R(ea)L Trader SoC Notes

Shubh Kumar

IIT-B, Summer of Code 2021

1 Markov Decision Process

- MDPs provides us a way to Mathematically formualte the typical RL Problem.
- **The Agent-Environment Relationship**:
  - Actions can be any decision we want the agent to learn and state can be anything which can be useful in choosing actions.
  - Not everything in the environment is unknown to the agent, but the agent can't change anything arbitarily.
  - The Agent could only interact through its actions.
  - The agent-environment relationship represents the limit of the agent control and not it's knowledge.
- **The Markov Property**: Future is independent of the past given the present.

$$P[S_{t+1}|S_t] = P[S|S_1, ..., S_t]$$

$S[i]$ denotes the states, and t is like time(present), so all the equation says is that its Probability of Transition is independent of the past states and even if we have the information pertaining to that, it won't change the Probability of current Transition which would be determined solely by our current state.

State Transition Probability: Probability of moving from one state to the next. It could be abstracted as a matrix with each row containing the probabilities for a particular state, all of which (row-entries) sum to 1.

- **Markov Process/Chains**: Sequence of Random states which fulfill the Markov Property.
- Rewards are the numerical values that the agent receives on performing some action at some state(s) in the environment.
- This total sum of reward the agent receives from the environment is called returns.

$$G_t = \sum_{i=t+1}^{T} r_i$$

t is time. A single action, has its rewards reaped not just at the end of that action, but till the end of a particular number of further actions as well(perhaps a weighted sum would be more accurate! Coming up next ;))

- Episodic Tasks: Those which have an end, Continuous Tasks: Those which don't
- In order to compensate for the summing up to infinity(as we don't want to do that!), We have the Discount Factor(γ). It's a value between 0 and 1. 0 means only consider the current reward for deciding the current action. 1 means keep considering it till infinity. Thus, a value between 0.2 and 0.8 is considered good.
- With that in mind, we change our expression for G_t to:

$$G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+1+i}$$

- Deciding on Discount Factor: It depends on the task that we want to train an agent for. (Hyper-parameterization)
- **Markov Reward Process**:

$$\mathbf{R_s} = E[R_{t+1}|S_t]$$

  Mathematically, we may define MRP as (S,P,R, γ), where:

  - S is set of States
  - P is the Transition-Probability matrix
  - R is the reward function
  - γ is the discount factor
- **Policy Function and Value Function**: Value Function determines how good it is for the agent to be in a particular state. A policy defines what actions to perform in a particular state s.
- A policy (π) is a simple function, that defines a probability distribution over Actions ($a \in A$) for each state ($s \in S$).

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$

- The Value of a state, maybe reached at by:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{t+1+i}|S_t = s \right] \text{, for all } s \in S.$$

  This equation gives us the expected returns starting from state(s) and going to successor states thereafter, with the policy π. One thing to note is the returns we get is stochastic whereas the value of a state is not stochastic.
- Bellman Equation helps us to find optimal policies and value function.
- The value of state(s) is the reward we got upon leaving that state, plus the discounted value of the state we landed upon multiplied by the transition probability that we will move into it.
- The key linear equation is: $v = (I - \gamma P)^{-1} R$, where v is the value-vector, I, the identity matrix, γ, the Discount factor, P, the Transition-Probability Matrix and R the Reward Vector
- Markov Decision Process : It is Markov Reward Process with a decisions.Everything is same like MRP but now we have actual agency that makes decisions or take actions.
- MDP could be Mathematically descirbed by S(set of States), A(set of Actions), P(Probability Transition Matrix), R(Rewards accumulated by actions of agents), γ (Discount Factor).
- State-action value function or Q-Function: This function specifies the how good it is for the agent to take action (a) in a state (s) with a policy π.

