Principles of Autonomy and Decision Making

(AI_PrincAutonomy_2808)

Week 8: Reinforcement Learning

Guest Lecturer: Zahra Zeinaly, Ph.D.

Ref: Adapted from RL Course by David Silver

Technische Hochschule Ingolstadt

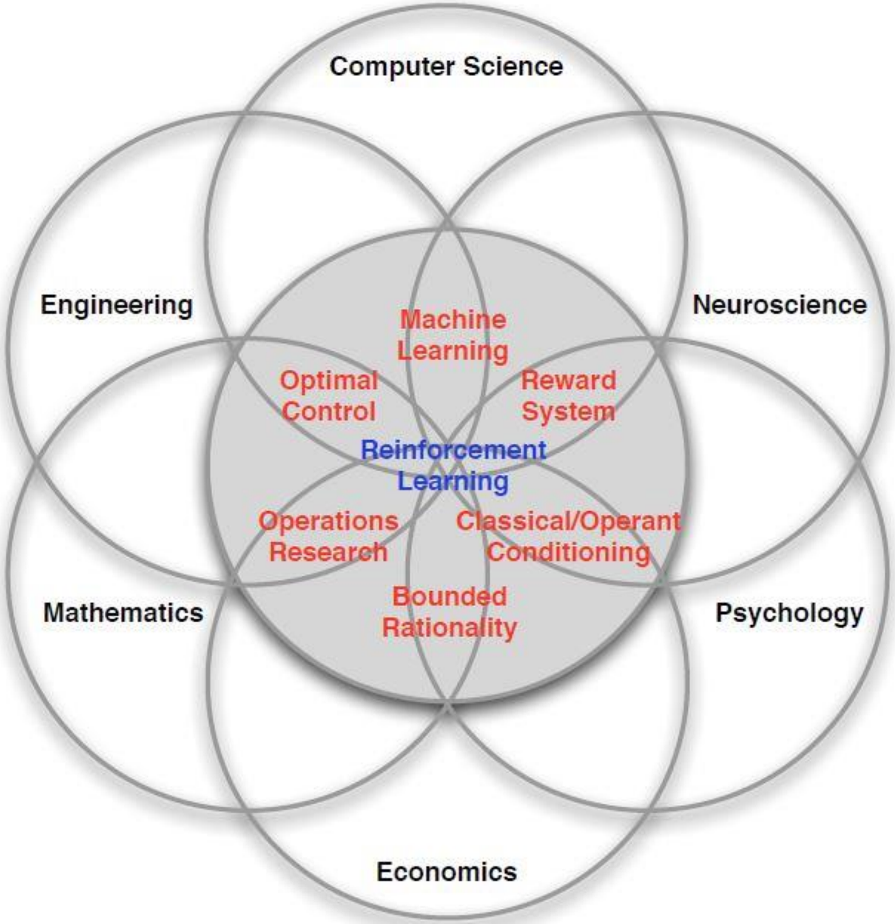
Team:

- Prof. Dr. rer. nat. Lenz Belzner
- Chidvilas Karpenahalli Ramakrishna, M.Eng.
- Adithya Mohan, M.Eng.
- Zahra Zeinaly, Ph.D.

Contents

- About Reinforcement Learning problem
- The Reinforcement Learning Formalism
- Inside an RL agent
- Problems within Reinforcement Learning

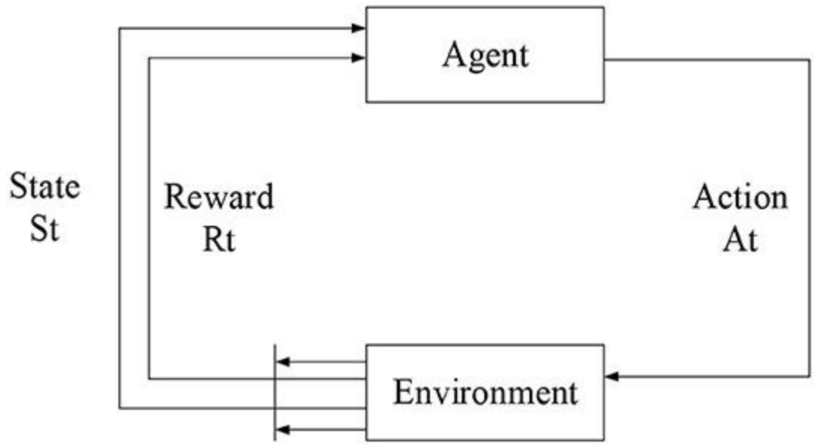
Many Faces of Reinforcement learning



Ref: RL course by David Silver

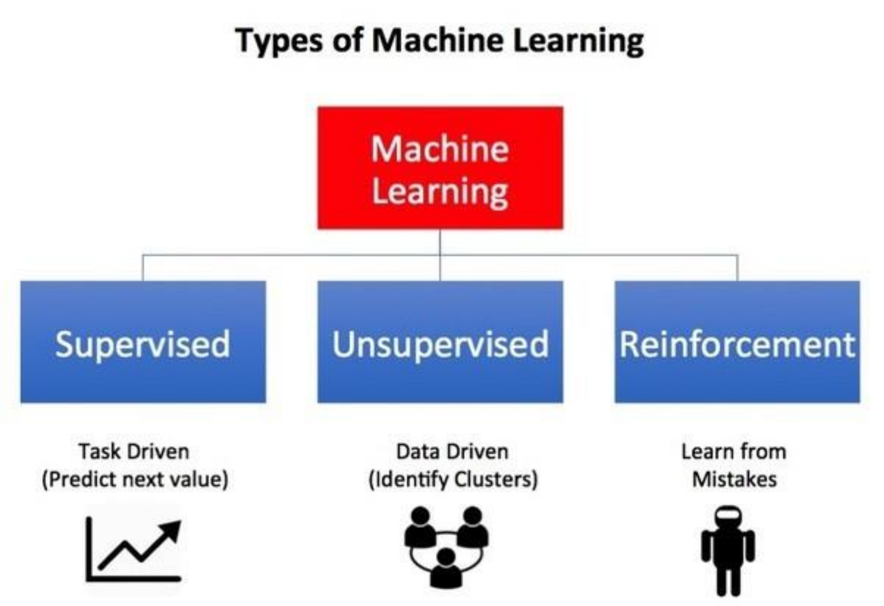
Reinforcement Learning in a nutshell

- RL is a general-purpose framework for decision-making
  - RL is for an agent with the capacity to act
  - Each action influences the agent's future state
  - Success is measured by a scalar reward signal
  - Goal: select actions to maximise future reward
  - Learning rather than direct planning



What makes reinforcement learning different from other machin learning paradigms?

