Introduction to Reinforcement Learning

Lec-1

→ Concerned with making a sequence of good/optimal decisions

Machine Learning

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Characteristics of RL

→ There is no supervisor, only a reward signal whether the action was good/bad or +10, -5 etc.

→ Feedback/reward is delayed, not instantaneous

→ Time really matters (sequential data, not iid)

→ In RL, agent takes actions that affect the subsequent data it receives

Examples of RL

1. Fly stunt manoeuvres in a helicopter
2. Defeat the world champion at Backgammon
3. Manage an investment portfolio
4. Make a humanoid robot walk

Rewards

* A reward R_t is a scalar feedback signal
* Indicates how well agent is doing at time t
* Job of agent is to sum these R_t & get as much total reward as possible

* RL is based on Reward hypothesis

All goals can be described by the maximization of expected cumulative reward.

Examples

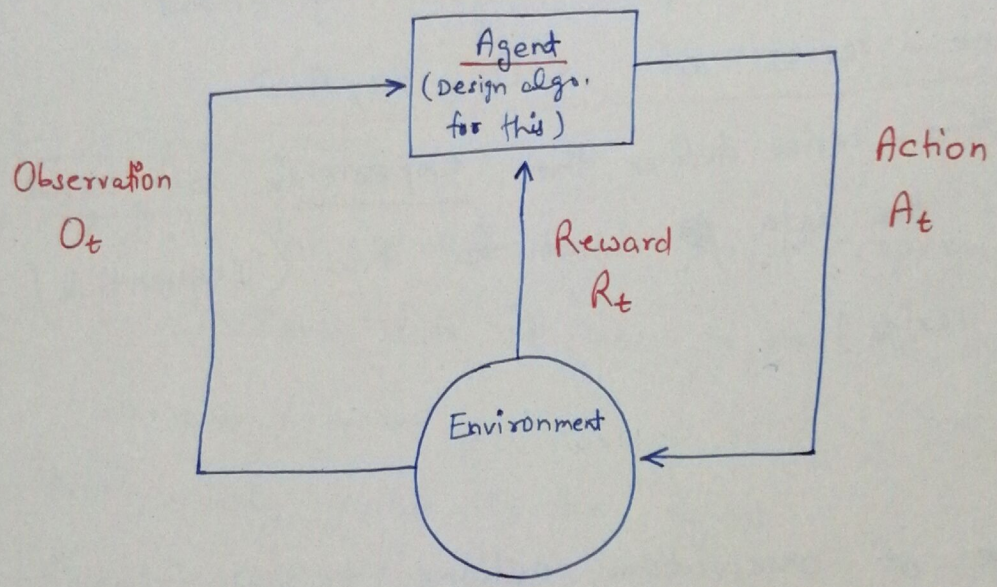
1. Fly stunt manoeuvres in a helicopter
   * +ve reward for following desired trajectory
   * -ve reward for crashing
2. Defeat the WC at backgammon
   * +ve/-ve reward for winning/losing the game
3. Manage an investment portfolio
   * +ve reward for each $ in the bank
4. Robot walk
   * +ve reward for forward motion
   * -ve reward for falling over

Sequential decision making

Goal: Select actions to maximize 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

Agent & Environment



- → Environment is the real world out there
- → Agent interacts with environment
- → We/agent have no control over the environment directly
- → Agent influences the environment by actions only

Agent: at each step t

1. Executes action A_t
2. Receives observation O_t
3. Receives reward (scalar) R_t

Environment: at each time step t

1. Receives action A_t
2. Emits observation O_t
3. Emits scalar reward R_t

* Trial & error loop of agent generates a time-series of $<O_t, R_t, A_t>$
* This time-series defines the experience of agent
* This is the data used for RL (sequential / time-series)

History

* 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.

→ It is all the information that the agent is exposed to

→ What happens next depends on the history

** The agent selects actions

* The environment selects observations / rewards

But, history is often too large / a lot of information to deal with, eg. think of video game.

State

→ Can think it as a summary of history
→ State is the information used to determine what happens next
→ State is a fun^n of history:

$$S_t = f(H_t)$$

Environment state

→ S_t^e is the environment's private representation
→ i.e. whatever data it uses to pick the next observation / reward
→ Not usually visible to the agent
→ Even if S_t^e is visible, it may contain irrelevant info.