2 Bellman Equation and Optimality

- Bellman's Expectation Equation for value function, helps in determining the value of a state: $v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \mid S_t = s\right]$
- Bellman's Expectaion Equation for the State-Action Value function would be: $q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) \mid S_t$
- The above two equations are recursive in the sense, that they provide us with a method to calculate the value function as well as the state-value-action function, provided the values for pre-exisiting states.
- Considering only a shallow bottom-top dependence(of depth 1) between $v_\pi(s)$ and $q_\pi(s,a)$ we get:

$$v_\pi(s) = \textstyle\sum_{a\in\mathcal{A}} \pi(a \mid s) q_\pi(s,a)$$

- Similarly, if we try to write out a relationship the other way round, and taking into account that there will also be a reward for every action which should be taken into account:

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \textstyle\sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_\pi\left(s'\right)$$

- Now, combining both of them for $v_\pi(s)$, we'll get:

$$v_\pi(s) = \textstyle\sum_{a\in\mathcal{A}} \pi(a \mid s)\left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_\pi\left(s'\right)\right)$$

- And for $q_\pi(s,a)$, we'll get:

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \textstyle\sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a'\in\mathcal{A}} \pi\left(a' \mid s'\right) q_\pi\left(s', a'\right)$$

- Now, we've found the formulation to Bellman Expectation Equation. We still need to figure how to optimize it.
- Now we yearn to get the Optimal Value and the Optimal Policy Function.
- Mathematically, the Optimal State-Value Function would look like:

$$v_*(s) = \max_\pi v_\pi(s)$$

- And Similarly for the Optimal State-Action Value Function (Q-Function), We'll get

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

- A crucial expression while trying to get to the value function was $\pi(a|s)$ .i.e. The Conditional Probability for us to go forward with the action a, while in state s. We could prove that a greedy approach is the way to go and thus, we could take this function as:

$$\pi(a|s) = \begin{cases} 1 & \text{if if } a = \underset{a\in\mathcal{A}}{\operatorname{argmax}} q_*(s,a) \\ 0 & \text{if else} \end{cases}$$

- These notions take us Bellman's Optimality Equation which is what we'll be using, it can be expressed as:

$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \textstyle\sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_*\left(s'\right)$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \textstyle\sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*\left(s', a'\right)$$

3 Solving MDP using DP

- If you want, you may learn about DP from any standard Algorithms' Text.
- The optimal Substructure which characterizes DP arises due to the Recursive Nature of Bellman's Optimality equation.
- Solving MDP will have two main parts:
  1. **Prediction:** Predict the optimal Actions using Bellman's Equation. This would essentially mean calculating the Value Function.
  2. **Control:** Optimizing the value function, we calculated during the prediction process. We find optimal value function and optimal policy for our Markov Decision Process.
- To evaluate a particular Policy, we use the Equation:

  $$v_{k+1}(s) = \textstyle\sum_{a\in\mathcal{A}} \pi(a \mid s) \left(\mathcal{R}_s^a + \gamma \textstyle\sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_k\left(s'\right)\right)$$

  where k denotes the iteration we're presently in. As it turns out in the last iteration we'll get to the optimal value Function for each state.
- Next, we calculate our $q_\pi(s,a)$ corresponding to the new value functions, and we change our Policy greedily, as:

  $$\pi'(s) = \underset{a\in\mathcal{A}}{\operatorname{argmax}} q_\pi(s,a)$$
- It is provable, that at each step, we'd be getting to a better version of our policy and eventually get the optimal one.
- This gives us:
  $q_\pi\left(s,\pi'(s)\right) = \max_{a\in\mathcal{A}} q_\pi(s,a) = q_\pi(s,\pi(s)) = v_\pi(s)$ and..

  $v_\pi(s) = \max_{a\in\mathcal{A}} q_\pi(s,a)$
- There are two more variations we should keep in mind:
  1. **Modified Policy Iteration :** It turns out that after a few iterations we already have achieved our optimal policy. We don't need to wait for our value function to converge. We can say if the value function gets updated by a very little amount, say less than an epsilon then we can stop this process. To get the exact conditions when we are to stop, We have the **Principle of Optimality**, which says that a Policy is optimal if we have for it:
     - An Optimal first step, say A
     - And then The Steps from the next step onwards, lead to little change in our Value Function
     - Thus, a policy is said to be optimal if it behaves optimally after we have taken one optimal step, say action A.
  2. **Value Iteration :** Value iteration is the special case for Policy Iteration. When the process of policy evaluation is stopped after one step. We put the values into the Bellman Optimal Equation, until we find the Optimal Value Function. We use only this optimal Value Function in order to calculate the Optimal Policy.
     The Bellman Optimal Equation as its used in Value Iteration could be written as:

     $$v_{k+1}(s) = \max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \textstyle\sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_k\left(s'\right)\right)$$