- There is no supervisor, only a reward signal.
- The agent learns by interacting with environment.
- The Learning can be done without examples of optimal behaviour.
- Feedback is delayed, not instantaneous.
- Time really matters (sequential, non i.i.d data).
- Agent's actions affect the subsequent data it receives.



Examples of Reinforcement Learning

- Fly manoeuvres in a helicopter
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Play many different video games better than humans

Helicopter Manoeuvres

- http://heli.stanford.edu/

Paper: **Autonomous Helicopter Aerobatics through Apprenticeship Learning**, Pieter Abbeel, Adam Coates, and Andrew Y. Ng, 2010

Traffic Light Control

Make a robot walk

https://www.dropbox.com/s/fdn1loibsh2p0sa/parkour.mp4?e=1&dl=0

Rewards

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

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots$$

- We call this the return

Reinforcement learning is based on the reward hypothesis.

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

Examples of Rewards

- Fly manoeuvres in a helicopter
  - + reward for following desired trajectory
  - - reward for crashing
- Manage an investment portfolio
  - + reward for each $ in bank
- Control a power station
  - + reward for producing power
  - - reward for exceeding safety thresholds
- Make a humanoid robot walk
  - + reward for forward motion
  - - reward for falling over
- Play many different video games better than humans
  - +/- reward for increasing/decreasing score

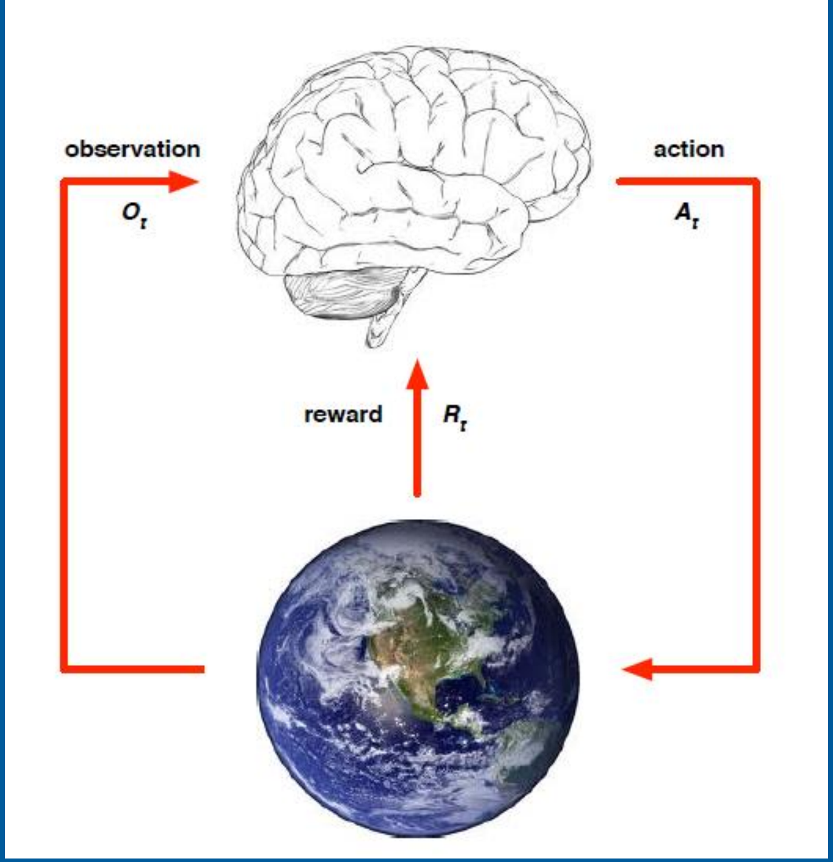
Sequential Decision Making

- 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)

Agent and Environment

- At each step t the agent:
  - Receives Observation O_t
  - Executes Action A_t
  - Receves Reward R_t

- The environment
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits reward R_{t+1}
- t increments at env. step



Ref: RL course by David Silver

History and state

- The **history** is the full sequence of observations, actions, rewards

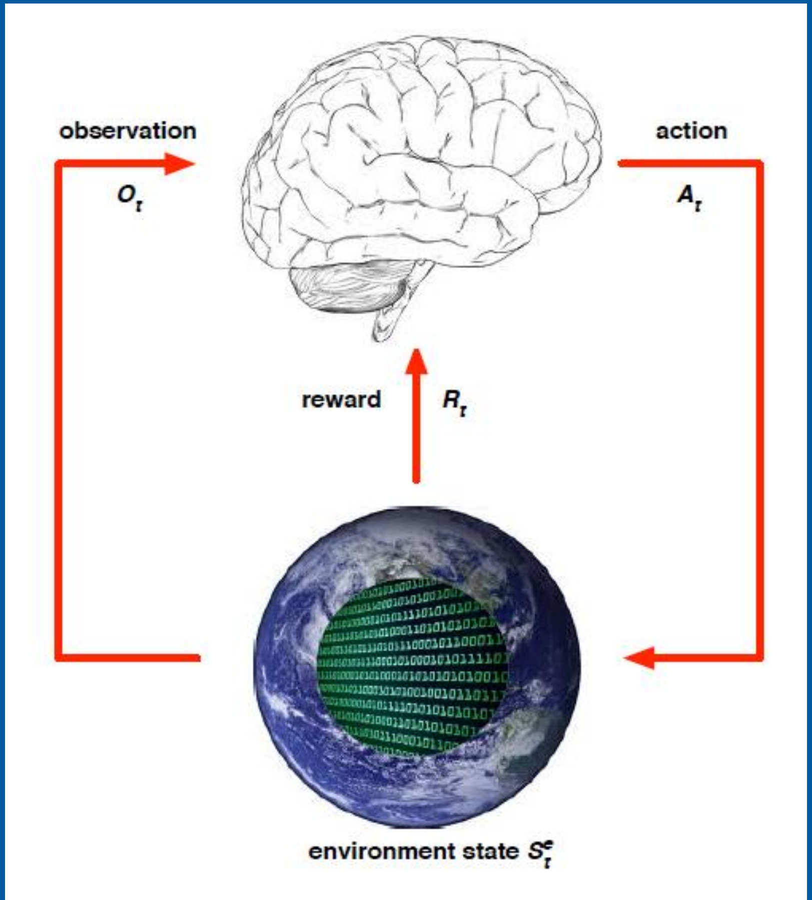
$$H_t = O_0, A_0, R_1, O_1, \ldots, O_{t-1}, A_{t-1}, R_t, O_t$$

