Lecture 5

→ on policy learning → "learn on the job"
You learn ~~and~~ about the policy which you are following. In simple words you are doing the task in one specific way and you are learning about your method while doing it.
You sample from a policy π and at the same time you are evaluating that same policy.

→ ~~Of~~ Off Policy → "Look over someone's shoulder".
There are two policies, the one way you are using to take actions 'u' and the other policy that you want to evaluate say 'π', Learning about 'π' while drawing samples from 'u' is off policy learning

---

→ Monte Carlo control → we can't use the previous monte carlo control method of first evaluating the V(s) and then act greedly. Because to act greedly, we must know the $Q^a(s)$ for the state. To be able to estimate the Q value for an action in a state, we need the state transition probability upon taking the action

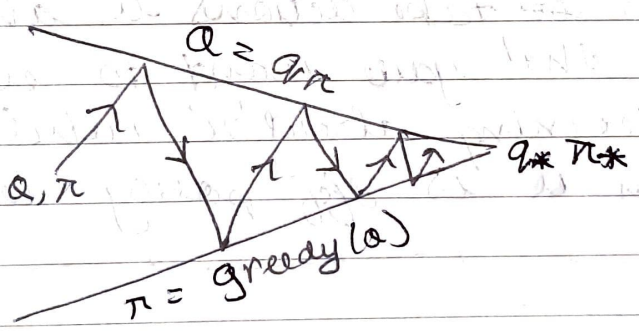
$$Q(s)^a = R_s^a + \sum_{s' \in S} P_{ss'}^a V(s')$$

As ~~the~~ ~~we~~ we are studying model free control, we don't have P values and thus this method will break

Instead we follow greedy policy improvement over $Q(s,a)$.

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

So in policy evaluation, instead of ~~generating~~ generating V^π values for all states we generate $Q^\pi(s,a)$ values for all state action pairs and directly follow greedy policy over all $Q(s,a)$ values for a state.



Policy evaluation - Monte carlo policy evaluation, $Q = q_\pi$

Policy improvement → ~~to~~ We can't take greedy policy. Because if we keep taking greedy right from the beginning, ~~then~~ there may be some state action pairs which may be unexplored.

Instead we use ε-greedy Exploration, with probability ε we choose randomly and with $1-\varepsilon$ we choose greedily.

$$\pi(a|s) = \begin{cases} \varepsilon/m + 1 - \varepsilon & \text{if } a^* = \underset{a \in A}{\operatorname{argmax}}\, Q(s,a) \\ \varepsilon/m & \text{otherwise} \end{cases}$$

→ Another method for Exploration - Exploitation

GLIE → Greedy in the limit with infinite exploration

two things must satisfy

- All state action pairs are explored infinity many time -

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

It means that there is no state - action pair left which is not sufficiently explored That is we must know the Q^* for every state action pair which is the Expected return generated by taking that particular action in that state This is possible by taking ∞ samples of the returns generated by $Q^{\pi}(s,s)$ and doing an empirical mean.

↳ The policy eventually becomes greedy

$$\lim_{k \to \infty} \pi_k(a|s) = 1 \left(a = \arg\max_{a \in A} Q_k(s, a')\right)$$

We want to ~~eq~~ eventually act greedily. We eventually always want to take the best action with max Q value and not act ~~sto~~ stocastically.

→ GLIE Monte Carlo Control

Sample k^{th} episode using π:

$\{S_1, A_1, R_1, \ldots, S_T\}$

For each state and action:
We ~~upto~~ update the $Q(s,a)$ using incremental average
We calculate the number of times we ~~visite~~ visited the state and ~~tod~~ took that action

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

and update $Q(S_t, A_t)$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} (G_t - Q(S_t, A_t))$$

Then ~~impo~~ policy improvement

$\epsilon \leftarrow 1/k$

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

as k increase ϵ decreases, we eventually get a greedy policy in the end as $\epsilon \to 0$.

