Lecture notes - Introduction to Reinforcement learning with David Silver

Lecture 4

By - Amal Sunny

Keywords - Model free, Monte-carlo, Temporal learning, TD(λ)

Model-free Prediction

- Predicting(evaluating policy) without knowledge of model
- Estimating the value function of an unknown MDP
- Multiple ways to do it:
  - Monte-Carlo learning
  - Temporal-difference learning
  - TD(λ)

Monte-Carlo Learning

- Not the most efficient, but extremely effective and widely used.
- To learn directly from episodes of experience (thus dont need knowledge of MDP)
- Essentially it takes the sample returns from various routes and averages over them to estimate value of that.
- **However**, this is limited to episodic MDPs.

Policy Evaluation

- **Goal**: learn v_π from episodes under policy π
- Value function is expected return
  - $$v_\pi(s) = E_\pi[G_t|S_t = s]$$
- However, Monte-Carlo uses empirical mean return instead of expected return.
- Two ways to evaluate under monte-carlo

First-Visit Monte-Carlo Policy evaluation

- To evaluate for a state s
- Essentially we consider the first time a state s is visited in an episode - a counter is incremented and total return is updated for that iteration.
  - This counter is retained across episodes

- And we average it across all returns by the counter count.
  - This average is done across episodes(otherwise N(s) is just 1 every iteration at best)
- i.e $V(s) = S(s)/N(s)$
  - N(s) - counter
  - S(s) - total return
- By law of large numbers, $V(s) \to v_\pi(s) \text{ as } N(s) \to \infty$
- Sampling here ensures dependency on size of the problem does not arise.

Every visit Monte-Carlo Policy evaluation

- Same as First-visit, but **every** visit to state s (not just once per episode) is considered in the average.
- Incase of better, both methods have their own domains they are better than each other in.

Incremental Monte-Carlo updates

- Incremental mean - the mean for a sequence can be computed incrementally for each element instead of at the end for all.

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

(easily derivable)

- Here, $\mu_k = \text{ mean of first k elements }, x_k = \text{k'th element}$
- For Monte-Carlo, we update V(s) incrementally after every episode(not every step t).
- For each state S_t with return G_5

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

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

- Sometimes when we want to forget older episodes or reduce their impact we can have an fixed step size to make it exponentially fade away.

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

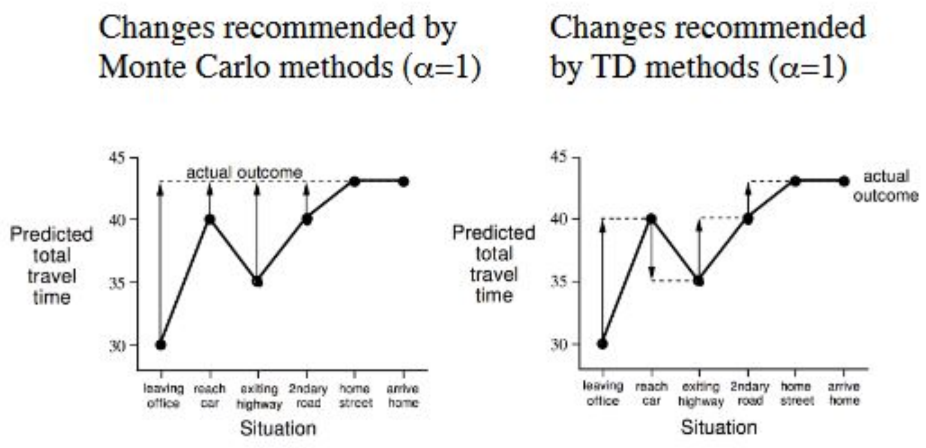
Temporal-Difference(TD) Learning

- TD methods learn directly from actual experience(interacting with the environment)
- TD is model free

- Main difference v/s Monte-Carlo is TD can learn from incomplete episodes - partial experiences are used along with estimates of the rest instead of the entire return.
    - This substitution is called bootstrapping
- TD updates a guess towards another guess made after.

