Lecture notes - Introduction to Reinforcement learning with David Silver

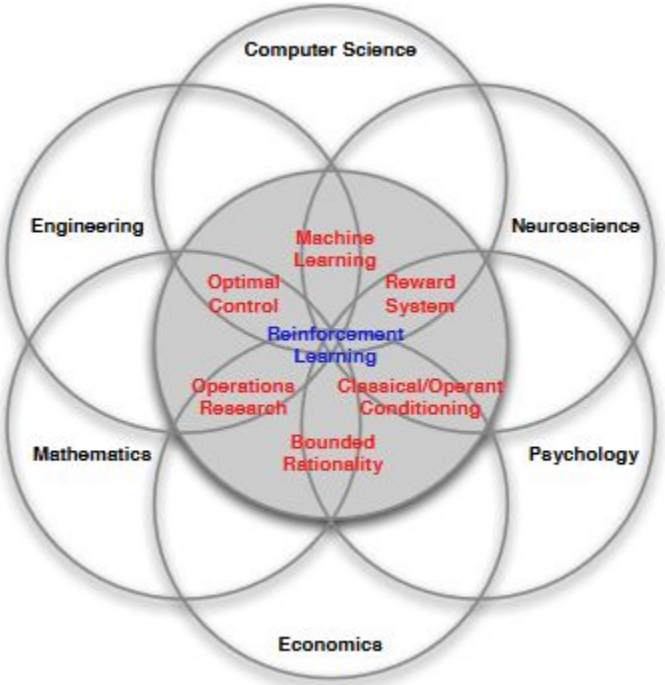
Lecture 1

By - Amal Sunny

Reinforcement Learning

About the subject

- Intersects with a large number of fields (venn diagram)
- Science of decisions making, trying to understand the optimal way to make decisions
- Neuroscience - one of the more recent breakthroughs is the rewards system in our brain, generating dopamine. Trying to maximime and understand its generation.
- Conditioning - is the same for animals. What causes certain reactions from animals.
- Economics - fields of game theory, optimality theory, etc. Again these deal with the optimality and reasons behind decisions taken by people



Difference from other ML methods

- No supervisor, no best option - only a reward signal
- No instant feedback (like labels incase of supervised), rewards (may) come out steps later as decisions unfold later
- Time, or rather sequence matters for RL a lot. Its a dynamic system, each decision and its impact varies a lot depending on when the event happens
- And following that, agent actions affect subsequent data that it may recieve. Agent makes action, ends up in different state and so actions present may be different.

The RL Problem Terminology

Rewards

- Reward is a scalar feedback signal - i.e just a number.
  - Denoted by R_t
  - Further down the line, the scalirity of the reward can be questioned. What if there are conflicting goals/decisions which are both favourable ?
    - In RL context (as well as in real life), we would have to weight the two goals and choose which has higher priority. That becomes the goal we strive to achieve.
- Reward indicates how well the agent is doing
- Agent's goal is to maximise cumulative reward.
- Thus we define all goal to be the maximisation of expected cumulative rewards, to ensure rewards line up with our expected outcome
  - Also known as **Reward Hypothesis**
- If no intermediate rewards
  - > Episodes are set with rewards, and cumulative episode rewards are maximised.
- If reward is time based
  - > Each second passing, is set to have a negative reward.
- Our goal is to set a unifying framework for all these problems.

Sequential Decision Making

- That goal would be: **select actions to maximsie total future rewards**
- Actions may have long term consequences
- Rewards may be delayed
- So sometimes immediate reward can be sacrifices for the long term, and vice versa in other cases.

Agent and Environment

- Observations O_t - every step it has some observation of the environment
- Action A_t - it executes some action
- Reward R_t - it receives a reward in that state informing it about the decisions it made.
- Environment - generates observation and reward. Independent of us, except of the action we take in the environment.
  - So it receives action A_t,
  - Generates observation and reward, O_t and R_t

History and State

- History is the sequence of observations, actions, rewards

$$H_t = A_1, O_1, R_1, ..., A_t, O_t, R_t$$

- History only holds the stream of data until given point of time t (Hence being denoted as H_t).
- Only until t as agent has only experienced until that point, and should only have information till then.
- The agent (our algorithm) would be a creating a mapping between history and the next action.
- The environment would be mapping the next observation based on the history.
- However, this history would be long and convoluted - would make processes too slow.