~~MC~~

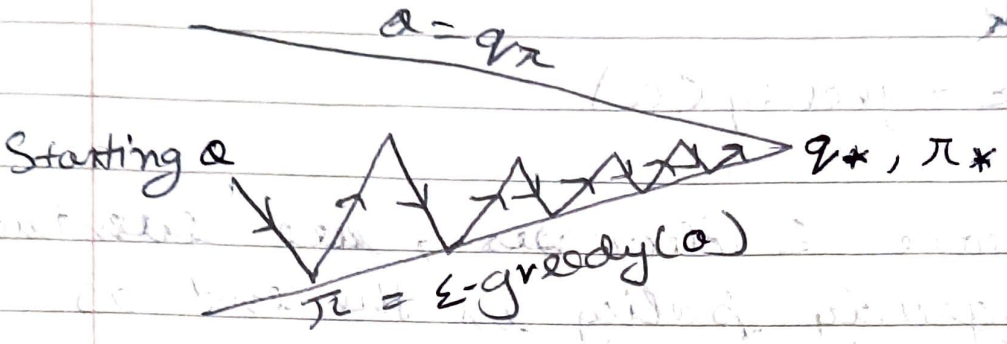
Updating action value function with Sarsa

we start with some state and pick an action ~~according to the policy~~. whose $Q(s,a)$ we need to evaluate. We get are reward and end up in a state. We then sample π and pick an action $\in A$ and uses it $Q(s',A)$ to get the $Q(s,a)$

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

SARSA update

→ on policy control with SARSA.



we update our policy π ~~with~~ after every time step.

We sample ~~a~~ from our policy and takes an action, updates the ~~policy~~ Q(s,a), then we ~~upd~~ update our policy π w.r.t ~~to~~ the new Q we have and then take another action w.r.t the new policy π, correct our policy again and this continues until the end of episode (if it is terminating)

Pseudo code →

→ Initialize Q(s,a) ∀ s ∈ S, a ∈ A(s), arbitrarily and Q(terminal state, •) = 0

→ Repeat (for each episode):
  → Initialize S
  → Choose A from S using policy derived from Q
  → Repeat (for each step of episode):
    → take action A, observe R, S'
    → Choose A' from S' using policy derived from Q
    → $Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$
    → $S \leftarrow S'$; $A \leftarrow A'$
  → until S is terminal

→ Convergence of Sarsa.

Sarsa converges to the optimal action value function, under the following conditions

↳ GLIE sequence of policy $\pi_t(a|s)$

↳ Robbin-Monro sequence of step size α_t

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

→ n-Step Sarsa →

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

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

⋮

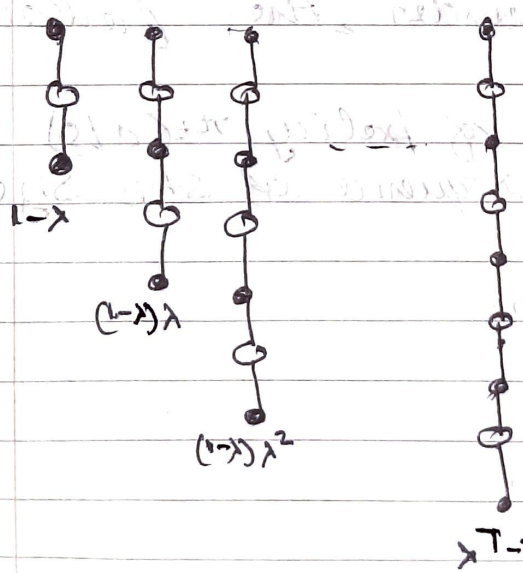
$n=\infty \quad q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$

↓ (MC)

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

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[q_t^{(n)} - Q(S_t, A_t)\right]$$

→ Forward view Sarsa (λ)



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

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(q_t^{\lambda} - Q(S_t, A_t)\right)$$

→ Backward view SARSA (λ)

$E_0(s,a) = 0$

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

This is done for all action-value pair

Similar to model free prediction.

$Q(s,a)$ is updated for every state s and action a.