MC and TD

- Goal: to evaluate v_π online(every step update) from experience under policy π
- So in Monte_Carlo (incremental every-visit)
    - Updated value $V(S_t)$ towards *actual return* G_t
    - $$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t)$$
- Taking the simplest TD algorithm: TD(0)
    - Update 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
- The updation interval remains the biggest difference and we can see so in this example below:



Bias/Variance Trade-off

- Return 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, we don't know $v_\pi(S_{t+1})$ usually
- TD target = $R_{t+1} + \gamma V(S_{t+1})$ is a biased estimate of $v_\pi(S_t)$
    - This bias is introduced due to our guess about $v_\pi(S_t)$
- However, TD target has much lower variance than the return as:
    - Return depends on many random actions, transitions, rewards

    - TD target depends on **one** random action, transition, reward

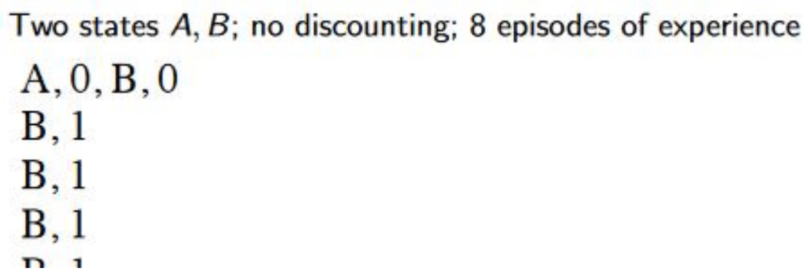
Advantages and Disadvantages of MC vs TD

- TD can learn before the final outcome
    - TD can learn online every step
    - MC must wait till end of episode to know return and update
- TD can learn without the final outcome
    - Useful where we have incomplete sequences
        - MC can only learn from complete sequence
    - And continuing (non-terminating) environments
        - MC only works for episodic(terminating) environemnts
- MC has high varience, zero bias
    - Good convergence properties
        - Even with function approx.
    - Not sensitive to initial value of value function
    - Simple to understand and use
- TD has low variance, some bias
    - Usually more efficient than MC
    - TD(0) converges to $v_\pi(s)$
        - But not always with function approx.
    - More sensitive to initial value of value function
- TD exploits Markov property - by building the implicit MDP like structure and solving it
    - Usually more effective in markovian environments
- MC ignores Markov property
    - Usually more effective in non-markovian environment (partially observed, signals are messy)

> Non-markovian environment does not mean there isn't an MDP anymore. It means the observed environment to the agent does not have enough info for an MDP but the environment still functions as an MDP.

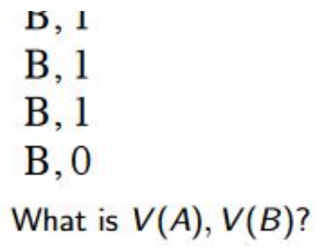
Batch MC and TD

- We know both MC and TD converge as experience $\rightarrow \infty$
- However, for finite experience that is no longer valid.
- We take an example to show this:



Two states A, B; no discounting; 8 episodes of experience

$A, 0, B, 0$
$B, 1$
$B, 1$
$B, 1$


B, 1
B, 1
B, 1
B, 0
What is $V(A), V(B)$?

- If we calculate via TD, we get V(A) = 0.75
- For MC, V(A) = 0

This fundamental difference is due to the fact that both converge, trying to minimize/maximize different terms.

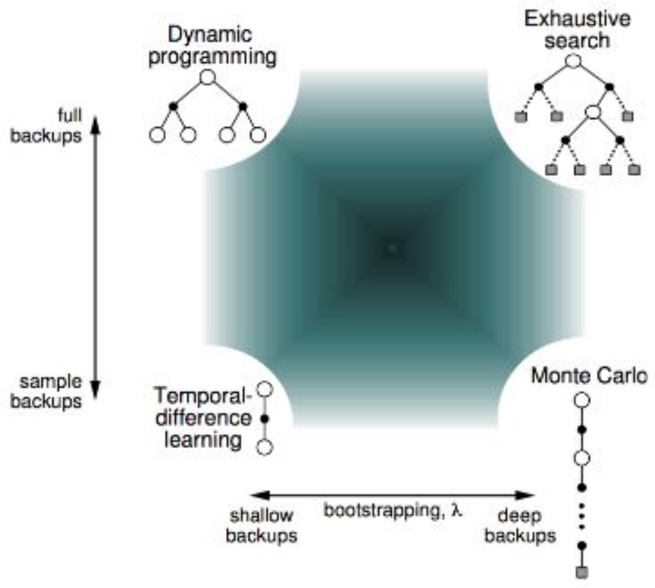
- **MC converges to solution with minimum mean-square error**
- **TD converges to solution of max likelihood Markov model for observed data.**

Bootstrapping and Sampling

- Bootstrapping: updating involves estimating\guessing values
  - MC does not bootstrap
  - DP and TD do.
- Sampling: updates values based on a full length sample(entire route taken till end)
  - MC and TD sample
  - DP does not.

