Lecture 4: Model-Free Prediction

David Silver

Outline

Model-Free Reinforcement Learning

- Last lecture:
  - Planning by dynamic programming
  - Solve a *known* MDP
- This lecture:
  - Model-free prediction
  - Estimate the value function of an *unknown* MDP
- Next lecture:
  - Model-free control
  - Optimise the value function of an *unknown* MDP

Monte-Carlo Reinforcement Learning

- MC methods learn directly from episodes of experience
- MC is *model-free*: no knowledge of MDP transitions / rewards
- MC learns from *complete* episodes: no bootstrapping
- MC uses the simplest possible idea: value = mean return
- Caveat: can only apply MC to *episodic* MDPs
  - All episodes must terminate

Caveats : MC is applicable for episodic tasks, i.e. where the tasks terminate.
we can calculate mean returns only upon termination of the episode

Monte-Carlo Policy Evaluation

Evaluation is subject to the policy π that we opt to evaluate.

- Goal: learn v_π from episodes of experience under policy π

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

- Recall that the *return* is the total discounted reward:

  this will be different for every time step in the episode

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

- Recall that the value function is the expected return:

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

- Monte-Carlo policy evaluation uses *empirical mean* return instead of *expected* return

  How we measure the returns for each state since we don't get to reset our MDP to a particular state??? Two different ways in the coming slides...

First-Visit Monte-Carlo Policy Evaluation

Since we are evaluating a certain policy, following the policy is enough to ensure we will visit the states that are reachable under this policy.
We then need to make sure that we visit a certain state several times under the policy we evaluate

- To evaluate state s
- The first time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$ counters [N(s) and S(s)] persist over all the episodes (across our samples)

  we need to track the mean returns for every state in the trajectory
- Increment total return $S(s) \leftarrow S(s) + G_t$ Gt measures returns from first visit to state s until end of the episode
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- By law of large numbers, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

By sampling we break the dependency on the size of the problem (i.e. state-space size). Remember, that standard error (variance of the estimate) decreases as a function of the sample size : SE = standard deviation / sqrt(sample size)

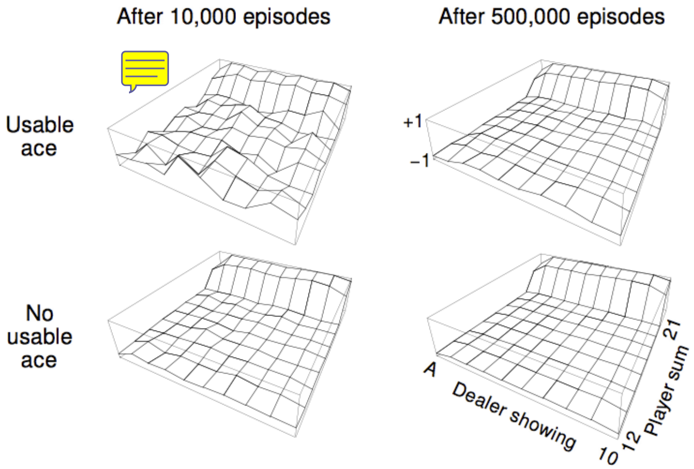
Every-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- Every time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$ increment every time a state is visited
- Increment total return $S(s) \leftarrow S(s) + G_t$ increment by a different number every time based on the point you are at, at that point t
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- Again, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

Blackjack Example

- States (200 of them):
  - Current sum (12-21)
  - Dealer's showing card (ace-10)
  - Do I have a "useable" ace? (yes-no)
- Action stick: Stop receiving cards (and terminate)
- Action twist: Take another card (no replacement)
- Reward for stick:
  - $+1$ if sum of cards $>$ sum of dealer cards
  - 0 if sum of cards $=$ sum of dealer cards
  - -1 if sum of cards $<$ sum of dealer cards
- Reward for twist:
  - -1 if sum of cards > 21 (and terminate)
  - 0 otherwise
- Transitions: automatically twist if sum of cards < 12

Blackjack Value Function after Monte-Carlo Learning



Policy: stick if sum of cards ≥ 20, otherwise twist

Here we are only after evaluating the said policy, rather than actually finding the optimal policy

Incremental Mean

The mean $\mu_1, \mu_2, ...$ of a sequence $x_1, x_2, ...$ can be computed incrementally,

$$\begin{aligned}
\mu_k &= \frac{1}{k} \sum_{j=1}^{k} x_j \\
&= \frac{1}{k} \left(x_k + \sum_{j=1}^{k-1} x_j \right) \\
&= \frac{1}{k} \left(x_k + (k-1)\mu_{k-1} \right) \\
&= \mu_{k-1} + \frac{1}{k} \left(x_k - \mu_{k-1} \right)
\end{aligned}$$

Online algorithm to calculate the mean.
Think of this as having an estimate of what the mean will be (μκ-1) and upon drawing a new sample, this estimate gets updated by a fraction (1/k) of the error (Xκ-μκ-1)