State

- Hence, State is introduced to solve that problem.
- State is a concise summary of the information needed by agent to determine what happens next.
- Formally, State is any function of history

$$S_t = f(H_t)$$

- Environment state, S_t^e is the environment's internal representation. Essentially all the information required to be able to generate next observation/reward
  - This state is usually invisible to the agent
  - Even if visible, it may contain irrelevant information
- Agent state S_t^a is the agent's internal representation. Essentially all the information required to pick the next action
  - Info used by RL algorithms

Information State (Markov State)

- An information state contains all useful information from the history.
- Defining condition
  - > A state S_t is Markov if and only if
    >
    > $$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, ..., S_t]$$
  - i.e Probability of the next state, given the current state would be the same as the probability of the next state, given all past states.
- Essentially the information of the current state is **ALL** that's needed for the next state.
- > "The future is independent of the past given the present"
- So only the state is needed, the history can be discarded
- Environment state S_t^e is Markov

Fully Observable Environment

- Full observability: agent directly observes environment state

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

- Agent state = env state = information state
- Environments like these are called **Markov decision process (MDP)**

Partially Observable Enviornment

- Agent indirectly observes environment (only partial information)
- Agent state $\neq$ Environment state
- Formally, this is a **partially observable Markov decision process (POMDP)**
- Agent constructs its own state representation S_t^a
- For eg:

- Naive approach - Complete history is state: $S_t^a = H_t$
- Probabilistic view: We build beliefs assuming where the agent is based on probabilites -

$$S_t^a = (P[S_t^e = s^1], ..., P[S_t^e = s^n])$$

- Recurrent neural network: Agent's state at last step is taken with latest observation with some linear transformation

$$S_a^t = \sigma(S_a^{t-1} W_s + O_t W_o)$$

Inside the RL agent

- Components of RL agent
  - Policy: agent's behavior function
  - Value function: reward given in a particular action/state
  - Model: agent's view of the environment(how it works)
- Not always required to have all of them, but key components

Policy

- Agent's behaviour (how it takes actions)
- Maps from state to action
- Denoted by π
- Policies can be
  - Deterministic : $a = \pi(s)$
  - Stochastic: $\pi(a|s) = P[A = a|S = s]$
    - More useful for exploration

Value function

- Prediction of future reward
- Denoted by v
- Used to evaluate the goodness of states
- Thus used to select between actions, eg:

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | S_t = s]$$

- γ is the discount factor, and limits how far ahead we're looking for potential reward
- $v_\pi(s)$ is the value function following policy π

Model

- Model predicts how the environment would react (to try and assume what the best action would be)
- Transitions: P predicts the next state (dynamics of the env)
- Rewards: R predicts the next (immediate) reward
  Eg:

$$P^a_{ss\prime} = P[S_{t+1} = s\prime | S_t = s, A_t = a]$$

$$R^a_s = E[R_{t+1} | S_t = s, A_t = a]$$

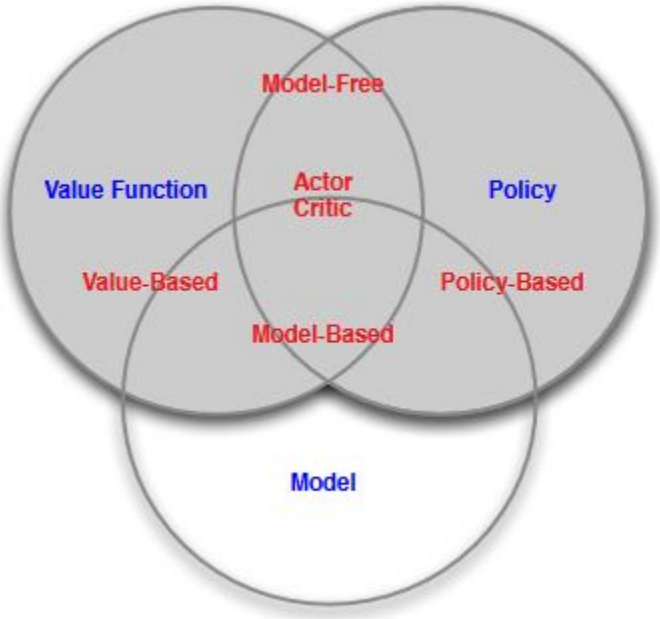
Categorizing RL Agents

---

1. Value Based
   - A Value function exists
   - No policy exists (implicit - greedy)
   - There exists a set of values mapped to each state, a path following maximum reward is taken
2. Policy Based
   - Policy exists
   - No value function
   - A structure is maintained to store actions to be taken at each step.
3. Actor Critic
   - Policy
   - Value function
   - Essentially a combination of both.

