(Mod 4)

Model Free Methods

→ In order to apply DP algo, we need to have knowledge abt. the ~~env~~ environment i.e. model of the env. $P(s', s | r, a)$ should be known.

However, in most real-world scenarios, the model is not available. We have to implicitly infer the model ~~from~~ from the observations. Methods designed to solve such scenarios are called 'Model free Methods'.

MONTE CARLO METHOD :

i) Type of model free algo.

ii) It involves letting the agent learn from the env. by interacting with it & collecting samples.

iii) MC learns from complete episodes & is ∴ only suitable for episodic tasks.

iv) MC is based on the law of large numbers

v) MC methods use random sampling of state-action sequences to estimate the value function or policy.

vi) Value function / policy function are viewed as the expected value of the total reward over ~~the~~ a prob. distribution of these samples.

$$E[X] = c_1 X_1 + c_2 X_2 + \cdots$$

where $c_1 + c_2 + \cdots = 1$

Goal: given samples under π, estimate q_π / v_π

$E[X] = C_1 X_1 + C_2 X_2 \ldots$ s.t. $C_1 + C_2 + \ldots = 1$

Let's re-arrange the q-value functn:

$Q_\pi(s,a) = \Sigma_{s'} \Sigma_r \, p(s', r \mid s, a) \, [r + \gamma V_\pi(s')]$

$= \Sigma_{s', r} M_{s', r} \times [\text{total rewards}_{s'}]$

$M_{s', r} = \Sigma_{s'} \Sigma_r \, p(s', r \mid s, a)$

total rew $= r + \gamma V_\pi(s')$

WKT, $E[X] = m_1 X_1 + m_2 X_2 + \ldots$

s.t. $\Sigma m_i = 1$

$\therefore Q_\pi(s,a)$ can be written as the exp. value of total rewards over model distributn

$Q_\pi(s,a) = E_{\text{model}}[\text{total rewards}]$

$= E_{\text{model}}[r + \gamma V_\pi(s')]$

Say, you run the episodes an infinite no. of times & calculate the rewards earned every time a (state-action) pair is visited. The avg. of these values will give you an estimate of actual expected state-action value ~~to~~

Prediction & control are two integral steps to solve any RL problem.

- Prediction – evaluating the value function / policy.
  ↳ exploring starts
- Control – improving the policy on the basis of state-value function estimates.
  ↳ ε-greedy

MONTE CARLO PREDICTION:

~~i) MC predi~~

i) Prediction problem refers to estimating the value function or policy of an ~~co~~ agent based on experience gained by interacting with the env.

ii) MC prediction problem is to estimate $q_\pi(s,a)$ i.e. ~~total~~ expected total reward the agent can get from taking action 'a' in state 's'.

iii) For estimating this, you need to run multiple episodes.

iv) Track the total reward you get in every episode corresponding to (s, a) pair.

v) Estimate the action value given by
$$q_\pi(s,a) \approx \sum_{i=1}^{n} \frac{r_i}{n}$$

- Ensure that while following the policy π, each (s, a) pair should be visited enough no. of times to get a true estimate of $q_\pi(s,a)$.
- For this, you ~~we~~ need a condition called 'Exploring ~~start~~'
- It states that every (s, a) pair should have non-zero prob. of being a starting pair.

Exploring starts -

i) Exploring starts is a method that ensures all the ~~sta~~ (s, a) pairs are visited with non-zero prob. in the early stages of learning.

ii) This is achieved by starting each episode from a randomly selected (s, a) pair, rather than following a fixed initial policy.

iii) Assume, there are total 10 states (S₁, ... S₁₀) in the env. & 10 actions (A₁, ... A₁₀) that can be taken in these state.

iv) MC prediction prob. is used to find ~~the~~ an unbiased estimate of $q_\pi(s, a)$ for each (s, a) pair for a deterministic policy π.

v) An episode can be started with any of the 10 states.

vi) Exploring starts allows you to perform any random action in this initial stat.

vii) However, from the next state onwards, it follows π (π is deterministic π(s) → a)

viii) So, the subsequent states will have a fixed action corresponding to ~~the~~ each state.

ix) The new state s' & reward 'r' can be different even if you perform ~~the~~ the same action 'a' in state 's', because the env is stochastic.

policy - deterministic, env - stochastic

x) Episodes are tracked as (state, action, immediate reward) series

MONTE CARLO CONTROL:

i) Control refers to the problem of finding out the optimal policy that maximizes the expected return.

ii) MC control algo starts with an arbitrary policy and estimates the value functn under the policy using MC prediction.

iii) It then updates the policy to be greedy w.r.t. the value function estimate $V(s)$

iv) Policy improvement is done by constructing an ~~imp~~ improved policy π' as the ε-greedy maximisation with ~~of~~ respect to $q_\pi(s,a)$

ε - GREEDY

↳ policy improvement technique

i) ~~ε Epb~~ Epsilon greedy is a popular algo. used in RL for balancing exploration- exploitation tradeoff.

ii) The purpose of ε-greedy maximisation is to ensure that the agent doesnot get stuck in a suboptimal policy due to early convergence or insufficient exploration.

iii) By allowing the agent to try new actions with a small prob, ε-greedy maximisation encourages the agent to try diff. actions and learn about the environment, while still ~~exploring~~ exploiting the actions that have already been learned ~~are~~ to be optimal

~~iv) It chooses a random action~~

iv) It choose the best action with prob. $(1-\varepsilon)$ and a random action with prob. ε

v) Here, ε is a hyperparameter that controls the tradeoff between exploration & exploitation.

vi) Action at time (t) = $\begin{cases} \text{Max } Q_t(a), & \text{prob} = 1-\varepsilon \\ \text{any action } (a), & \text{prob} = \varepsilon \end{cases}$

If ε is too small $\Rightarrow$ actions are biased to be more greedy.

If ε is too large $\Rightarrow$ action explores more.

Off Policy:

Control Problem seeks the best action values for the agent to behave optimally. However, the agent has to behave non-optimally in order to explore ~~more~~ other actions.

A way to handle this dilemma of exploration & exploitation is by using ~~policies~~ off policy

Estimating the expected result of one distribution (policy) using the data / sample from another

Off Policy

Note:
A policy has 2 jobs -
1) generate data ((state, action, reward) trajectories)
2) optimize / improve itself.

i) Off policy learning is a type of ~~lea~~ RL in which the agent learns from the data that was generated by a diff. policy than the one being currently evaluated.

ii) Here, we have 2 policies -

a) Behaviour Policy -
- Policy is used to generate episodes & is more exploratory in nature.
- generates data
- $b(a/s)$

b) Target Policy -
- Policy that is to ~~be~~ be evaluated / improved and that becomes the optimal policy.
- $\pi(a/s)$

iii) In on-policy methods → $b = \pi$
In off policy method → $b \neq \pi$

iv) We make sure that the (s, a) pairs produced by the target policy are also explored by ~~the~~ behaviour policy via importance sampling