RL Lecture-5   Model free control.

On-policy → learning on the job, observing self behaviour

off-policy → learning following someone else's behaviour.

~~Model~~

→ optimize the value functn of an unknown MDP.

Why?

→ most common problem are either:

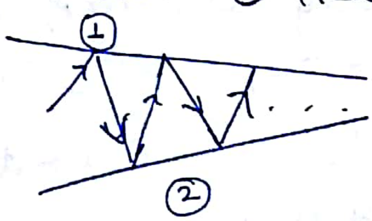
i) model / MDP unknown, but experience can be sampled

ii) MDP known, ~~but~~ it is too expensive to compute even 1 step using the full model, except by sampling

On-policy learning → learning from on/job → while following policy π, learn that policy π.

Off-policy learning → observe someone do something and sampling trajectories from human's behavior(eg.), learn self behaviour (eg. robot).

In DP, alternating betn policy evaluatn & iteration converges to the optimal policy



We only used iterative methods. we'll try to change 1 & 2.

Eg. Monte Carlo control:

1-way → do monte Carlo policy evaluatn & then do greedy policy improvement.

Problems:

1 computing $V(s)$ requires a model, but we want model free

$$V(s) = \max_a \left(R_s^a + \gamma P_{ss'}^a V(s') \right)$$

But in model free, we want to learn a policy without any model. The Q - function defined earlier → means that final score we would get at the end of a game (eg.) if we took action a, when in state s. → $q(s, a)$
So, no model dependence now. Just a state and possible actions that can be taken.

→ greedy over $Q(s, a)$ → action value function

$\pi'(s) = \underset{a}{\operatorname{argmax}}\ Q(s, a)$

For each (s, a) pair taken, mean across all experience, to compute $Q(s, a)$

Proposal changes:

→ MC evaluate $Q = q_{\pi}$

→ Greedy policy improvement?

2nd problem → choosing greedily → we never visit some states or take some actions which means we don't evaluate them correctly & hence, not selecting them.

eg. to illustrate that point. → suppose 2 ~~actions~~ states → S_1 & S_2 upon choosing both once, we found:

$V(S_1) = -1, \quad V(S_2) = +1$ we choose S_2 greedily

now say, $V(S_2) = +3$ → mean = 2, again choose S_2 and so on...

This shows that we haven't even considered state S_1 only after 1 seeing and hence, don't know what value it might have as we keep choosing S_2 infinitely.

ε- greedy exploration ← solution to ensure continual exploration

→ prob. ε → choose random action

→ $1-\varepsilon$ → choose the greedy option.

Thus
$$\pi(a/s) = \begin{cases} \varepsilon/m + (1-\varepsilon) & \rightarrow \text{for greedy action} \\ \varepsilon/m & \rightarrow \text{for any other action} \end{cases}$$

→ keep exploring

ε greedy → policy improvement (ensures) $v_{\pi'}(s) \geq v_\pi(s)$.

π, π' → both ε-greedy policies

$$q_\pi(s, \pi'(s)) = \sum_a \pi'(a/s)\, q_\pi(s,a)$$

$$= (1-\varepsilon) \max_a q_\pi(s,a) + \frac{\varepsilon}{m} \sum_a q(s,a)$$

Now, max is greater than any weighted sum

$$\geq (1-\varepsilon) \sum_a \frac{\pi(s/a) - (\varepsilon/m)}{1-\varepsilon}\, q_\pi(s,a) + \frac{\varepsilon}{m} \sum_a q(s,a)$$

$$= \sum_a \pi(s/a)\, q_\pi(s,a) = v_\pi(s)$$

$\therefore v_{\pi'}(s) \geq v_\pi(s)$. (From earlier similar derivation)

$\therefore$ Finally:

① MC policy evaluation (MCPE) $Q = q_\pi$ ✓ softening of the greedy policy

② ε - greedy policy improvement

Idea → no need to completely ~~eval~~ evaluate the policy, when we can get a better policy in only a few initial steps (Similar to DP lecture).

One extreme → do update every episode, basically on the most fresh ~~da~~ information / estimate of the value function.

after every episode

① MCPE → $Q = q_\pi$

② ε-greedy → For these states which were visited in the episode.

How to ensure we get π^*?

→ Trade off betn exploration & exploitation. as when at π^*, we won't have much exploration.

