RL Lecture 4 – Model free Learning

What we did earlier using DP, required the agent to know the entire model dynamics i.e. the state transition probabilities, reward functions, etc. But since in most real life problems, the agent wouldn't know what the entire dynamics of the environment are. What can we do? That is discussed now! ↳ Model-free Learning

Just like earlier, we break it into 2 components:

1. Policy evaluation → model free prediction
2. Finding the optimal policy → model free control (Lecture 5).

estimate the value function of an unknown MDP.

Model free prediction

→ Monte Carlo Learning:

- Learns directly from episodes of experience
- Look only at complete episodes
- Idea → value = mean of the sample returns across many episodes.

Problem: Applicable to only episodic MDPs, i.e. those where episodes ~~which~~ terminate.

Scanned by CamScanner

Monte Carlo Policy Evaluation:

→ Policy already given ⇒ π

→ Learn V_π from episodes of experience ~~using~~ following π.

$\forall$ $S_1, A_1, R_2, \ldots S_k \sim \pi$.

Earlier: we had <u>expected</u> return:

$$V_\pi(s) = E_\pi(G_T \mid S_t = s).$$

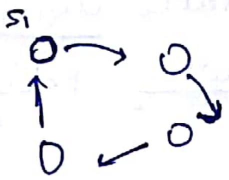
But here, we'll have <u>empirical mean</u> instead of expected.

Challenge: How to reset to a particular ~~step~~ state everytime and start our evaluation? And how to do this for every state?

2 methods:

① First visit: Imagine: circular MDP i.e. loop in an <u>MDP.</u>

Now, we consider only the time when a particular state, say S_1, is <u>1st visited</u> <u>in an episode</u>.



After that, update the count of # visits in that state: $N(s) += 1$

$$S(s) = S(s) + G_t$$

add reward from that time step onwards to the total reward for the state

estimated value for the state:
$$V(s) = \frac{S(s)}{N(s)}$$

According to the law of large numbers, the mean of a large # samples actually converges to the true mean

$$P\left(\left|\frac{X_1 + X_2 + \ldots X_n}{n} - \mu\right| < \varepsilon\right) \to 1$$

as $n \to \infty$, $\forall \varepsilon > 0$.

$X_1, X_2 \ldots X_n \rightarrow$ n samples of the distb

$\mu \rightarrow$ actual mean of the distribution

$\varepsilon \rightarrow$ any arbitrary small +ve value.

Scanned by CamScanner

Clarifications :

Q) Does # states ~~depends~~ decide the # episodes required ?

No !

From Central Limit Theorem, the variance decrease as a factor of $1/n$. Hence, it doesn't depend on the # states, just how many times a particular state is visited

The trajectory in each episode should be such that it covers most of the states that we are about.

~~Expt~~

Q) How to make sure that we reach all the states we care about,

→ Problem of control. (next lecture).

→ Here, we care about states under policy π and we make sure to visit those states by following policy π.

② Every visit → instead of just the 1st visit, add on for every visit → i.e. multiple visits possible for each episode.

(Choice depends on the domain).

Incremental mean :

$$\mu_k = \frac{1}{k}\left(\sum_{i=1}^{k} x_i\right) = \frac{1}{k}\left[x_k + \sum_{i=1}^{k-1} x_k\right]$$

$$= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) = \mu_{k-1} + \frac{1}{k}\underbrace{(x_k - \mu_{k-1})}_{\text{error term}}$$

Thus, we take a small step from our previous expectation following a error function.

as difference bet^n what we expected and what we actually got

Incremental Monte Carlo updates:

(Idea → want to do online learning instead of computing all values beforehand & then taking the mean).

$$N(s) = N(s) + 1.$$

$$V(s_t) = V(s_t) + \frac{1}{N(s)}(G_T - V(s_t)).$$

Better → add a weightage to discard old terms/ values.

$$V(s_t) = V(s_t) + \alpha(G_T - V_{s_t})$$

$$= (1-\alpha) V(s_t) + \alpha G_T$$

We can weigh older terms accordingly.

Temporal Difference Learning: (TD)

* Learn from incomplete episodes → bootstrapping
  → don't need to get to the end and figure out the reward, rather, go to an intermediate step and estimate the reward from there to the end.

So, we have to make an initial guess and after making/taking some steps, we make another guess of the final reward from that point. Then, we try to bring our initial guess closer to that guess.

Goal: Learn V_π online from experience under π.

Here, we update $V(s_t)$ towards the ~~expected~~ estimated return:

Simple TD learning: TD(0)

$$V(s_t) = V(s_t) + (\underbrace{\underbrace{R_{t+1} + \gamma V(s_{t+1})}_{\text{TD target}} - V(s_t)}_{\text{TD error}})$$

Advantages/Disadvantages

→ TD can learn before knowing the final outcome. i.e. online after every step.