Incremental Monte-Carlo Updates

- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, ..., S_T$
- 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 non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

$$V(S_t) \leftarrow V(S_t) + \alpha \overset{\text{error in estimate}}{\boxed{(\underset{\boxed{\text{actual return}}}{G_t} - \underset{\boxed{\text{estimate of return}}}{V(S_t)})}}$$

Temporal-Difference Learning

TD applies to non-episodic tasks as well

- TD methods learn directly from episodes of experience
- TD is *model-free*: no knowledge of MDP transitions / rewards
- TD learns from *incomplete* episodes, by *bootstrapping*
- TD updates a guess towards a guess

we do not need the full sequence of transitions. we can also do with incomplete sequences. bootstrapping is the idea of substituting the trajectory with estimates for the remainder of the trajectory, estimates of what will happen from a certain point onwards

MC and TD

- Goal: learn v_π online from experience under policy π
- Incremental every-visit Monte-Carlo
  - Update value $V(S_t)$ toward *actual* return G_t

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

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward *estimated* return $R_{t+1} + \gamma V(S_{t+1})$

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

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

One advantage of the TD approach is that you get immediate feedback. Where in MC setting you would only judge by the final outcome, in TD you get to observe more details of the progress and update accordingly (example with near-death experience while driving)

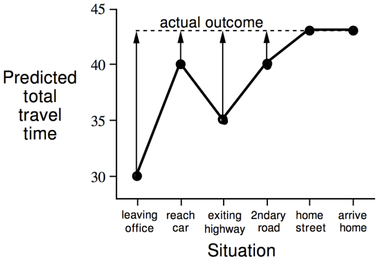
Driving Home Example

State	Elapsed Time (minutes)	Predicted Time to Go	Predicted Total Time
leaving office	0	30	30
reach car, raining	5	35	40
exit highway	20	15	35
behind truck	30	10	40
home street	40	3	43
arrive home	43	0	43

Driving Home Example: MC vs. TD

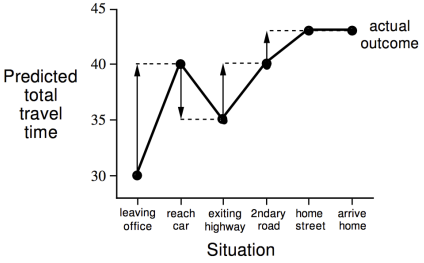
Example here is an MRP, we are after policy evaluation, we are not after optimizing our actions.
Remember that any MDP can be flatten out into an MRP given a specific policy.

Changes recommended by Monte Carlo methods (α=1)



in MC, in every step update happens against the final outcome

Changes recommended by TD methods (α=1)



in TD, updates take place against the most updated estimate that existed by that time

Advantages and Disadvantages of MC vs. TD

- TD can learn *before* knowing the final outcome
  - TD can learn online after every step
  - MC must wait until end of episode before return is known
- TD can learn *without* the final outcome
  - TD can learn from incomplete sequences
  - MC can only learn from complete sequences
  - TD works in continuing (non-terminating) environments
  - MC only works for episodic (terminating) environments

Bias/Variance Trade-Off

- Return $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$ is *unbiased* estimate of $v_\pi(S_t)$ in MC
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is *unbiased* estimate of $v_\pi(S_t)$ if there was an oracle revealing the actual value - then we wouldn't have any bias
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimate of $v_\pi(S_t)$
  bias occurs from the fact that V(St+1) != vπ(St+1)
- TD target is much lower variance than the return:
  - Return depends on *many* random actions, transitions, rewards
  - TD target depends on *one* random action, transition, reward

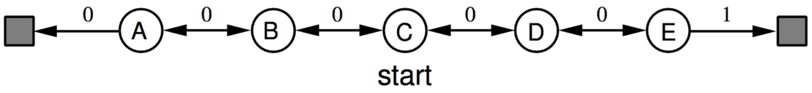
  variance in Gt = Rt+1 + γRt+2 + ... + γ^(T-1)*RT comes from the various future states and discounting factors. However the TD target may be biased but has much lower variance

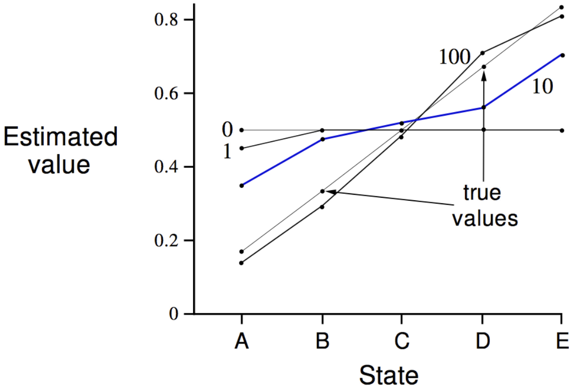
Advantages and Disadvantages of MC vs. TD (2)

