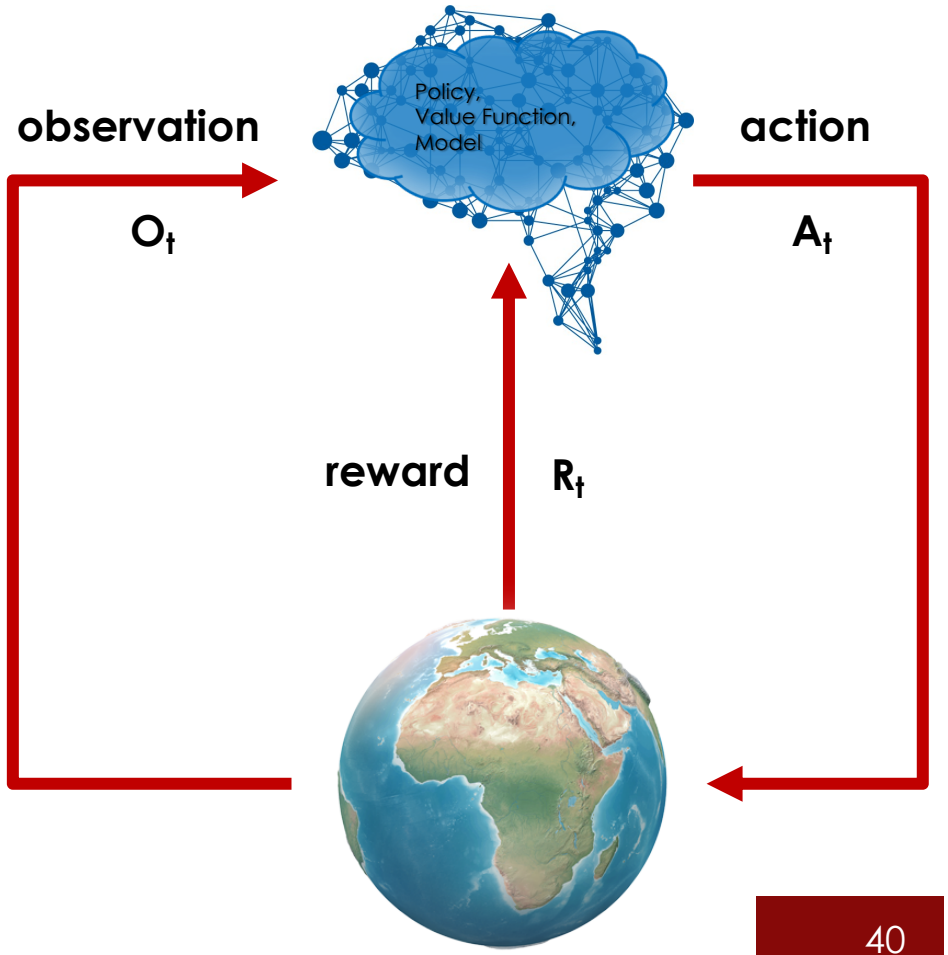
Agent State

- The state representation defines what happens next
- Our job is to build an agent state that is useful in doing the best job in predicting what will happen next
- Agent must construct its own state representation S_t^a, e.g.
  - Complete history of states
  - Beliefs of environment state: assign a probability to each state
  - Function of the history (linear combination of current and previous state)

Inside an RL agent

Major Components of an RL Agent

- An RL agent may include one or more of these components:
- **Policy**: agent's behavior function
- **Value function**: how good is each state and/or action
- **Model**: agent's representation of the environment



Policy (what to do)

- A policy defines the learning agent's behavior at a given time
- It is a **map from state to action**
- The policy is the core of a RL agent in the sense that it alone is sufficient to determine behavior
- The reward signal is the primary basis for altering the policy: if an action selected by the policy is followed by low reward then the policy may be changed to select some other action in that situation in the future
- Policies may be
  - Deterministic – a simple function of the state
  - Stochastic - specifying probabilities for each action

