Model Free Prediction

A) Monte-Carlo Learning

MC methods learn directly from episodes of experience, we dont need knowledge of the MDP/transitions or rewards.

We look at complete episodes, eg we start a process like play a game then after a while it does finish.

We then go through our episodes, take sample returns for each episode, & then we average our them.

The only caveat is this works for episodic MDPs only, that is the episodes must terminate

B) Monte-Carlo Policy Evaluation

Goal: learn V_π from episodes of experience under policy π

$$S_1, A_1, R_2, \dots S_k \sim \pi$$

So given a policy π we take a look at the episode of the state, action, reward that follows

We can calculate the return from a particular state as well or rather the total discounted reward

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

And our value function is the expected return

$$V_\pi(s) = E_\pi[G_t | S_t = s]$$

However MC methods use empirical mean return instead of expected return

c) First Visit - Monte Carlo Policy Evaluation

Ideally a state appears at different times in different episodes, so every time it appears, we compute the returns from that point onwards
And then for all episodes we avg out return values for each state

Now lets say in a particular episode we have multiple visits to the state, so we can have a value for this state at a single particular time step onwards. So the question is whether to treat the multiple visits as multiple or one sample

In first visit MC, In each episode we only use the first time a state appears

So here if our goal is to evaluate a particular state s,

We maintain an $N(s)$ & $S(s)$ for each state
$N(s)$ tells us how many samples has s appeared in & $S(s)$ tells us sum of returns from those samples

So the first time-step t that state s is visited in an episode we increment $N(s) \leftarrow N(s) + 1$
We calculate return for state s given that time step in the episode
Then we also update ~~G~~ $S(s) \leftarrow G(s)_t + S(s)$

Then after multiple such episodes the value is estimated as $V(s) = \frac{S(s)}{N(s)}$

The law of large Numbers says if we do this enough times or as $N(s) \rightarrow \infty$, $V(s) \rightarrow V_\pi(s)$

D) Every-Visit Monti Carlo Policy Evaluation
The only difference is that at every visit of s in an episode we update $N(s)$ & $S(s)$ at all t's we encountering s

E) Incremental Mean:
The mean $\mu_1, \mu_2, \mu_3 \cdots$ of a sequence $x_1, x_2 \cdots$ can be computed incrementally,

$$\mu_k = \frac{1}{k} \sum_{j=1}^{k} x_j$$

$$= \frac{1}{k} \left(x_k + \sum_{j=1}^{k-1} x_j \right)$$

$$= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right)$$

$$= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)$$

F) Incremental Monti-Carlo Updates

Now, we take over MC Algorithm 2 instead of maintaining the mean, we update episodically now

-> Update $V(s)$ incrementally after episode $S_1, A_1, R_2 \ldots S_T$

-> For each state S_t with return G_t

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

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\left(G_t - V(S_t)\right)$$

In nonstationary problems, it can be useful to track a running mean i.e forget old episodes

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

with alpha we track move recent avg's

if α is large we highly sensitize to recent returns, forget past episodes quickly, if its small it means there are slow stable updates

9) Temporal Difference Learning
- TD methods learn directly from episodes of experience, it also applies to non episodic cases
- TD is model free when no knowledge of MDP transition or rewards are required
- We can learn from incomplete episodes, we can take a partial trajectory & then take an estimate for the actions ahead
- TD updates a guess towards a guess

10) MC & TD
Goal: learn v_π online from experience underpolicy π

Simplest Temporal-Difference learning Algorithm: TD(0)
→ Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$

$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

$R_{t+1} + \gamma V(S_{t+1})$ is called the TD target
$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error

Also here we simple assume G_t or the estimated return to be the immediate reward R_{t+1} plus the discounted value for the reward of the next state

1) Advantages & Disadvantages of MC vs TD

- TD can learn before knowing the final outcome
  TD learns online after every step & MC must wait until end of episode before return is known

- TD can learn without the final outcome
  It learns from incomplete sequences
  MC only learns from complete ones
  TD works in continuing (non-terminating) environments
  MC only works for episodic (terminating) environment