We update in proportion to the eligibility trace

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

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

Pseudo-code →

→ Initialize $Q(s,a)$ arbitrarily for all $s \in S$ and $a \in A(s)$

→ Repeat (for each ~~epsode~~ episode):

→ $E(s,a) = 0 \quad \forall s \in S, a \in A$

→ Initialize S, A

→ Repeat for each step in episode

→ Take action A; observe R, S'

→ Choose A' from S' using policy

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

→ $E(S,A) \leftarrow E(S,A) + 1$

→ for all $s \in S$, $a \in A(s)$:

→ $Q(s,a) \leftarrow Q(s,a) + \alpha \delta E(s,a)$

→ $E(s,a) \leftarrow \gamma \lambda E(s,A)$

→ $S \leftarrow S'$, $A \leftarrow A'$

→ until S is terminal

→ Off policy learning →

Evaluate some other policy while following one policy.

That is draw samples and rewards from one policy and use that to evaluate another policy.

→ Evaluate target policy $\pi(a|s)$ to compute $V_\pi(s)$ or $q_\pi(s,a)$ while following behaviour policy ~~$\mu(a|s)$~~ $\mu(a|s)$

→ Importance sampling →

$$E_{x\sim P}[f(x)] = \sum P(x) f(x)$$

$$= \sum Q(x) \frac{P(x)}{Q(x)} \cdot f(x)$$

$$= E_{x\sim Q}\left[\frac{P(x)}{Q(x)} f(x)\right]$$

→ Monte carlo learning doesn't work for off policy learning ~~... Explain at last~~

~~Off policy TD learning.~~

Given a starting state S_t, the probability of the subsequent state-action trajectory; $A_t, S_{t+1}, A_{t+1} \ldots S_T$ occurring under policy π is

$$\prod_{k=t}^{T-1} \pi(A_k|S_k)\, p(S_{k+1}|S_k, A_k)$$

p is the state transition probability and π is the policy.

Thus the relative probability of the trajectory under the target and behaviour policy

$$\rho_t^T = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)\, p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} u(A_k|S_k)\, p(S_{k+1}|S_k, A_k)}$$

$$\Rightarrow \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)}{\prod_{k=1}^{T-1} u(A_k|S_k)}$$ which only depends upon u and π and not on rest of the MDP.

Let t denote timestep which doesn't end or terminate after the end of the episode but rather continues to increase in the next episode.

Let $\mathcal{T}(s)$ denote ~~all the~~ the set of all the time step t in which state s is visited. This can be first visit or every visit.

Let $T(t)$ denote the first time of termination following time t and G_t denote

The return after t up through T(t)
Then $\{G_t\}_{t \in \mathcal{T}(s)}$ are the returns that pertain to state s, and $\{\rho_t^{T(t)}\}_{t \in \mathcal{T}(s)}$ are corresponding importance sampling ratios. To estimate $v_\pi(s)$, we simply scale the returns by ratios and average the returns

Averaging can be
1) simple average by dividing by the number of time the state is visited.

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_t^{T(t)} G_t}{|\mathcal{T}(s)|}$$

2) weighted average

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \rho_t^{T(t)} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_t^{T(t)}}$$

or zero if denominator is 0.

1) has high variance and 2) has high bias.

→ Incremental updates → for ordinary mean, the incremental update would follow the

$$V(s) \leftarrow V(s) + \frac{1}{N(s)} (G_t - V(s))$$

However for weighted average things would

differ.

Suppose we have a ~~the~~ series of returns $G_1, G_2, \ldots, G_{n-1}$, all starting in the same state and each with a corresponding random weight. We we need

$$V(s) = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k} \quad ; n \geq 2$$

can keep it uptodate as we generate more ~~G's~~, G_n.

to For this we must metain the cumulative sum C_n of the weights given to the first n returns. Therefore the update rule becomes

$$V_{n+1} = V_n + \frac{W_n}{C_n}\left[G_n - V_n\right] \quad ; n \geq 1$$

and $C_n \leftarrow C_n + W_{n+1}$

→ Off Policy T-D learning

It is in simple terms defined as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

In this case the learned action value function, Q, directly approximates q_*, the optimal action value function independent of the policy being followed.

