Temporal difference Learning

Learn directly from episodes of experiance

Model free like MC

Learns from incomplete episodes.

No requirement of termination

Bootstraps → update guess towards guess

TD updates the estimates as it progresses.

TD is one step lookahead

TD requires that reward is calculated at every transition

TD is better than MC.

TD exploits Markov property
MC does not exploit Markov property

TDL is a predition algorithm, which means it only predicts V given π. It does not optimize π

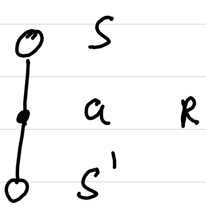
The goal here is only to find V values for all states given a policy π.

Incremental every visit Monte carlo Prediction

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

In T.D., we replace G_t by estimated reward

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



Estimated return at s = Reward at s' + Goodness of s'

So, we need only info about 2 states and action for TDL. We dont need complete episode

Bias & Variance in TDL & MC

TD can learn before knowing the final outcome unlike MC which must wait until the end of the episode before return is known

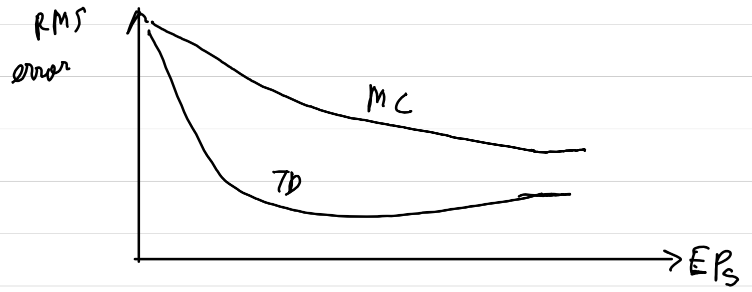
Return $G_t = R_{t+1} + \gamma R_{t+2} \cdots$ of MC is unbiased estimate.

But $R_{t+1} + \gamma V(S_{t+1})$ is biased estimate

G_t depends on many random actions, transitions and rewards

TD depends on one action, transition and reward

MC has high variance, zero bias
TD has low variance, some bias



In Monte carlo, G can drastically differ between episodes. Also due to delayed rewards, a lot of variance is introduced

TD updates are performed at each time step allowing the value function to converge smoothly. Each small update contributes only a portion of the total change needed which smooths the learning process and reduces the variance of each update

Frequent updates lead to incremental improvements that average out fluctuations caused by noisy rewards or random events.

TD relys on immediate next reward, that is local information rather than the entire episode. This helps avoid large swings in estimates that might result from unusual rewards further along an episode, thus keeping the update variance lower.

Batch updating TD & MC

Updates are postponed till a collection of experiances has been gathered. Then values are updated in one go.

Accumulate error in various transitions and update in bulk

Reduce variance and increase stability in learning.