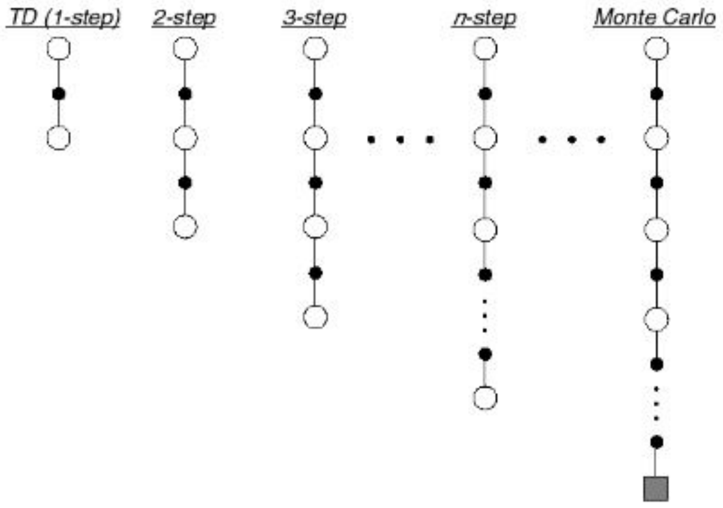
Unified view of RL

The image gives a good view of all approaches to RL covered so far



TD(λ)

- The TD target is allowed to look n steps ahead, instead of just 1



n-Step Return

- We define n-step returns for $n = 1, 2, ..., \infty$:

$$\begin{array}{cc} n = 1 & (\text{TD})\ G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\ n = 2 & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\ \vdots & \vdots \\ n = \infty & (\text{MC})\ G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T \end{array}$$

- And that gives us the n-step return

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

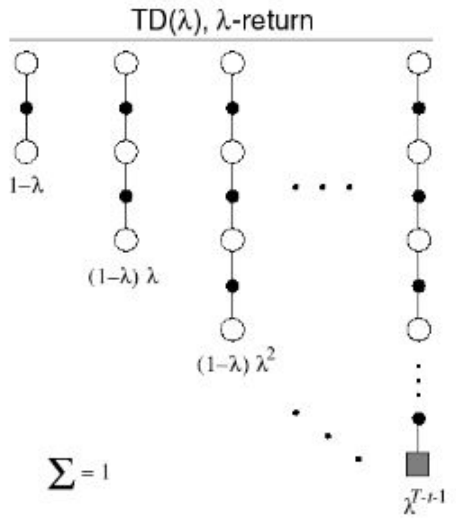
- And with that we have n-step TD learning as:

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

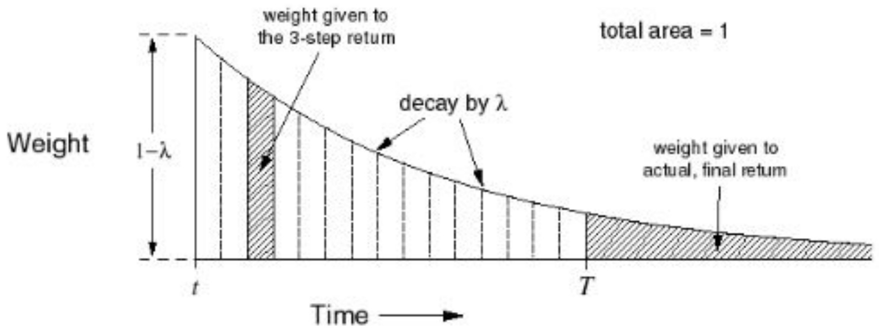
Averaging n-Step returns

- Instead of just taking one n-step return, we can average multiple of them over different n
- eg: average the 2-step and 4-step returns
- So we effectively get information from two different steps in the same process
- Then, can't we extend this to **all** steps ? Yes.