- i.e. all observable variables up to time t
- 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 is the environment’s internal state
- It is usually invisible to the agent
- Even if it is visible, it may contain lots of irrelevant information

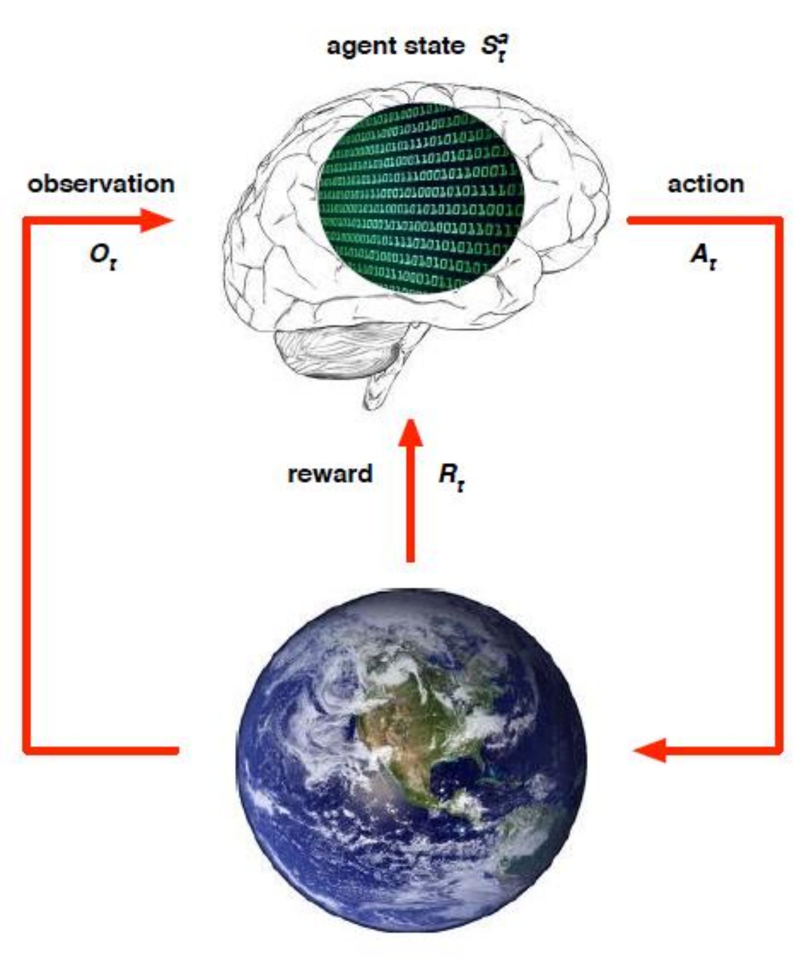


Ref: RL course by David Silver

Agent state

- The agent state S_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
- Agent State is the information used to determine what happens next
- It can be any function of history:

$$S_t = f(H_t)$$



Ref: RL course by David Silver

Information state

- An information state (a.k.a. Markov state) contains all useful information from the history.

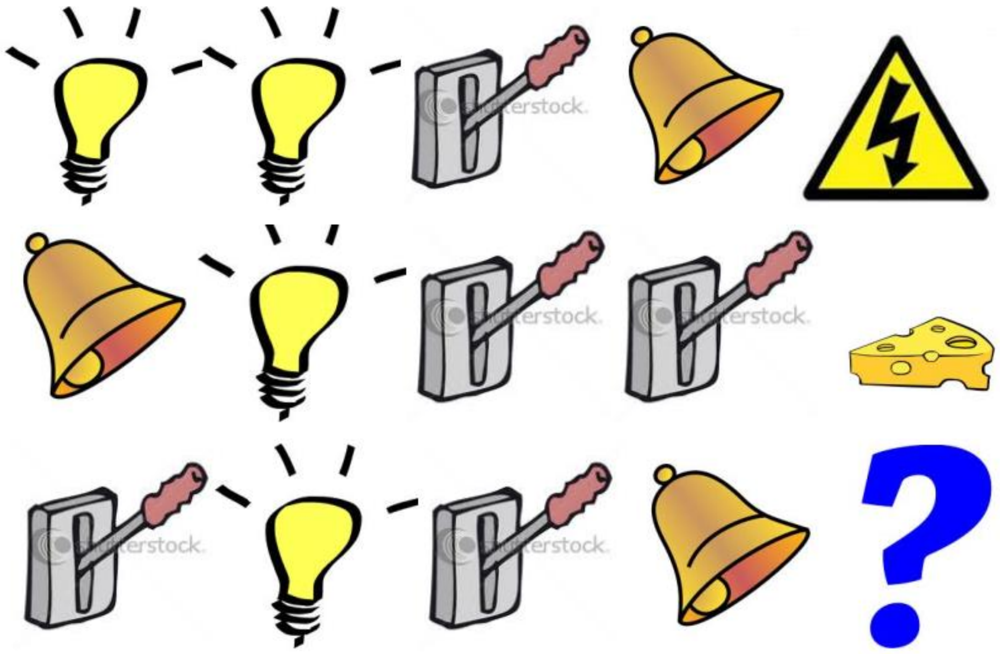
Definition

A state S_t is Markov if and only if

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \ldots, S_t]$$

- The future is independent of the past given the present
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future
- The history H_t is Markov.