→ TD can learn without the final outcome. i.e. it can be applied to processes that do not end. Hence, learning from incomplete sequences. (episodes)

TD update :

Step 2 :

Step 1 →

S_1

$S_1 \rightarrow S_2$

$V(S_1) = R_2$

$V(S_2) = R_3$

$V(S_1) = V(S_1) + (R_2 + V(S_2) - V(S_1))$

Step 3 :

$S_1 \rightarrow S_2 \rightarrow S_3$

$V(S_3) = R_4$, $V(S_2) = V(S_2) + (R_3 - V(S_3) - V(S_2))$

$V(S_1) = V(S_1) + (R_2 + V(S_2) - V(S_1))$.

i.e. update after each step.

TD → less noise

$G_T = R_{t+1} + \gamma R_{t+2} + \dots \gamma^{T-1} R_T$ → unbiased estimate of $V_\pi(S_t)$.

(Bias of an estimate ($\hat{\theta}_m$) is given by :

$\text{Bias}(\hat{\theta}_m) = E(\hat{\theta}_m) - \theta$, $\hat{\theta}_m$ → unbiased estimator when $E(\hat{\theta}_m) = \theta$, i.e. $\text{Bias}(\hat{\theta}_m) = 0$.)

E ↳ expectat^n

$R_{t+1} + \gamma V_\pi(S_{t+1})$ ↳ unbiased estimate of $V_\pi(S_t)$.

But, we use : $R_{t+1} + \gamma V(S_{t+1})$ → biased estimate as we don't know the true future $V_\pi(S_{t+1})$.

But $G_T = R_{t+1} + R_{t+2}\gamma + \dots R_T \gamma^{T-1}$

↳ adds noise at each stage (term)

$R_{t+1} + \gamma V(S_{t+1})$ → noise just in one term.

Hence, TD → less noisy.

MC → high variance, 0 bias
→ Good convergence
→ independent of initializatn.

TD → some bias, low variance
→ more efficient
→ TD (0) → $V_\pi(s)$
→ sensitive to init value.

Batch MC / TD

Both converge as experience $\to \infty$

In batch case,
→ repeatedly sample
$k \in [1, K]$.

Do they converge?

(eg. run 3 episodes & repeatedly iterate & learn from those 3).

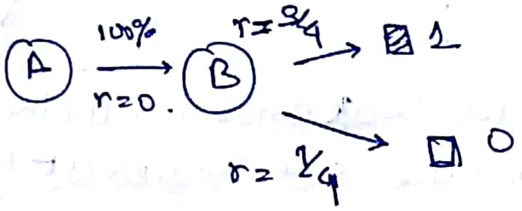
eg. There are observed

1 A 0 B 0
2 B 1
⋮
7 B 1
8 B 0

→ Monte carlo for $B = \frac{6}{8}$

→ Monte carlo for $A = \frac{0}{1}$

TD estimate → tries to build an MDP that best explains the data.



MC → reduces the mean squared error

• Best fit to actual returns received → $\sum_{k=1}^{K} \sum_{t=1}^{T_k} \left(g_t^k - V(s_t^k)\right)$

TD → converges to the MDP that best fits the data

→ soln to max likelihood Markov model

→ just by counting normal

$$P_{ss'}^{a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} I\,(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

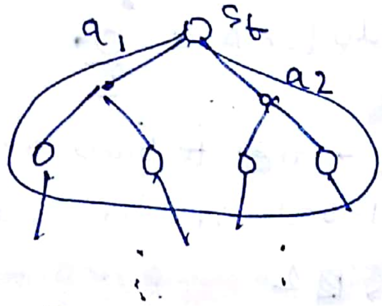
$$R_s^a = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} I(s_t^k, a_t^k = s, a)\, r_t^k.$$

TD makes use of markov property → much more efficient in those environments.

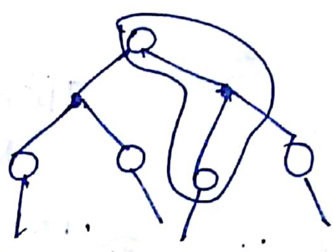
Situations maybe non-markov.

ie. Partially observed MDP → under the hood, there is an MDP, but what is observed is partial → so TD(0) doesn't work well on such environments & MC still finds a reasonable solution.

In DP → we did a one-step lookahead over all possible next states.

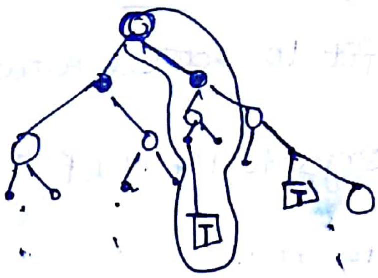


Whereas, for eg. in TD, we just look over one step, one sample



In MC, we look over until the entire terminate but only take one sample at each stage.

We could take all the way to the bottom for DP too, which would be called exhaustive tree search



→ Bootstrap → don't use the real value but the estimated value of the next step to update the current step.

- Not by MC
- By DP & TD

→ Sampling → don't do full width backup & only take samples.