λ-return



- We take all the n-step returns possible for each update in λ-return under $G_t^{(n)}$
- λ-return is a geometrically weighted return of all n-step returns
  - λ - gives the decay factor
  - Taken as a GP for compuational optimal reasons
- Each n-step return gets a decaying weight of $(1-\lambda)\lambda^{n-1}$



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

- For the forward-view TD(λ), we take this as the target term to base error off

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

Forward-view TD(λ)

- Updates value function towards the λ-return
- Looks into the future till the end

- Functions very much like the MC approach, waits till the end to update
- Thus, only can be computed from complete episodes

Backward View TD(λ)

- The forward view provides theory upon which the backward view draws upon
- It retains all the good features: updates online, every step, from incomplete sequences

> ### Eligibility Traces
>
> - There are two heuristics we use to assign credit to cause of some event
>   - Frequency Heuristic: assigns credit to most frequent states
>   - Recency heuristic: assigns credit to most recent states
>   Eligibility trace combines both heuristics
>
> $$E_0(s) = 0$$
> $$E_t(s) = \gamma\lambda E_{t-1}(s) + 1(S_t = s)$$

- Building on this eligibilty trace, Backward view keeps an eligibility trace for every state s
- Updation of value V(s) for every state s is done proportionally to TD-error δ_t and eligibility trace $E_t(s)$

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

- This makes it so that we're looking backwards as we go on, as eligibility trace relies on older value of it.

TD(λ) and TD(0)

- When λ = 0, only current state is updated

$$E_t(s) = 1(S_t = s)$$
$$V(s) \leftarrow V(s) + \alpha\delta_t E_t(s)$$

- When you look at it, this is exactly equivalent to TD(0) update

$$V(S_t) \leftarrow V(S_t) + \alpha\delta_t$$

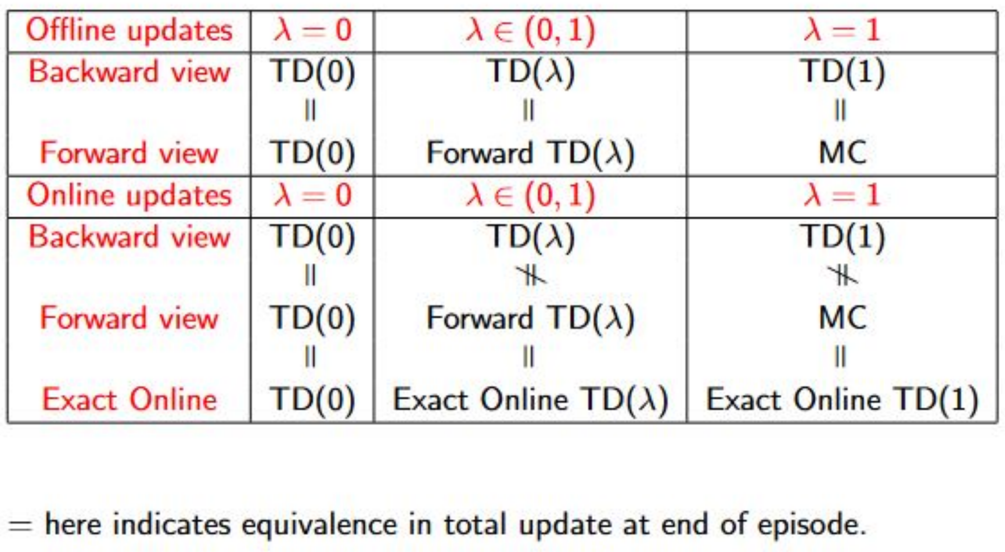
TD(λ) and MC

- When we have λ=1, credit is deffered until end of episode(like MC)
- Considering episodic environments with offline updates
    - Over the course of an entire episode - the total update for TD(1) is the same as for MC

> Theorem
> The sum of offline updates is identical for forward-view and backward-view TD(λ)

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

Summary of Forward and Backward TD(λ)