Idea for balancing the 2 ideas → GLIE.

→ come up with schedule such that 2 conditions are met.

① all state-action pairs explored. infinitely many times

$$\lim_{k \to \infty} N_k(s,a) \to \infty$$

② converges to the optimal greedy policy

$$\lim_{k \to \infty} \pi_k(a|s) = I\left(a^* = \underset{a' \in A}{\operatorname{argmax}}\, Q_k(s,a')\right)$$

One idea → schedule / decay ε for ε - greedy.

→ hyperbolic schedule → $\epsilon_k = 1/k$.

GLIE MC control

→ sample k^{th} episode following π
↓ $\{S_1, A_1, R_1 \dots R_T\}$

→ update mean q value & $N(S^t, A^t)$.

$$N(S_t, A_t) = N(S_t, A_t) + 1$$

$$Q(S_t, A_t) = Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\left(G_T - Q(S_T, A_t)\right)$$

this mean is not the actual statistical mean as we are accumulating values as we improve the policy
↓ changing over time

→ Improve policy

$\epsilon \to 1/k$

$\pi \to \epsilon$- greedy (Q)

GLIE makes sure that these collected statistics converge to the actual ~~mean~~ value over time.

$Q(s,a) \rightarrow q^*(s,a)$

Iterate over the entire process.

→ considerably more efficient ~~ly~~ that updating after a batch

Initialization → In this case, for the 1st time we observe a state $N(s^t, a^t) = 1$, and $Q(s^t, a^t) = G_T$, so init doesn't matter. But for weighing other than $\frac{1}{N(s_t, a_t)}$, it will affect much more.

Now, we'll show TD.

TD vs MC

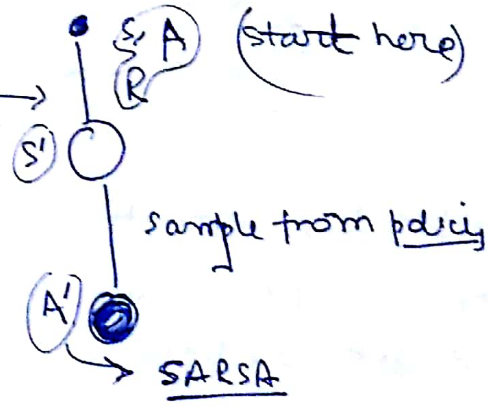
(+) → variance low → online → incomplete sequences.

- look at $Q(s,a)$ for model-free
- slot in TD learning for policy evaluation
- ϵ-greedy

} can update after 1 step., no need to wait till episode end

So, update policy after each step.

Called SARSA.



$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma Q(S', A') - Q(S,A)\right).$$

Sarsa update.

In every time step, update value function.

→ started with S. took action A (from current policy) and reached S'. → update only $\boxed{Q(S,A)}$ → only for this pair

SARSA algo for on-policy

1. Initialize $Q(s, a)\ \forall s, \forall a$. $Q(\text{terminal}, \cdot) = 0$.
2. Repeat for each episode:

   Initialize S

   Choose $A \rightarrow$ sampled using policy derived from Q.

   Repeat (for each step):

   Take action A, got reward R, landed in S'.

   Choose A' from S', using policy derived from Q

   $Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$

   $S \leftarrow S', A' \leftarrow A$.

Sarsa $\rightarrow$ on-policy algorithm.
& converges to the optimal policy, just (1) need a GLIE policy $\rightarrow$ eg. $\epsilon_k = 1/k$

(2) α_t step sizes follows:

$$\sum_{t=1}^{\infty} \alpha_{t'} = \infty \quad \text{(sufficiently large to move } Q \text{ values)}$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty \quad \text{(}\cancel{\text{step sizes}}\text{ changes to } Q \text{ values becomes small) or else noise}$$

In practise, sometimes don't depend on both (1) & (2). (?)
Again apply similar concept of TD(λ) $\rightarrow$ n-step SARS to get the best of both worlds.

$n = 1$. (SARSA)

⋮

$n = \infty$ (MC).

n-step Q return:

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots \gamma^n Q(S_{t+n})$$

Scanned by CamScanner

n-step Q return:

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha (q_t^{(n)} - Q(S_t, A_t))$.