Rat Example



Ref: RL course by David Silver

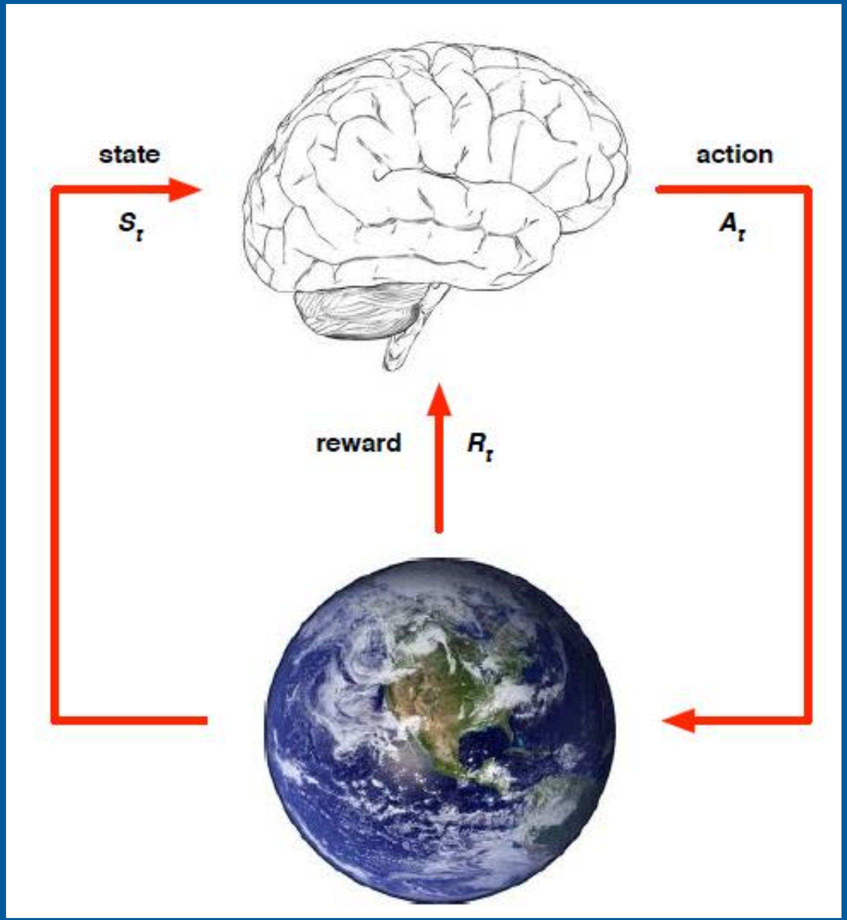
- What if agent state = last 3 items in sequence?
- What if agent state = counts for lights, bells and levers?
- What if agent state = complete sequence?

Fully observable environment

- Agent directly observes environment state
- Agent state= environment state= information state

  $S_t = O_t$= environment state

- Formally, this is a Markov decision process (MDP)



Ref: RL course by David Silver

Partially Observable Environment

- Partial observability: agent indirectly observes environment:
  - A robot with camera vision isn't told its absolute location
  - A poker playing agent only observes public cards
- Now agent state ≠ environment state
- using the observation as state would not be Markovian
- Formally this is a partially observable Markov decision process (POMDP)

Major Components of an RL Agent

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

Policy

- A policy defines the agent's behaviour
- It is a map from state to action
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}(A_t = a | S_t = s)$

Value Function

- Value function is a prediction of future reward
- Can be used to evaluate the goodness/badness of states
- Can be used to select between actions
- The actual value function is the expected return

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

- Discount factor $\gamma \in [0,1]$: Trades off importance of immediate vs long-term rewards

Model

- A model predicts what the environment will do next
- Predicts the next state

$$\mathcal{P}_{s\acute{s}}^{a} = \mathbb{P}[S_{t+1} = \acute{s} | S_t = s, A_t = a]$$

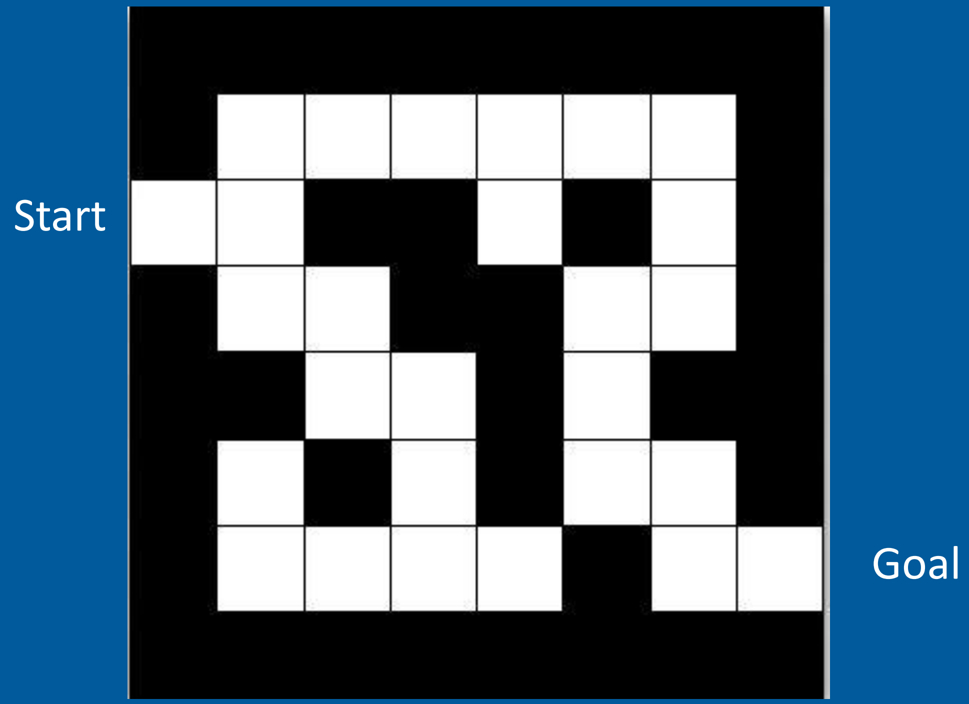
- Predicts the next reward

$$\mathcal{R}_{s}^{a} = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

- A model does not immediately give us a good policy - we would still need to plan