- MC & TD samples
- DP doesn't.

Exhaustive search → Bad!

Algorithms that are in the middle: full width or sample backups & shallow or deep backups.

TD(λ).

n-step prediction → look n steps into the future [instead of one in TD(0)]

$n=1 \rightarrow$ TD(0): $G_T^{(1)} = R_{t+1} + \gamma V(S_{t+1})$

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

$\vdots$

$n=\infty$ $\quad$ MC. $\rightarrow G_T^{(\infty)} = R_{t+1} + \dots + R_T \gamma^{T-1}$

n-step return:

$\rightarrow G_T^{(n)} = R_{t+1} + \gamma R_{t+2} \dots R_{t+n} \gamma^{n-1} + \gamma^n V(S_t)$

n step TD

$\rightarrow V(S_{t}) = V(S_t) + \alpha (G_t^{(n)} - V(S_t))$

what value of n to be used? → Hard to choose & depends on problem

online / offline update →

↓ update the value immediately

→ update the value only after reaching the end.

Use average n-step returns. ← over several n.

how to do this efficiently for all n.

↓ TD(λ)

eg $\frac{G^{(2)} + G^{(4)}}{2}$ → get the best of all n.

$\lambda \rightarrow 0 < \lambda < 1$

λ return → geometrically weighted average of all n

decaying for each specific λ.

$(1-\lambda)$ → normalizing factor: $(1-\lambda) + (1-\lambda)\lambda + \dots$

~~$G_t^{\lambda} = (1-\lambda)\lambda^{n-1} \dots V(S_{t+n})$~~

sum $= (1-\lambda)\frac{1}{(1-\lambda)} = 1$

Scanned by CamScanner

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

$$TD(\lambda) \Rightarrow V(s_t) = V(s_t) + \alpha (G_T^{\lambda} - V(s_t))$$

Why geometric mean? → computational complexity
→ memoryless → doesn't require storing separate values for each timestep!

as $(1-\lambda), (1-\lambda)\lambda, (1-\lambda)\lambda^2 \ldots$
(just store one value and keep multiplying λ).

This is the forward view $TD(\lambda)$ → needs to wait until completing the ~~future~~ episode → suffers from same disadvantages as MC. ($\lambda = 1 \to$ MC)

Backward view → $TD(\lambda)$: compute the above stuff more efficiently without having to wait till the completion of the episode.

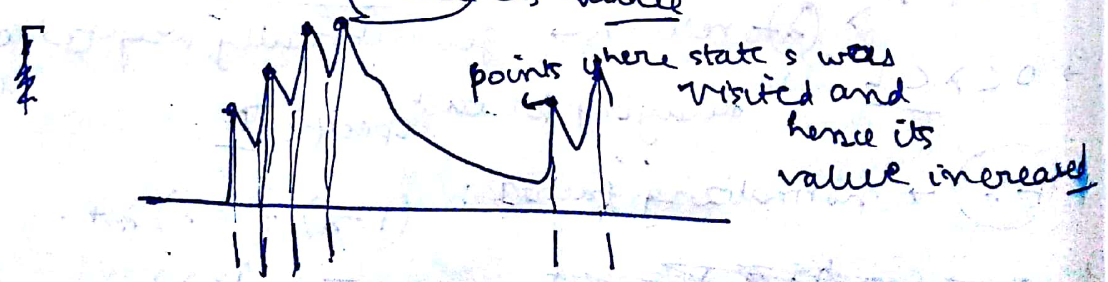
Eligibility traces

→ Frequency heuristic → assign importance to most frequent states
→ Recency heuristic → importance to most recent states
eligibility trace → combines both

$$E_0(s) = 0$$

$$E_t(s) = \gamma \lambda E_{t-1}(s) + I(s_t = s)$$

→ whenever state is visited, we increase the trace & if not visited, we reduce its value



$I(S_t = s) \rightarrow$ Identity function $= \begin{cases} 1 & \text{if } S_t = s \\ 0 & \text{else.} \end{cases}$

For backward view:

→ Keep ET for each state

→ update $V(s)$ in proportion to $E_t(s)$ & TD-error δ_t

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t).$$

for each state

$$V(s) = V(s) + \alpha E_t(s)\, \delta_t$$

→ λ → decaying factor.

→ $\lambda = 0$ → we squash the value of $E(s)$ to 0 if $S_t \neq s$

$$\therefore E_t(s) = I(S_t = s)$$

TD(0) Thus, value function updated for a state only if we observe that state in that time step. visit

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

$\lambda = 1 \rightarrow$ TD(1) same total update as MC.

Theorem → total offline updates same for backward as well as forward views.

$$\sum_{t=1}^{T} \alpha\, \delta_t\, E_t(s) = \sum_{t=1}^{T} \alpha \left(G_t^{\lambda} - V(S_t)\right) I(S_t = s)$$

TD(λ) → Spectrum betⁿ TD(0) & MC

$E_t(s)$ → can be computed efficiently too.