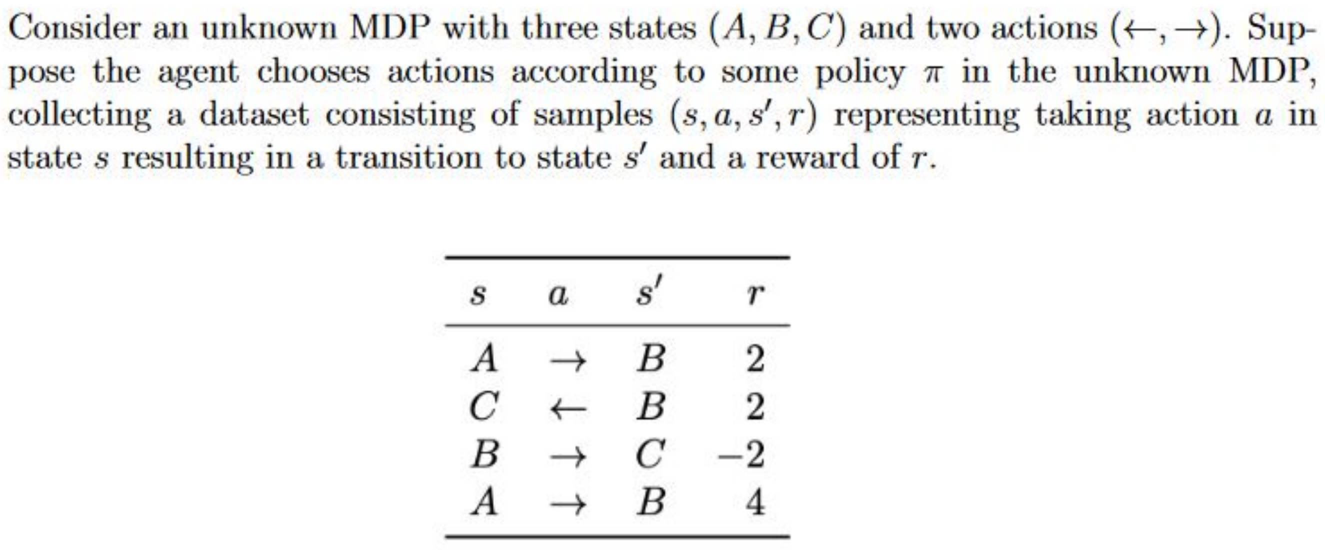


Offline updates	$\lambda = 0$	$\lambda \in (0,1)$	$\lambda = 1$
Backward view	TD(0)	TD(λ)	TD(1)
	$\parallel$	$\parallel$	$\parallel$
Forward view	TD(0)	Forward TD(λ)	MC
Online updates	$\lambda = 0$	$\lambda \in (0,1)$	$\lambda = 1$
Backward view	TD(0)	TD(λ)	TD(1)
	$\parallel$	$\nparallel$	$\nparallel$
Forward view	TD(0)	Forward TD(λ)	MC
	$\parallel$	$\parallel$	$\parallel$
Exact Online	TD(0)	Exact Online TD(λ)	Exact Online TD(1)

$=$ here indicates equivalence in total update at end of episode.

Problems

Problem 1: Taken from CS234 Stanford questions



Consider an unknown MDP with three states (A, B, C) and two actions $(\leftarrow, \rightarrow)$. Suppose the agent chooses actions according to some policy π in the unknown MDP, collecting a dataset consisting of samples (s, a, s', r) representing taking action a in state s resulting in a transition to state s' and a reward of r.

s	a	s'	r
A	$\rightarrow$	B	2
C	$\leftarrow$	B	2
B	$\rightarrow$	C	-2
A	$\rightarrow$	B	4

You may assume a discount factor of $\gamma = 1$.

Recall the update function of Q-learning is:

$$Q(s_t, a_t) = (1-\alpha) \cdot Q(s_t, a_t) + \alpha \cdot (r_t + \gamma max_{a'} Q(s_{t+1}, a')) \tag{1}$$

Assume that all Q-values are initialized to 0, and use a learning rate of $\alpha = \frac{1}{2}$.

(a) Run Q-learning on the above experience table and fill in the following Q-values:

$Q(A, \rightarrow) =?$

$Q(B, \rightarrow) =?$

Ans:

Updating for every action, one by one

$$Q_1(A, \rightarrow) = 1/2 \cdot Q_0(A, \rightarrow) + 1/2(2 + \gamma max_{a\prime} Q(B, a\prime)) = 1$$
$$Q_1(C, \leftarrow) = 1$$
$$Q_1(B, \rightarrow) = 1/2(-2+1) = -1/2$$
$$Q_2(A, \rightarrow) = 1/2 \cdot 1 + 1/2(4 + max_{a\prime} Q_1(B, a\prime)) = 1/2 + 1/2(4+0) = 5/2.$$

Problem 2: Cont.d from above

After running Q-learning and producing the above Q-values, you construct a policy π_Q that maximizes the Q-value in a given state: $\pi_Q(s) = argmax_a Q(s, a)$. What are the actions chosen by the policy in states A and B?

Ans:

From the table we can see $Q(A, \rightarrow) > Q(A, \leftarrow) = 0$

Thus,

$\pi_Q(A) = \rightarrow$

For B, $Q(B, \leftarrow) = 0 > Q(B, \leftarrow) = -1/2$

Thus,

$\pi_Q(B) = \leftarrow$