Maze Example

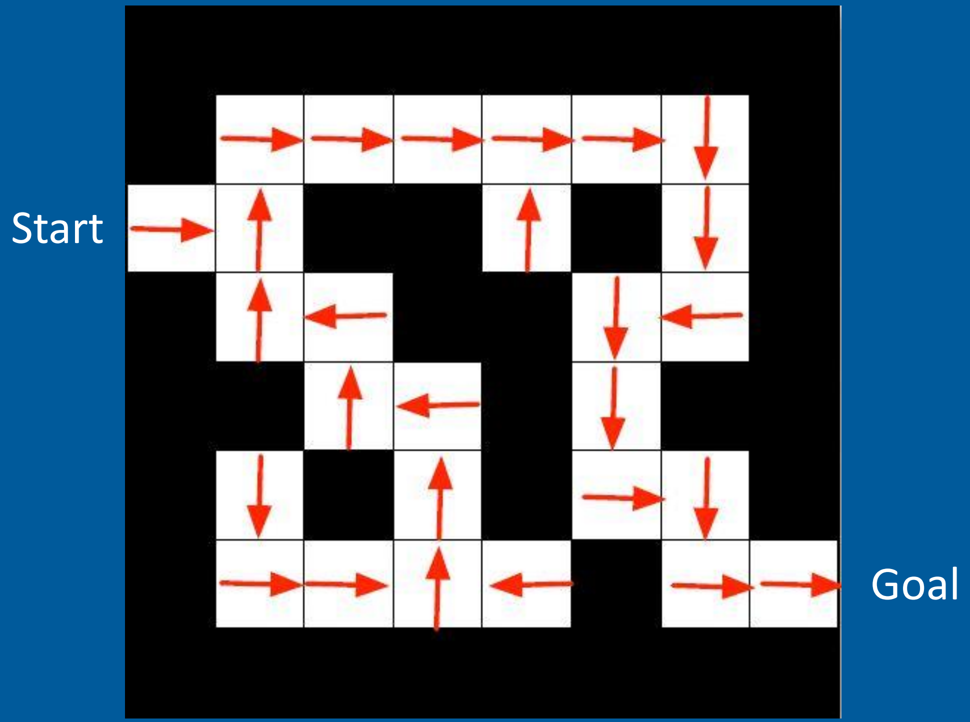
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent’s location



Ref: RL course by David Silver

Maze Example: Policy

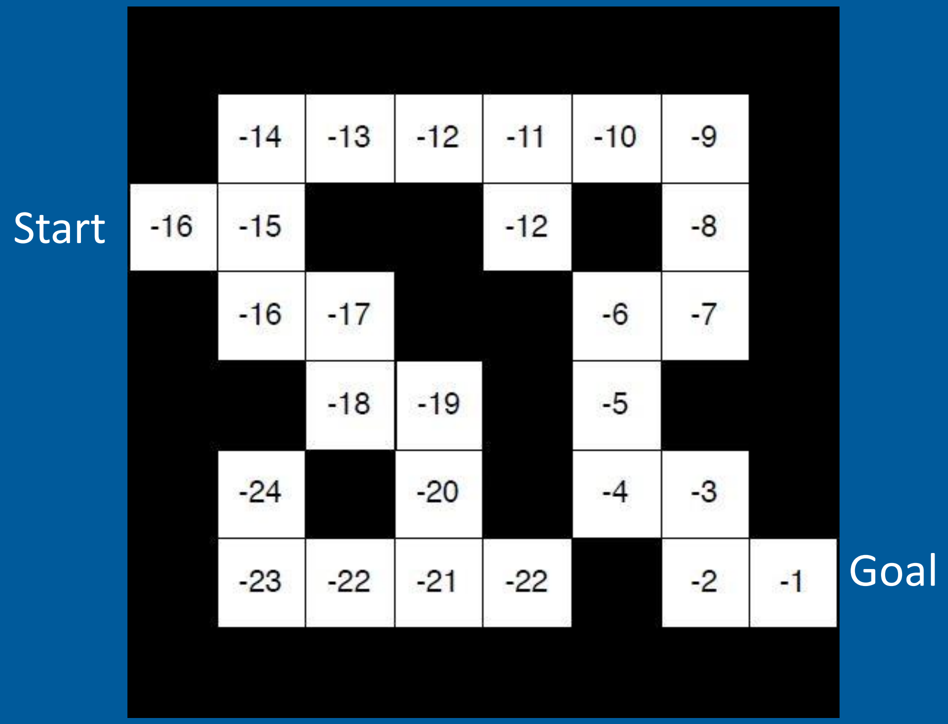
- Arrows represent policy $\pi(s)$ for each state s



Ref: RL course by David Silver

Maze Example: Value function

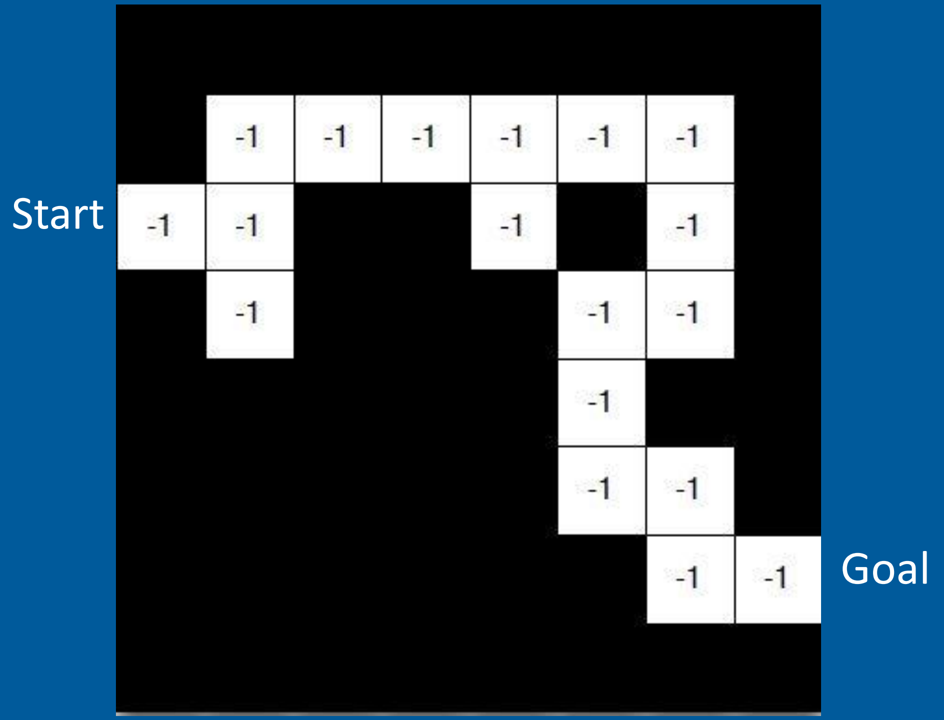
- Numbers represent value $v_\pi(s)$ of each state s