faster convergence as updates are in bulk

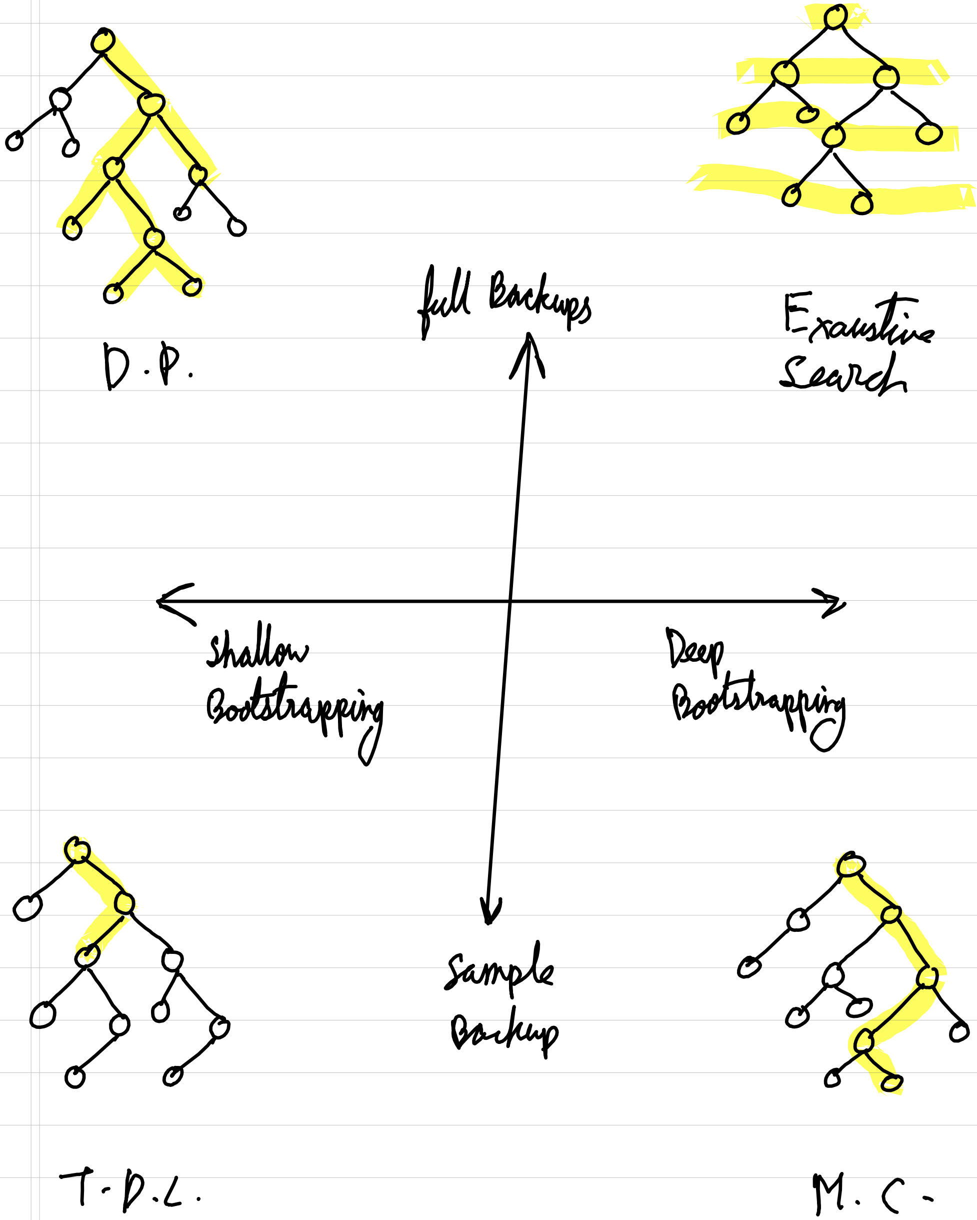
Makes model more robust

Useful for non stationary and complex environments

Average out the variability in episodic returns.

Monte Carlo	Temporal difference Learning
Complete episodes	Incomplete episodes also
Actual return G_t	Estimated return $R_t + V(S_{t+1})$
Low Bias	High bias
High variance	Low variance
Need episode to terminate	Can work for continuing tasks
No Bootstrapping	Bootstrapping
Computationally expensive	Cheaper computations
doesnt exploit markov property	Exploits markov property

Comparison of Methods



$V(S) \leftarrow V(S) + \alpha(R_t + \gamma V(S') - V(S))$

$V(S) \leftarrow V(S) + \alpha(G_t + V(S))$

SARSA, Q also are here

Model dependant	Model free
D.P. Exaustive search	MC, TDL, SARSA, Q-L
require model	No model required
Learn through equations	learn through episodes
Info of model, P, γ defined	learning through model interaction

Bootstrapping → guess towards guess

D.P.	Exaustive Search
Only all nodes of a single path explored	All nodes of all the paths explored
Fast	Slow (infeasable)

TDL, SARSA, QL	MC
Estimate the reward	G_t direct reward calculated
High bias, low variance	low bias, High variance
Useful when intermediate rewards known	Useful when model is black box