However if we want to learn a policy π (target policy) while following the behaviour policy μ we take one step according to our behaviour policy μ and record the reward. Then we update our $V(S_t)$ as

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

$V(S_t)$ is the expected value of all the returns that we can generate. The expectation depends upon the probability of getting a certain G_t. Higher the P_t, higher would be the weightage of G_t in the $E(G_t)$

This probability is determined by two distributions, the policy and the transition probability.

So to know what would be the probability of getting some return G_t when followed a target policy while generating G_t samples from behaviour policy, we use importance sampling

We take the G_t obtained for that particular action and acknowledging the fact that the contribution of that G_t^{π} would not be that same as G_t^{μ} or in other words, If we sample the episodes from that state many times we will get ~~that pert~~ to the one particular ~~S~~ state a different number of times while following the behaviour policy than following target policy; ~~Hence~~ hence to get the $V^{\pi}(s_t)$ we multiply the ~~target~~ $R_{t+1} + \gamma V(S_{t+1})$ by the ratio of

Probability of it occurring in π
Probability of it occurring in μ

$$= \frac{\pi(A_t/S_t) \cdot P(S_{t+1}/S_t, A_t)}{\mu(A_t/S_t) \cdot P(S_{t+1}/S_t, A_t)}$$

$$\Rightarrow \frac{\pi(A_t/S_t)}{\mu(A_t/S_t)}$$

this would dilute down or concentrate the weightage of one particular G_t according to its relative probability of occuring while following the target policy π

∴ the T-D equation becomes (SARSA)

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

This has much lower variance than mati carlo off policy learning.

→ Q-Learning →

→ ~~Q-learning~~

→ Q-Learning →

We follow a normal SARSA to update ~~policy~~.

We take first action using our behaviour π and get a reward while calculating the $q(S_{t+1}, A)$ we use our target policy.

Let $Q(S_t, A')$ denote q value at state S and following target policy π and $q(S_t, A)$ be the action value for state S and following target π.

⇒

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

Think of it, in simple terms, as using π to just to generate episodes of data and ultimately updating q π using those episodes.

→ Off policy control with Q learning

In this case the target policy is greedy policy. w.r.t $Q(s,a)$

~~$T(S_{t+1}) = \arg\max_{a'} Q(S_{t+1}, a')$~~

$$\pi(S_{t+1}) = \arg\max_{a'} Q(S_{t+1}, a')$$

the behavior and the target policy can improve.

The behaviour policy will be ϵ greedy wrt $Q(s,a)$ so then we have sufficient exploration.

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

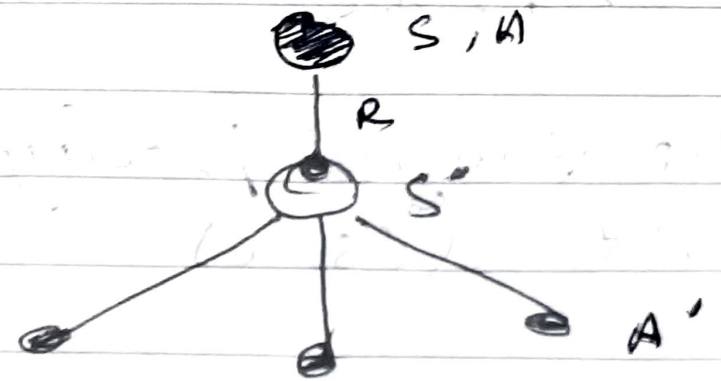
$R_{t+1} + \gamma Q(S_{t+1}, A) \rightarrow$ target

$$\Rightarrow R_{t+1} + \gamma Q\left(S_{t+1}, \arg\max_{a''} Q(S_{t+1}, a')\right)$$

~~$\Rightarrow R_{t+1} + \gamma Q$~~

$$\Rightarrow R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a')$$

Basically go ϵ-greedy while sampling and completely greedy while bootstrapping or calculating the target.



$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_{a'} Q'(S',a') - Q(S,a)\right)$$

It converges to optimal action value function.