Ref: RL course by David Silver

Maze Example: Model

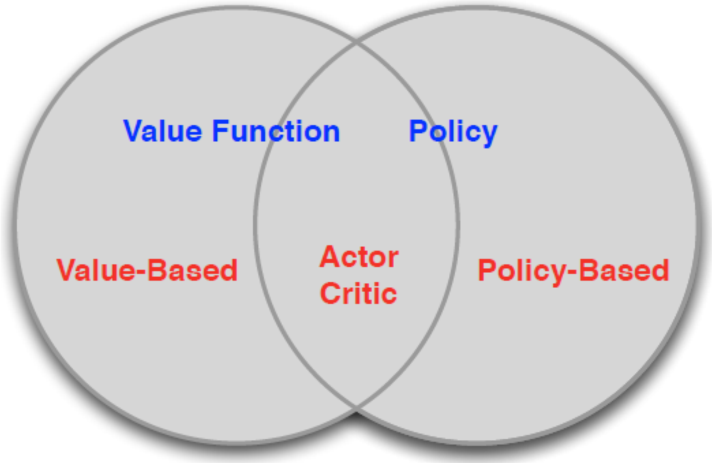
- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- Grid layout represents transition model $\mathcal{P}_{s\acute{s}}^{a}$
- Numbers represent immediate reward $\mathcal{R}_{s}^{a}$ from each state s (same for all a and $\acute{s}$ in this case)



Ref: RL course by David Silver

Categorizing RL agents(1): Value based and Policy based

- Value- based : will determine a value function that quantifies the reward and using this value function we determine the optimal policy
  - No policy(implicit)
  - Value function
  - Q- learning
  - Deep Q network
  - SARSA
- Policy based: will determine an optimal policy directly which means the policy that maximizes reward
  - Policy
  - No value function
  - REINFORCE
  - PPO (Proximal Policy Optimization)
  - TRPO (Trust Region Policy Optimization)
- Actor Critic
  - Policy network (Actor)
  - Value function (Critic)

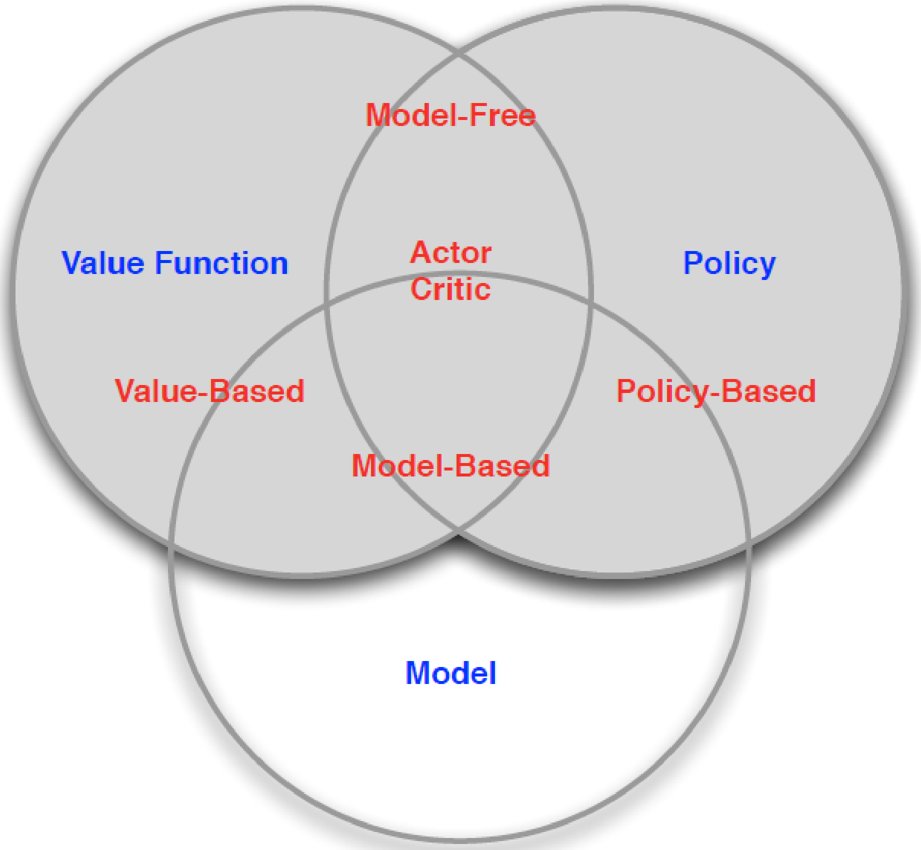


Ref: RL course by David Silver

Categorizing RL agents(2): Model free and Model Based

- Model free
  - Policy and/or Value Function
  - No Model
- Model based
  - Policy and/or Value Function
  - Model

