Silver Class and Sutton Points: Episode 1

Introduction to Reinforcement Learning Lecture 1

★ What makes reinforcement learning different from other machine learning algorithms?

→ There is no supervisor, only a signal called reward.

→ Feed back is delayed, not instantaneous.

You make a decision now and it may be you know many many steps later that you actually see whether that was a good decision or a bad decision.

→ Time really matters. (sequential)

↳ step after step.

→ Agent's action affect the subsequent data it recieves.

★ 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.

Reinforcement Learning is based on the reward hypothesis.

↳ Definition: All goals can be described by the maximisation of expected cumulative reward.

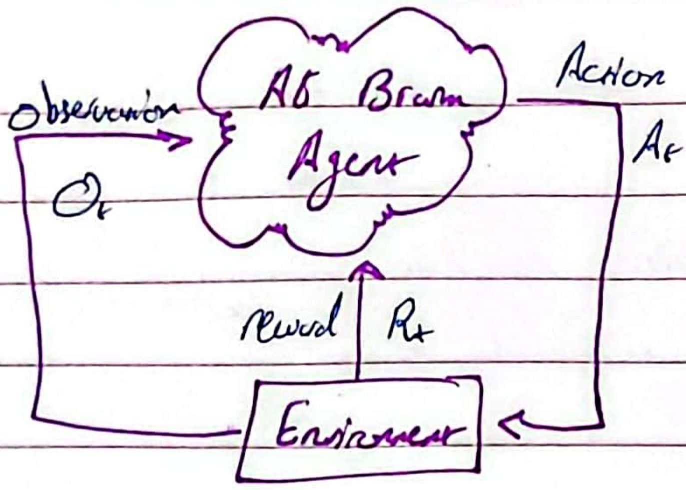
Goal: select actions to maximise total future reward

Actions may have long term consequences.

Reward may be delayed.

It may be better to sacrifice immediate reward to gain more long-term reward.

ex financial investment.



★ At each step t the agent:

- ★ Execute action A_t
- ★ Recieves observation O_t
- ★ Recieves scalar reward R_t

★ The environment:

- ★ Recieves action A_t
- ★ Emits observation O_t
- ★ Emits scalar reward R_t

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

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

all observable variables up to time t.
the sensorimotor stream of a robot or embodied agent
what happens next depend on the history.

- The agent selects actions.
- The environment selects observations/rewards

Point: The history isn't very useful because it's typically enormous, we want to have agents that have long lives and can deal with microsec interactions and each of these observations might a video and we don't want to have to go back to this history every time and typically, what we talk about is **State**.

State: Is like a summary of the information that's used to determine what happens next.
→ Formally, State is a function of the history:

$$S_t = f(H_t)$$

s.a.m

The environment state S_t^e is the environment's private representation.
↳ is basically the information that's used within the environment to determine what happens next.
what the state environment is in.

* whatever data the environment uses to pick the next observation/reward
* The environment state is not usually visible to the agent.
* Even if S_t^e is visible, it may contain irrelevant information.

* The agent state S_t^a is the agent's internal representation.
↳ whatever information we choose to store and capture in our agent or RL algorithm is agent state
whatever information is used to pick our next action that's what we call the agent state.

* whatever information the agent uses to pick the next action.
* It is the information used by reinforcement learning algorithms
* It can be any function of history.

$$S_t^a = f(H_t)$$

An information state (Markov state) contains all usefull information from the history.

↳ Definition: A state S_t is Markov if and only if

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

The probability of the next state conditioned on the state you're in is the same as the probability of the next state if you showed all of the previous states to this system.

s.a.m

In other way you can throw away all of the previous state and just retain your current state and you would get the same characterization of the future.

"The future is independent of the past given the present"

$$\underbrace{H_{1:t} \longrightarrow S_t \longrightarrow H_{t+1:\infty}}_{\text{only need to store this } S_t}$$

- ★ Once the state is known, the history may be thrown away.
- ★ The state is a sufficient statistics of the future.
  - ↳ The state S_t fully characterize the distribution over future actions, observations and rewards.
- ★ The environment state S_t^e is Markov
- ★ The history H_t is Markov.