Bootstrapping makes the process faster

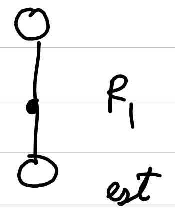
n- step TDL

MC uses Deep backups
TDL uses No backups

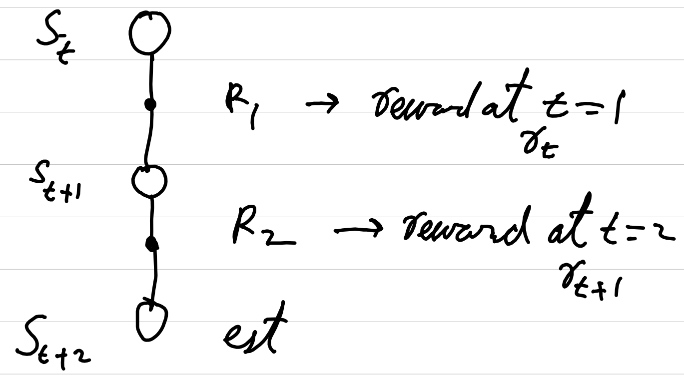
Middle ground is n-step bootstrapping
Instead of using only 1 reward & future estimates we can use n rewards.

$n = 1$: Normal

$$G_t^1 = r_1 + \gamma V(S_{t+1})$$



$n = 2$: 2 step look ahead



$$\text{goodness} = r_t + \gamma r_{t+1} + \gamma^2 V(S_{t+2})$$

(Expected reward)

$$\text{goodness} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \gamma^n V(S_{t+n})$$

That is, the observed return becomes

$$G_t^n = \sum_{i=0}^{n} r_{t+i} \gamma^i$$

If episode terminates beforehand →

$$G_t^n = \sum_{i=0}^{\min(T,n)} r_{t+i} \gamma^i$$

$$\therefore \quad V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^n - \gamma^n V(S_{t+n}) - V(S_t) \right)$$

Basic idea of n step is that the V value is updated after n episodes..

Update V

	TDL	n-step	Monte carlo
no of ep	1	n	all
		actually $\min(n, \text{all})$	

Choosing n can be difficult
Book keeping (sliding window etc) required

Consider a path followed by an episode the rewards obtained and previous v values

$\alpha = 0.5, \gamma = 0.9$

$$\begin{tikzcd}
\underset{V=2}{A} \arrow[r, "1"'] & \underset{V=0}{B} \arrow[r, "2"'] & \underset{V=3}{C} \arrow[r, "4"'] & \underset{V=1}{D} \arrow[r, "5"'] & \underset{V=2}{E} \arrow[d, "7"] \\
 & & & & \underset{V=3}{F}
\end{tikzcd}$$

TDL

$$\begin{tikzcd}
A \arrow[d, "1"] \\
B \; V=0
\end{tikzcd}$$

$$V_A = V_A - \alpha\left(r_t + V_B \cdot \gamma - V_A \right)$$
$$= 2 - 0.5\left(1 + 0 - 2 \right)$$

3 step TDL

$$\begin{tikzcd}
A \arrow[d, "1"] \\
B \arrow[d, "2"] \\
C \arrow[d, "4"] \\
D
\end{tikzcd}$$

$$V_A = V_A - \alpha'\left(G_t + \gamma^3 V_D - V_A \right)$$
$$= 2 - 0.5\left(1 + 2 \times 0.9 + 4 \times 0.9^2 + 1 \times 0.9^3 - 2 \right)$$

Monte Carlo

$$V_A = V_A - \alpha\left(G_t - V_A \right)$$
$$= 2 - 0.5\left(1 + 2 \times 0.9 + 4 \times 0.9^2 + 5 \times 0.9^3 + 7 \times 0.9^4 - 2 \right)$$

Parameters of T.D.L.

① Learning rate α → Between 0 – 1

Large learning rate might lead to fluctuating training results

Small learning rate adjusts slowly, which will take more time to converge

$\alpha = 1$: fully trust most recent information discarding previous knowledge

$\alpha = 0$: Ignore new information, never learns or updates the knowledge

---

② Gamma γ → The discount rate. Represents how much we are valuing future rewards.

Bigger the discount rate, more we value future rewards.

Generally between 0 – 1

High γ (close to 1) future rewards valued (farsighted)
Low γ (close to 0) Intermediate rewards proritized (Nearsighted)

---

③ Epsillion ε → Exploration vs Exploitation

Prob e → Explore
Prob $1-e$ → exploit

Larger ε means more exploration while training