RL Agent taxonomy

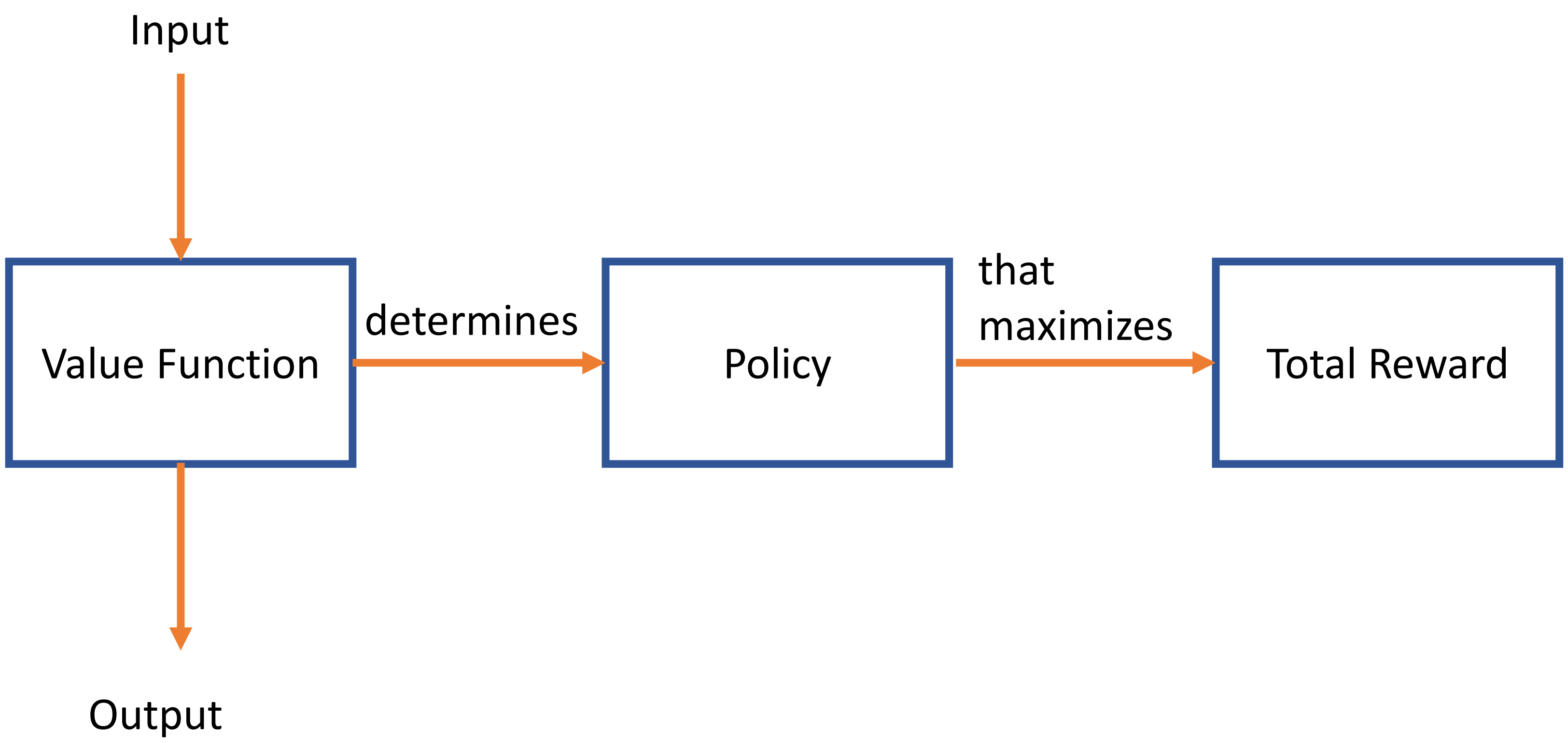


Ref: RL course by David Silver

Summary: Key concepts in RL

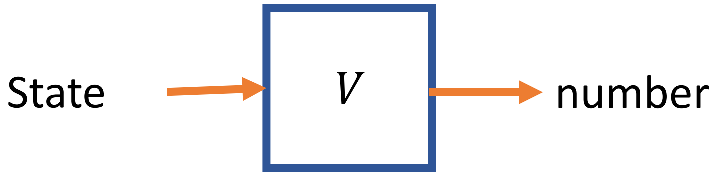
- Environment: Physical world in which the agent operates
- State: Current situation of the agent
- Reward: Positive or negative feedback from the environment
- Policy: The rules that change agent’s state to actions
- Value: Future reward that an agent would receive

Value Based Methods



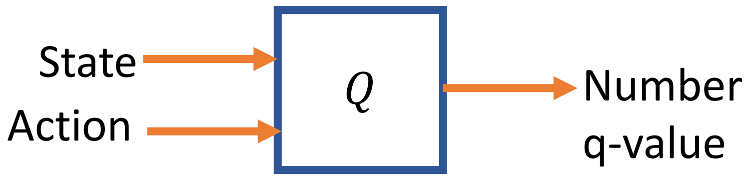
Value Functions

State- value functions $V(s)$



- How good is it to be in the state s

State-action value functions $Q(s, a)$



- How good is it to be in the state s and take an action a in this state

Bellman Equation

- fundamental concept in dynamic programming and reinforcement learning, named after Richard Bellman, who introduced it in the 1950s
- It provides a recursive decomposition for solving optimization problems, particularly those involving decision-making over time.
- the Bellman equation is used to describe the relationship between the value of a state and the values of subsequent states.
- Bellman equation provides a way to compute the value of each state (or state-action pair) recursively by considering the expected rewards and the values of subsequent states.

- Bellman Expectation Equation
- Bellman optimality Equation

Bellman Expectation Equations

- For value function

$$V_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) = |S_t = s]$$

  - $V_\pi(s)$ is the value of state s under policy π
  - $\mathbb{E}_\pi$ is the expected value given that the agent follows policy π
  - R_{t+1} is the reward received after transitioning from state S to state S_{t+1}

- For Q-values

$$Q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma Q_\pi(S_{t+1}\,, A_{t+1}) = |S_t = s, A_t = a]$$

Bellman Optimality Equations

- For value function
  - For the optimal value function $V^*(s)$, which represents the maximum expected return achievable from state s, the Bellman optimality equation is:

$$V^*(s) = max_a \mathbb{E}[R_{t+1} + \gamma V^*(S_{t+1}) = |S_t = s, A_t = a]$$

  - $V^*(s)$ is the optimal value of state s
  - max_a is the maximization over all possible actions a.
  - The expectation $\mathbb{E}$ is taken over the possible next states S_{t+1} and rewards R_{t+1} , given action A_t $= a$ in state s

- For Q-values

$$Q^*(s, a) = \mathbb{E}[R_{t+1} + \gamma max_{\acute{a}}(S_{t+1}\,, \acute{a}) = |S_t = s, A_t = a]$$

Exploration and Exploitation

- Reinforcement learning is like trial-and-error learning
- The agent should discover a good policy
- From its experiences of the environment
- Without losing too much reward along the way
- Exploration finds more information about the environment
  - trying out new actions that may not be the best according to the agent's current knowledge, but could potentially lead to discovering better long-term strategies.
- Exploitation exploits known information to maximise reward
  - The agent uses its current knowledge to choose actions that it believes will give the highest reward based on past experiences.
- It is usually important to explore as well as exploit

Examples

- Restaurant Selection

Exploitation Go to your favourite restaurant

Exploration Go to new restaurant

- Online Banner Advertisements

Exploitation Show the most successful advert

Exploration Show a different advert

- Oil Drilling

Exploitation Drill at the best known location

Exploration Drill at a new location

- Game Playing

Exploitation Play the move you believe is best

Exploration Play an experimental move

The Exploration-Exploitation Trade-off

- RL agent needs to make decisions about whether to use known strategies to get immediate rewards (exploitation) or try new strategies that might lead to better rewards in the future (exploration).
- Too much exploitation: The agent might not find better strategies that could improve its performance in the long run, resulting in less effective outcomes over time.
- Too much exploration: The agent may spend too much time trying new actions, resulting in lower immediate rewards and slow learning.
- Balancing these two approaches is known as the exploration-exploitation trade-off.
- **Epsilon-Greedy Strategy:**
  - select a random action with probability ϵ (exploration)
  - Select the best-known action with probability $1-\epsilon$ (exploitation). $a = \underset{a \in A}{\operatorname{argmax}}\, Q(a)$
  - Lower ϵ over time: Often, ϵ starts high to encourage exploration and gradually decreases to shift towards more exploitation.