Value function (what is good because it predicts reward)

- Whereas the reward signal indicates what is good in an immediate sense, a value function specifies **what is good in the long run**
- Value function is a prediction of future reward: the total amount of reward an agent can expect to accumulate over the future, starting from that state
- Used to evaluate the goodness/badness of states
- Values are predictions of rewards

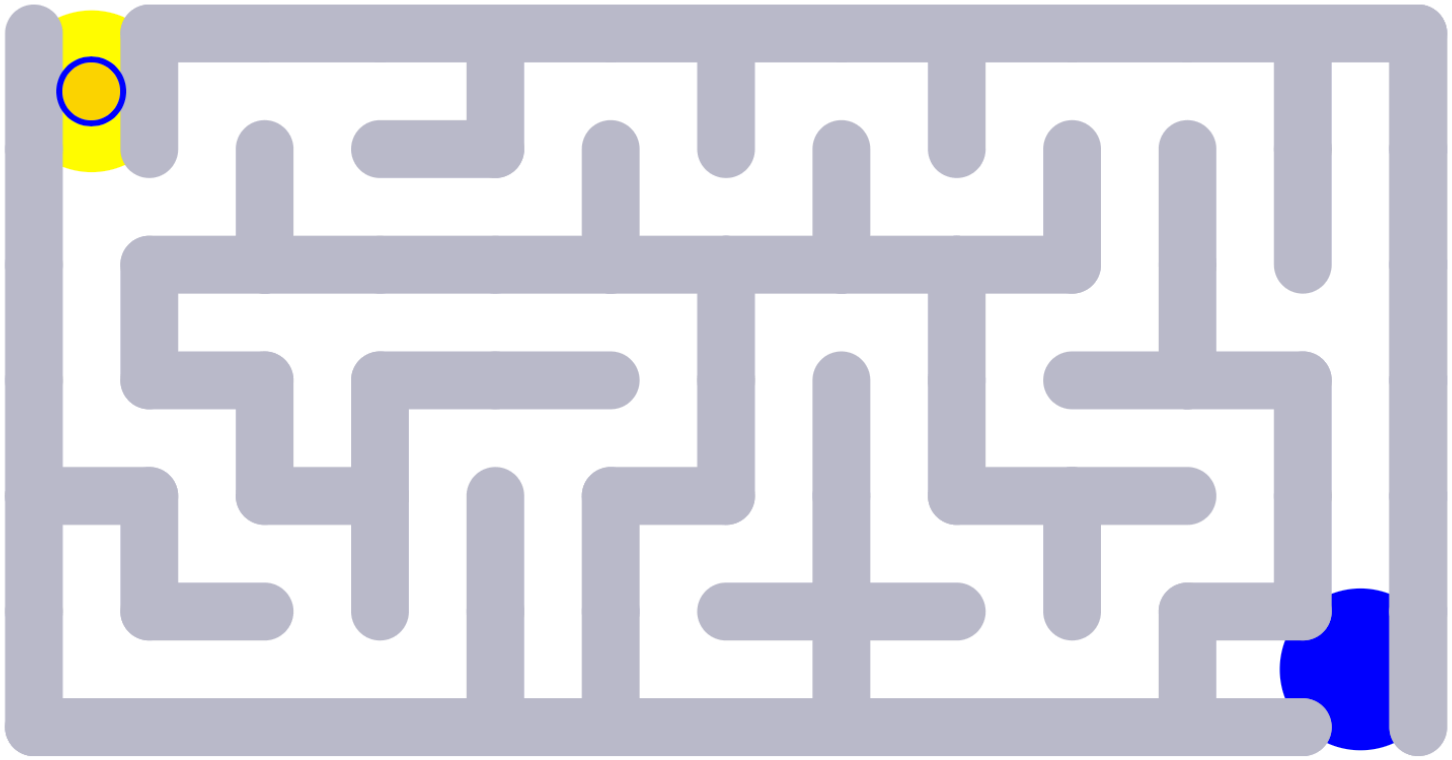
$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