Another set of categories for RL agents

1. Model free
    - Policy and/or value function
    - No model
    - We are not concerned with learning how the world works in these agents, just generating policy and/or value functions to function
2. Model Based
    - Policy and/or value function
    - Model is created.



Problems in RL

Two fundamental problems exist:

1. Reinforcement learning

- Environment is unknown
- Agent has to interact with environment to learn it
- Agent improves policy as a result

2. Planning

- Has the environment model already known (for eg: dynamics of the system)
- The agent performs internal computations on its model (doesn't have to actually take an action on actual environment there)
- Agent improves policy

Further problems under RL

Exploration and exploitation

- Reinforcement learning is like trial-and-error.
- Exploration - knowledge of environment at expense of reward.
- Exploitation - Go for best known reward at the expense of more knowledge of environment
- Essentially, agent has to balance exploration to ensure a good policy is formulated with exploitation so it doesn't lose too much reward exploring.

Prediction and Control

- Prediction: Evaluates the future
    - A policy is given, and we evaluate it and calculate how much reward we'd obtain.
- Control: Optimizes the future, by deciding best policy
    - Finding best policy to obtain max reward.
- Usually, we have to solve the prediction problem to solve the control problem. Need to evaluate all policies to find best one.

Weekly Problems

1. Problem 1: MDP Design (taken from CS234 RL course at Standford University)

You are in a Las Vegas casino! You have $20 for this casino venture and will play until you lose it all or as soon as you double your money (i.e., increase your holding to at least $40). You can choose to play two slot machines: 1) slot machine A costs $10 to play and will return $20 with probability 0.05 and $0 otherwise; and 2) slot machine B costs $20 to play and will return $30 with probability 0.01 and $0 otherwise. Until you are done, you will choose to play machine A or machine B in each turn. In the space below, provide an MDP that captures the above description.
Describe the state space, action space, rewards and transition probabilities. Assume the discount factor γ = 1. Rewards should yield a higher reward when terminating with $40 than when terminating with $0. Also, the reward for terminating with $40 should be the same regardless of how we got there (and equivalently for $0)

Answer

An MDP consists of State, Action, Reward, Transition Probability(probability of transitioning to the next state for all possible states there).

- Technically we have γ as the 5th componenent but thats given as 1 here.

States: $S = \{0, 10, 20, 30, 40\}$ representing each state as the amount of money in hand, with 0 and 40 being the terminal ones where the game ends.

Action space: $A = \{A, B\}$.A & B are representing which slot machine is played.

Rewards: $R(s, a, s') = \{1 \text{ if } \{s' = 40\} \text{ else, } 0\}$

- The checking condition is to give reward only if overall goal is achieved
- This reward also ensures regardless of how we get to $40, reward is the same.

Transition Probabilities:
$P(s + 10|s, A) = 0.05$
$P(s - 10|s, A) = 0.95$
$P(s + 10|s, B) = 0.01$
$P(s - 20|s, B) = 0.99$

These probabilities are only valid:

- Under action A when,
  - s = {10,20,30}
- Under action B when,
  - s = {20,30}

All other transition possible have the probability 0 (apart from terminating states where they go into a self loop without any reward).

2. Problem 2: Stochastic Optimal Policies (same as above)

Given an optimal policy that is stochastic in an MDP, show that there is always another deterministic policy that has the same (optimal) value

Answer

A stochastic policy by defination means that under it, any action which has a **non-zero probability** assigned to it, then all those actions have the same expected return in terms of reward(the optimal one in that state). The key point to note here is **all** those actions give the same expected reward. So, if we need to formulate a deterministic one, we can arbitarily pick one and have it be the optimal action in our deterministic policy.