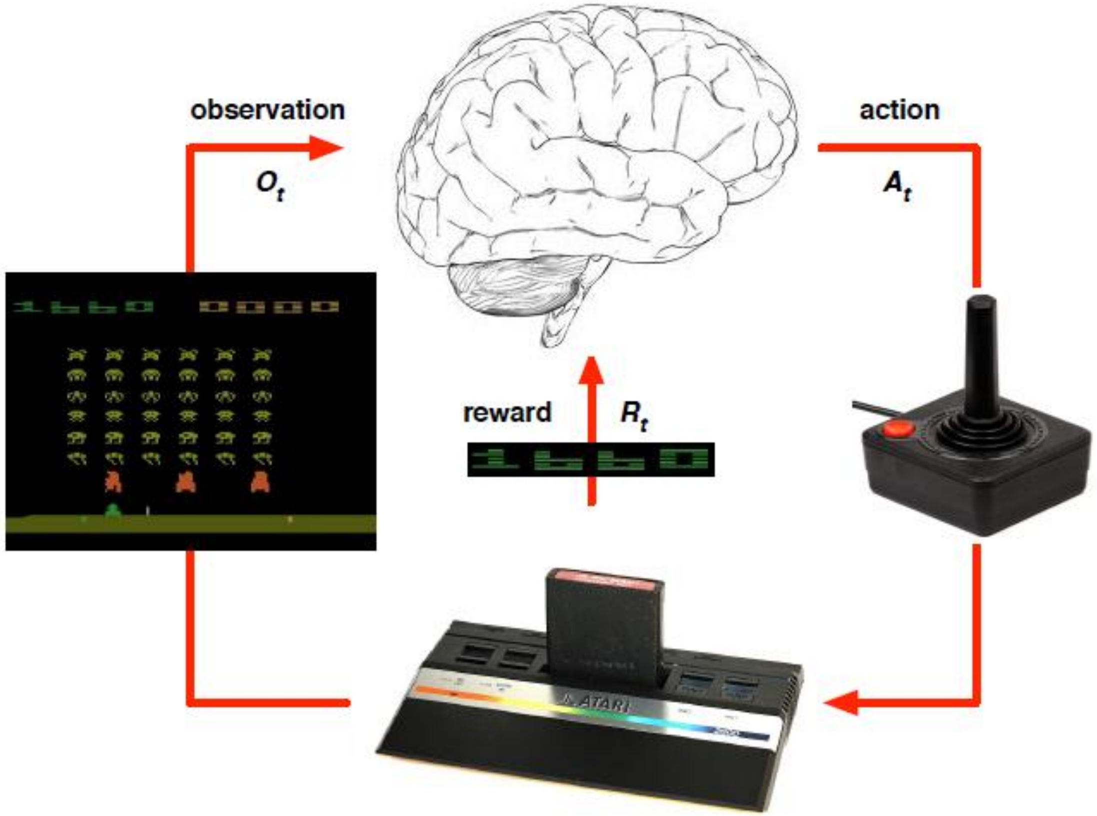
Learning and Planning

Two fundamental problems in sequential decision making

- Reinforcement Learning
  - The environment is initially unknown
  - The agent interacts with the environment
  - The agent improves its policy
- Planning
  - A model of the environment is known
  - The agent performs computations with its model (without any external interaction)
  - The agent improves its policy
  - a.k.a. reasoning, thought, search, planning

Atari Example: Reinforcement Learning

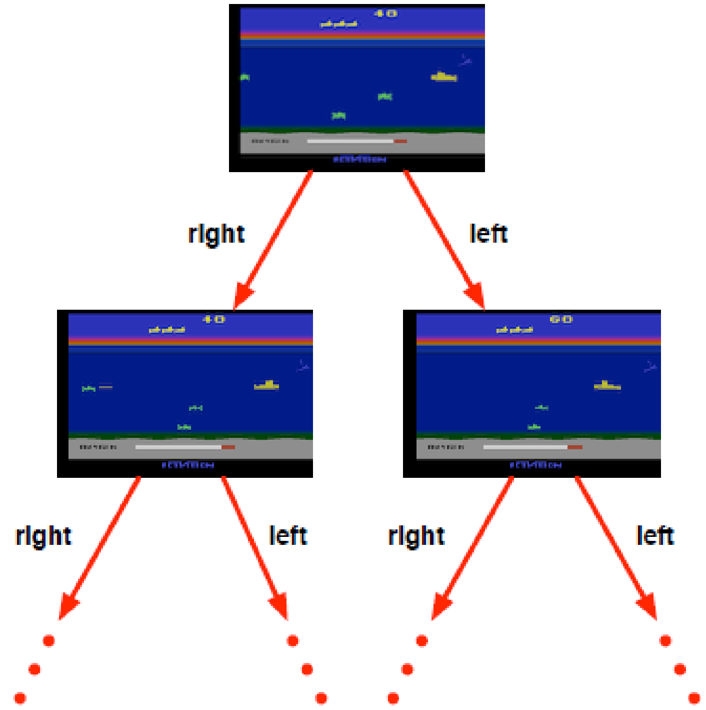
- Rules of the game are unknown
- Learn directly from interactive game-play
- Pick actions on joystick, see pixels and scores



Ref: RL course by David Silver

Atari Example: Planning

- Rules of the game are known
- Can query emulator
  - perfect model inside agent's brain
- If I take action a from state s:
  - what would the next state be?
  - what would the score be?
- Plan ahead to find optimal policy
  - e.g. tree search



Ref: RL course by David Silver