- Example: a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards
- Without rewards there could be no values

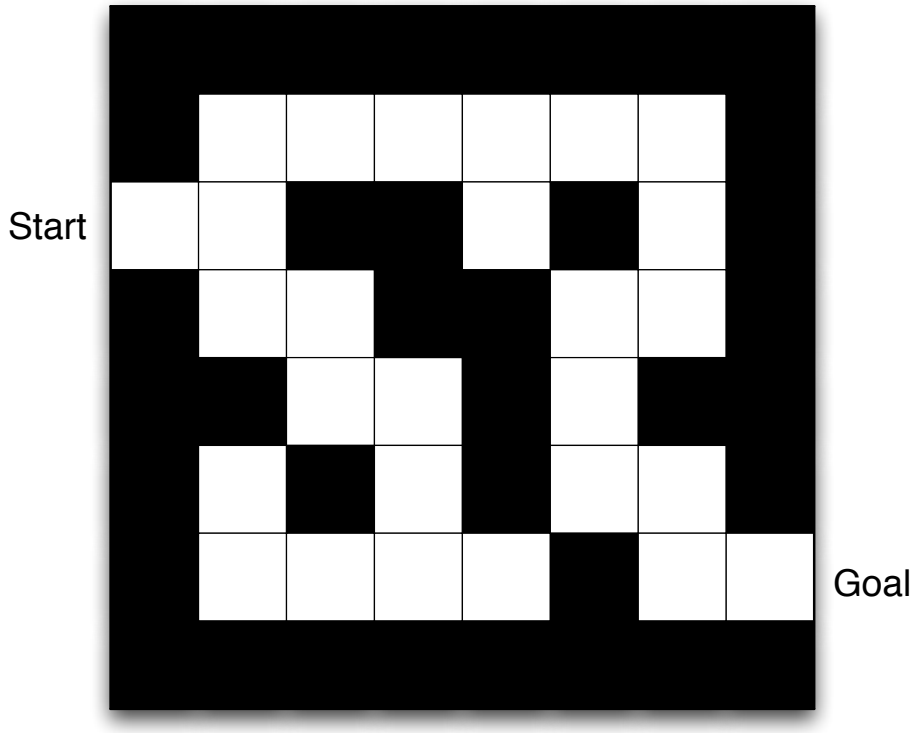
Model of the environment (what follows what)

- A model predicts what the environment will do next
- It may predict the resultant next state
- It may predict the next (immediate) reward
- Many RL problems are model free