Agent State

→ S_t^a is the agent's internal representation
→ Info. used by agent to pick next action
→ Info used by RL algos
→ Can be any fun^n of history

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

→ Agent can choose fun^n f to convert history into state

Information state (Markov State)

→ Contains all useful info. from the history

☞ A state is Markov iff

$$P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_1 \dots S_t]$$

"future is independent of past given the present"

→ If the state representation is Markov, then you can throw away the history

$$H_{1:t} \longrightarrow S_t \longrightarrow H_{t+1:\infty}$$

→ State is a sufficient statistic of the future

* The environment state S_t^e is Markov
* The history is Markov H_t

→ Agent's state influences the action of agent

☞ Different states might / probably will result in different actions

Fully Observable Environments

→ Agent directly observes environment states

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

→ Agent state = Env. state = Info. state

→ formally, this is a Markov Decision Process (MDP)

Partially Observable Environments

→ Agent indirectly observes environment

Eg. Poker playing agent only observes public cards

→ Agent state ≠ Env. state

→ This is partially observable Markov Decision Process (POMDP)

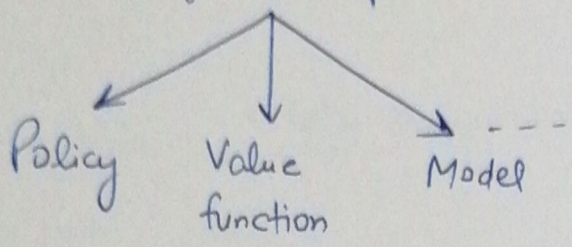
→ Agent must construct its own state representation S_t^a. Eg.

(1) Comple History: $S_t^a = H_t$

(2) $S_t^a = \left(P(S_t^e = s^1), \ldots, P(S_t^e = s^n)\right)$

(3) RNN: $S_t^a = \sigma\left(S_{t-1}^a W_s + O_t W_o\right)$

RL Agent components



→ Policy is the agent's behaviour

→ Map from state to action

Deterministic policy: $a = \pi(s)$

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

Value fun^n

→ Prediction of expected future reward

→ used to choose b/w different actions

→ Given policy π, its value fun^n

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

Model

→ A model predicts what the environment will do next

Transition model: $\mathcal{P}$ predicts the next state (i.e. dynamics)

Reward model: R predicts the next (immediate) reward

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

$$R^a_s = \mathbb{E}[R \mid S = s, A = a]$$

An RL agent typically doesn't need all these components to work.

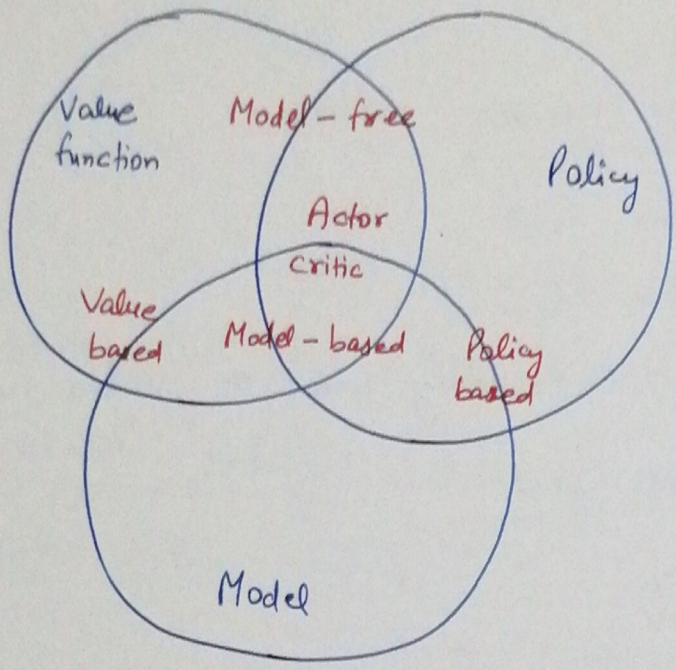
1. (a) **Value-based agents**
   - Has value fun^n
   - No policy required
   - Take actions greedily based on value fun^n

   (b) **Policy-based agents**
   - Has policy
   - No value fun^n

   (c) **Actor Critic**
   - Policy
   - Value fun^n

2. (a) **Model free agents** (*)
   - Don't try to explicitly build model
   - Use policy and/or value fun^n

   (b) **Model based RL**
   - Model
   - Policy and/or value fun^n



Planning

- A model of env. is known
- The agent performs computations with its model (w/o external interactions)
- Agent improves its policy

Exploration and Exploitation

Exploration finds out more info about the environment

Exploitation Exploits known information to maximize rewards

→ RL is like trail & error learning

→ Discover good policy from experiences of the env. w/o loosing too much reward along the way