Do this for all n's.

average over 'n'

Sarsa(λ) → $Q(S_t, A_t) = Q(S_t, A_t) + \alpha (q_t^{\lambda} - Q(S_t, A_t))$

$$q_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} (\lambda^{n-1})\, q_t^{(n)}$$

↓

Forward view.

to make algo online

↑

Backward view → using Eligibility Traces

$E_0(s, a) = 0$

$E_t(s, a) = \lambda \gamma E_{t-1}(s, a) + I(S_t = s, A_t = a)$

$Q(s, a)$ → updated for every (s, a) pair

$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$

$Q(s, a) = Q(s, a) + \alpha E_t(s, a) \delta_t$.

- Init $Q(s, a)$ = arbitrarily
- Repeat - (episode)
  - $E(s, a) = 0 \quad \forall s, a$.
  - Init S, A.
  - Repeat - (step):
    - Take action A, observe R, S'
    - Take A' → from. Q.
    - $\delta_t = R_t + \gamma (Q(S', A')) - Q(S, A)$.
    - $E(S, A) = E(S, A) + 1$
    - $\forall a, \forall s$
      - $Q(s, a) = Q(s, a) + \alpha \delta_t E_t(s, a)$
      - $E(s, a) = \lambda \gamma E(s, a)$
    - $A \leftarrow A'$, $S \leftarrow S'$.
  - until termination.

$\boxed{\text{SARSA}(\lambda)}$

↘ faster flow of info. back through time

SARSA(0)

↓

needs many steps to flow the info. back.

Off-policy learning

→ evaluate target policy → to compute v_π / q_π following behaviour policy → $\mu(a/s)$

Why?

→ Learn from obsv. of other agents (eg. human) look at traces of behaviour, not just supervised Learning.

→ Re-use experience generated from old policies $\pi_1, \pi_2, \ldots \pi_{t-1}$ → using off-policy learning can do that.

→ Learn about optimal policy while following exploratory policy

We want policy to be deterministic but also want to explore. optimal off policy learning allows that as we can learn a deterministic optimal policy while following an exploratory policy.

→ Learn about multiple policies while following one policy

Importance Sampling → estimate expectatn of a different distr

$$E_{X\sim P}[f(X)] = \sum_X P(X)\,F(X) = \sum_X Q(X)\,\frac{P(X)}{Q(X)}\,F(X)$$

eg. reward functn

$$= E_{X\sim Q}\left[\frac{P(X)}{Q(X)}\,F(X)\right]$$

Use IS for off policy MC

→ Sample return G_t from μ to evaluate π

→ weigh G_T as per similarity betn policies

→ multiply importance sampling corrections

$$G_t^{\pi/\mu} = \frac{\pi(A_t/s_t)}{\mu(A_t/s_t)}\,\frac{\pi(A_{t+1}/s_{t+1})}{\mu(A_{t+1}/s_{t+1})}\cdots\frac{\pi(A_T/s_T)}{\mu(A_T/s_T)}\,G_t$$

→ update value towards corrected return

$$V(S_t) = V(S_t) + \alpha\left(G_t^{\pi/\mu} - V(S_t)\right)$$

→ Very high variance → as multiplying so many ratios diminishes value.

→ MC → very bad off-policy

Thus, only TD learning for off-policy

Now, IS only after/upto 1 step.

$$V(S_t) = V(S_t) + \alpha\left(\frac{\pi(A_t/S_t)}{\mu(A_t/S_t)}\left(R_{t+1} + \gamma V(S_{t+1})\right) - V(S_t)\right).$$

→ policies need to be similar only over a single step.

→ much lower variance than MC (and can still blow off)

Q-learning → Best with off-policy

→ make use of Q values.

→ no IS reqd. what actually took

→ next action from π $A_t \sim \mu(\cdot|S_t)$

→ also an alternative successor action π $A' \sim \pi(\cdot|S_t)$
(what we could have taken following target-policy in future.)

→ update $Q(S_t, A_t)$ towards value of successive action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t)\right)$$

main

→ allow both behaviour & target-policies to improve

→ learn greedy policy from exploratory policy.

π → greedy wrt. $Q(s,a)$

$$\pi(S_{t+1}) = \underset{a'}{\operatorname{argmax}}\, Q(S_{t+1}, a').$$