1) Bias / Variance trade off

- Return $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-1} R_T$ is unbiased estimate of $V_\pi(S_t)$
- True TD target $R_{t+1} + \gamma V_\pi(S_{t+1})$ is unbiased estimate of $V_\pi(S_t)$

However TD target $R_{t+1} + \gamma V(S_{t+1})$ is a biased estimate of $V_\pi(S_t)$

TD target is much lower variance than return
return depends on many random actions, transitions rewards, however target depends on just one random action, transition reward

k) MC has high variance, zero bias
- Good convergence properties
- Not very sensitive to initial value
- Very simple to understand & use

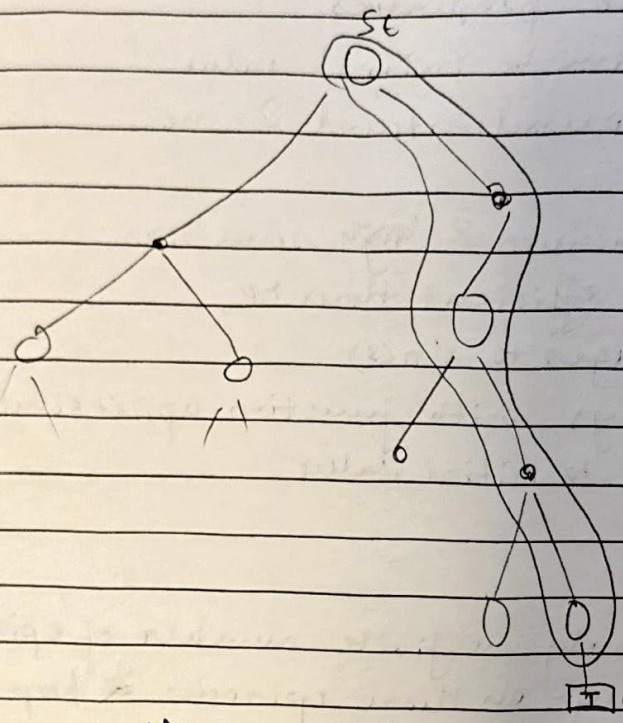
TD has low variance & some bias
- Usually more efficient than MC
- TD(0) converges to $V_\pi(s)$
(but not always with function approximation)
- More sensitive to initial value

l) Batch TD & MC
If we have only a finite number of episodes & keep iterating on those episodes & keep learning based on this same data, would MC & TD find the same solution?

Monti-Carlo tries to converge to solution which finds the best fit to the returns we see whereas TD(0) converges to the solution that best explains the data

m) TD exploits Markov Property - So this is usually more efficient in markov environments
But MC does not exploit Markov property, but usually more effective in non-markov environments

iv) MC Backup

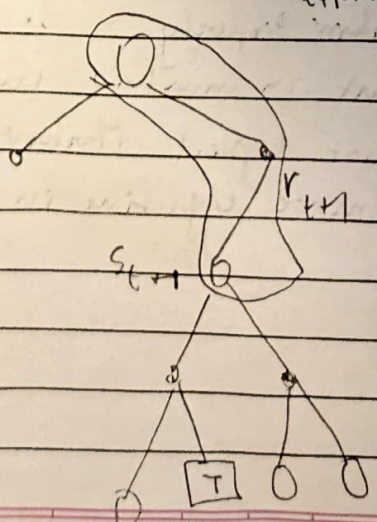


$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$

here for every state we update $V(S_t)$ given the returns from that point to the end

v) TD Backup

$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



here we update the value for S_t just given a single look ahead.