Full observability: agent directly observes environment state.

$$O_t = S_t^e = S_t^a$$

→ Agent state = environment state = information state.

Formally this is a Markov Decision Process (MDP)

→ **Partially observability** / **Partial**: agent indirectly observe the environment.

ex:
- a robot with camera vision isn't told its absolute location.
- a trading agent only observe current prices.
- a poker playing agent only observes public cards

Now agent state ≠ environment state

مثلا پوکر پلیر نمی داند بقیه ورق چیه

دقت

Formally this is Partially observable Markov decision process (POMDP)

s.a.m

Agent must construct its own state representation S_t^a, eg. Complete history $S_t^a = H_t$

54:11

Date

Beliefs of environment state $S_t^a = (P[S_t^e = s^1], \ldots, P[S_t^e = s^n])$

↳ I don't know what's happening in the environment but I'm going to keep a probability distribution over where I think I am in the environment.

defines the the state that we're going to use to actually decide what to do.

Another Choice: Recurrent Neural Network:

ترکیب خطی

$$S_t^a = \sigma(\underbrace{S_{t-1}^a}_{\text{قبلی state}} W_s + \underbrace{O_t}_{\text{فعلی observation}} W_o)$$

Inside An RL Agent:

★ An RL agent may include one or more of these Components

- Policy: agent's behaviour function → what action to take
- Value Function: how good is each state and/or action
- Model: agent's representation of the environment

how much reward do we expect to get if we take the action in this particular state.

★ A policy is the agent's behaviour.

★ It is a map from state to action, e.g.

★ Deterministic policy: $a = \pi(s)$

★ Stochastic policy: $\pi(a|s) = P[A = a | S = s]$

Subject:
Date

* Value function is a prediction of future reward.
* used to evaluate the goodness / badness of states.
  And therefore to select between actions,

$$V_\pi(s) = E[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots \mid S_t = s]$$

  how much total reward we can get from some state onwards if we're going to follow this particular behaviour.

* A model predicts what the environment will do next
  Transitions: P predicts the next state (i.e. dynamics)
  Rewards: R predicts the next (immediate) reward. e.g.

$$P_{ss'}^{a} = P[S' = s' \mid S = s, A = a]$$

$$R_s^a = E[R \mid S = s, A = a]$$

Categorizing RL Agents (1)

- Value Based
  - No Policy (Implicit)
  - Value Function

  The policy is kind of implicit it just has to read out look at the value function and pick the best action.

- Policy Based
  - Policy
  - No Value Function

  instead of representing inside this value function, the agent, instead we explicitly represent the policy, so we work with the policy.

Date

- Actor Critic
  - Policy
  - Value Function

} Combine them both together

Categorizing RL Agents (2)

- Model Free
  - Policy and/or Value Function
  - No Model

- Model Based
  - Policy and/or Value Function
  - Model

Problem within RL:

Two fundamental Problems in sequential decision making

- Reinforcement Learning:
  1) The environment is initially unknown
  2) The agent interacts with the environment
  3) The agent improves its Policy
- Planning:
  1) A model of the environment is known.
  2) The agent performs computations with its model (without any external interaction) — Perfect model
  3) The agent improves its Policy

Subject:
Date

Atari Problem: RL
* Rules of the game are unkown.
* Learn directly from interactive game-play.
* Pick actions on Joystick, see pixels and scores.

Atari problem: 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?
  * " " " score be?
* Plan ahead to find optimal policy
  * eg tree search

Exploration and Exploitation (1) (2)

- Reinforcement Learning is like a trial and error learning
- the agent should discover a good policy
- from its experience of the environment
- without losing too much reward along the way

Exploration finds more information about the environment
Exploitation exploits known information to maximise reward
} trade off

- Prediction: evaluate the future
  → Given a policy
- Control: optimise the future
  → Find the best policy