Maze game

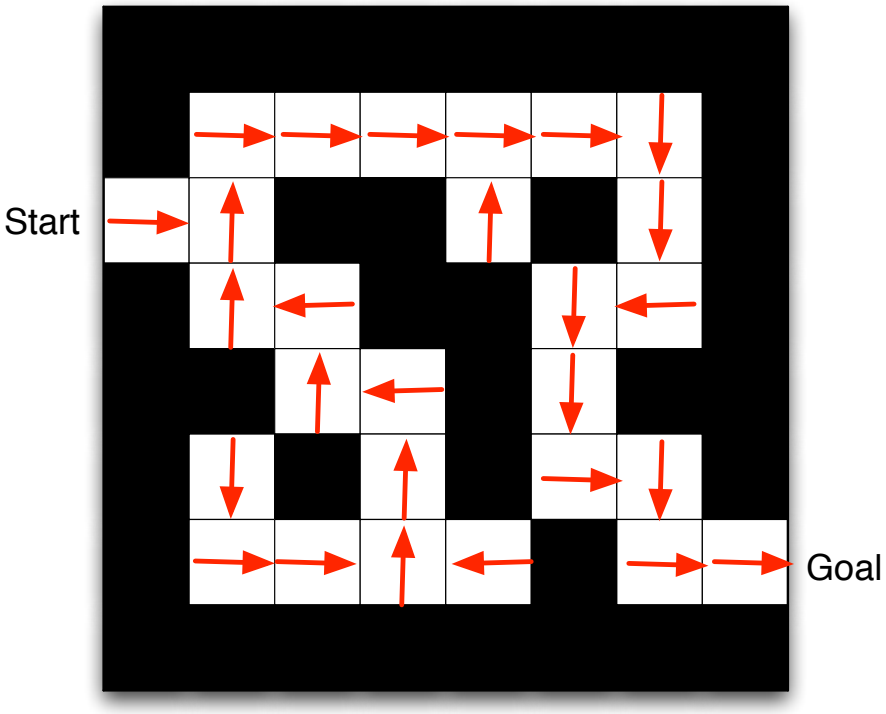


Maze example



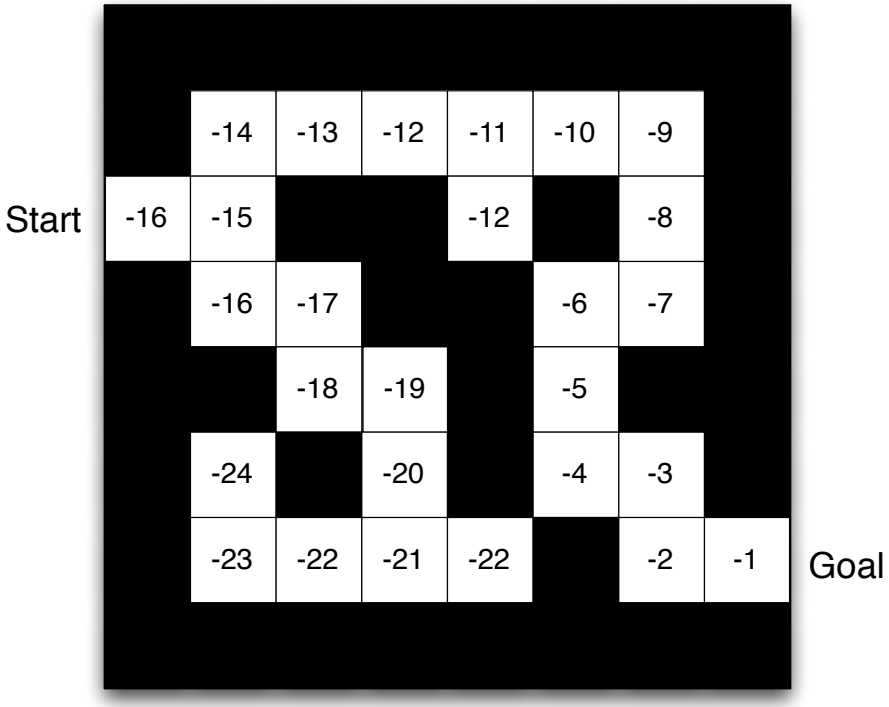
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze example: policy



- Arrows represent policy $\pi(s)$ for each state s

Maze example: value function



- Numbers represent value $v_{\pi}(s)$ of each state s

Problem: communication in a battery-free smart home

- Consider a battery-free smart home with many (*n*) heteroheneous devices
- Each device *i (i=1,…,n)* produces new data with $Rate_i$ that depends on the environment
  - A joystick produces new data only when is used
  - Temperature depends on current state (normal condition, fire, etc.)
  - Presence sensor produces new data only when a person walks in front of it
  - Videocamera contiuously produces new data
- Devices need to be queried by the reader
- Reader can query devices sequentially (one at each time step)
- Assume prefixed timeslots

Problem: communication in a battery-free smart home

- *Problem:*

In which order should the reader query tags?

We want to query a device that has a new data sample and avoid loss of data

- Formulate the problem as a RL problem
  - Agent ?
  - Actions ?
  - Rewards ?
  - State ?
  - Etc.