P) n-step Returns

Instead of a single step look ahead like we have for TD(0) we can have a multiple n step look ahead

∴ for n-step returns $n = 1, 2, \infty$

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

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

⋮

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

So we can define the n-step return

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

n-step temporal difference learning

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

1) Averaging n Step Returns

We can avg n-step returns over different n

eg average the 2 step & 4 step returns

$$\frac{1}{2} G^{(2)} + \frac{1}{2} G^{(4)}$$

We combine information from two different timesteps

2) TD-λ

The λ-return G_t^λ combines all n-step returns $G_t^{(n)}$

Using weight $(1-\lambda)\lambda^{n-1}$

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

Forward view

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t^\lambda - V(S_t))$$

The issue with the forward view is we still need to wait for n steps to compute $G_t^{(n)}$ or G_t^λ

5) Backward view

For every state we maintain an eligibility trace
Update value $V(s)$ for every state

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

in proportion to TD-error δ_t & Eligibility trace $E_t(s)$

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

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$

7) Understanding with an example

A → B → C → D → A → terminal
+1 +1 +1 +1 +0

$\gamma = 1$
$\lambda = 0.5$
$\alpha = 0.1$

We initialize $V(A) = V(B) = V(C) = V(D) = 0$
Now we track $E(s)$ & $V(s)$

With eligibility trace the goal is to update it for every state which has appeared till the given time step

- Thus at step 1 ((update $E_t(s) = \gamma\lambda E_{t-1}(s) + 1(s=s_t)$)
$E(A) = 1$ ($\delta = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$)
$E(B) = 0$ $V(s) = V(s) + \alpha \times \delta_t \times E_t(s)$)
$E(C) = 0$
$E(D) = 0$

For A → B $r = 1$

TD-error : $\delta = 1 + V(B) - V(A) = 1$

$V(A) \pm 0.1 \times 1 \times 1 = 0.1$

For step 2

B - C , $r = 1$

$E(A) = 0.5$

$E(B) = 1$

$E(C) = 0$

$E(D) = 0$

TD error

$\delta = 1 + V(C) - V(B) = 1$

$V(A) \pm 0.1 \times 1 \times 0.5 = 0.15$

$V(B) \pm 0.1 \times 1 \times 1 = 0.1$

For Step 3

C → D $\alpha = 1$

$E(A) = 0.25$

$E(B) = 0.5$

$E(C) = 1$

$E(D) = 0$

TD error

$$\delta = 1 + V(D) - V(C) = 1$$

Value updates

$V(A) \pm 0.025 = 0.175$

$V(B) \pm 0.05 = 0.15$

$V(C) \pm 0.1 = 0.1$

Step 4 (~~[illegible]~~) (Here is an error here yuti caught that too late)

D → A, $\alpha = 1$

$E(A) = 0.125$ ~~[illegible]~~ $= 0.125$

$E(B) = 0.25$

$E(C) = 0.5$

$E(D) = 1$

$$\delta = 1 + V(A) - V(D) = 1 + 0.175 - 0 = 1.125$$

Value updates

$V(A) \pm 0.1 \times 1.175 \times 0.125 = +0.014$ (Jarvis to be added)

$V(B) \pm 0.1 \times 1.175 \times 0.25 = +0.029$

$V(C) \pm 0.1 \times 1.175 \times 0.5 = +0.059$

$V(D) \pm 0.1 \times 1.175 \times 1 = +0.118$

$\therefore V(A) = 0.1896$

$V(B) = 0.174$

$V(C) = 0.159$

$V(D) = 0.118$

Step 5 A → Term

$E(A) = 0.0625 + 1 = 1.0625$

$E(B) = 0.125$

$E(C) = 0.25$

$E(D) = 0.5$

TD error

$\delta = 0 + 0 - V(A) = -0.1896$

Then update $V(A), V(B), V(C), V(D)$

The point is at every time step you update E for all states seen till the given time step & calculate the error (TD error) & propagate to back to every state seen in the episode.

∴ V(A) = 0.1896
V(B) = 0.174
V(C) = 0.159
V(D) = 0.118

Step 5 A → turn

E(A) = 0.0625 + 1 = 1.0625
E(B) = 0.125
E(C) = 0.25
E(D) = 0.5

TD error

$\delta = 0 + 0 - V(A) = -0.1896$

Then update V(A), V(B), V(C), V(D)

The point is at every time step you update E for all states seen till the given time step & calculate the error (TD error) & propagate to back to every state seen in the episode.