- MC has high variance, zero bias
  - Good convergence properties
  - (even with function approximation)
  - Not very sensitive to initial value
  - Very 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 approximation)
  - More sensitive to initial value

Random Walk Example

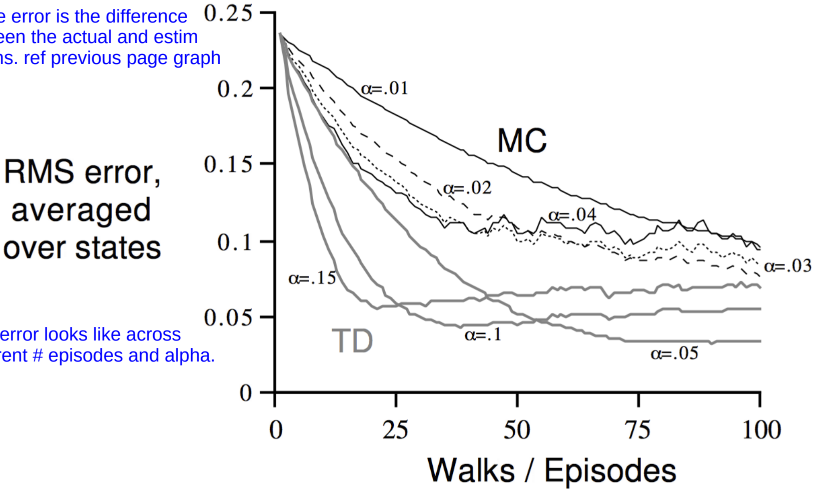
random uniform action space : left (0.5) and right (0.5).





Random Walk: MC vs. TD

where error is the difference between the actual and estim returns. ref previous page graph



how error looks like across different # episodes and alpha.

Batch MC and TD

what would happen if we were to use a finite number of episodes and apply
MC|TD over and over again using this finite number of episodes??

- MC and TD converge: $V(s) \to v_\pi(s)$ as experience $\to \infty$
- But what about batch solution for finite experience?

$$
\begin{gathered}
s_1^1, a_1^1, r_2^1, ..., s_{T_1}^1 \\
\vdots \\
s_1^K, a_1^K, r_2^K, ..., s_{T_K}^K
\end{gathered}
$$

- e.g. Repeatedly sample episode $k \in [1, K]$
- Apply MC or TD(0) to episode k

AB Example

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

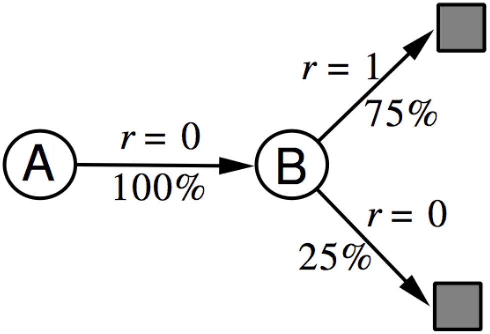
$\mathrm{A}, 0, \mathrm{B}, 0$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 0$

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

AB Example

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 $V(A), V(B)$?

Certainty Equivalence

- MC converges to solution with minimum mean-squared error
  - Best fit to the observed returns

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} \left(G_t^k - V(s_t^k)\right)^2$$

  - In the AB example, $V(A) = 0$
- TD(0) converges to solution of max likelihood Markov model
  - Solution to the MDP $\langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \rangle$ that best fits the data

$$\hat{\mathcal{P}}_{s,s'}^{a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\hat{\mathcal{R}}_{s}^{a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k = s, a) r_t^k$$

  - In the AB example, $V(A) = 0.75$

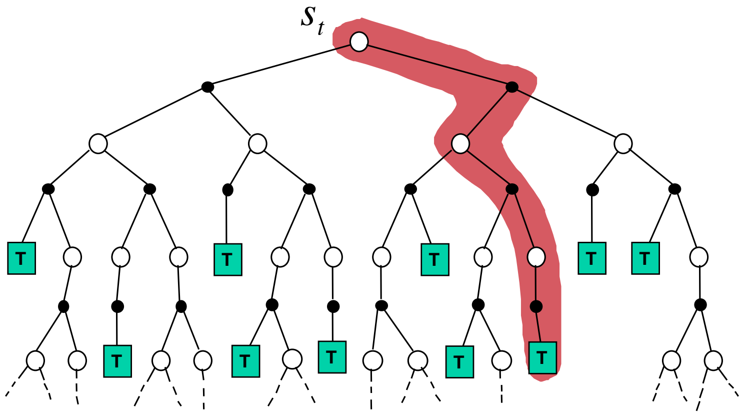
Advantages and Disadvantages of MC vs. TD (3)

TD understands problem structure in terms of transitions and states and the fact that every state has to summarize what has happened before and exploits this by implicitly building and then solving for the underlying MDP

- TD exploits Markov property
  - Usually more efficient in Markov environments
- MC does not exploit Markov property
  - Usually more effective in non-Markov environments

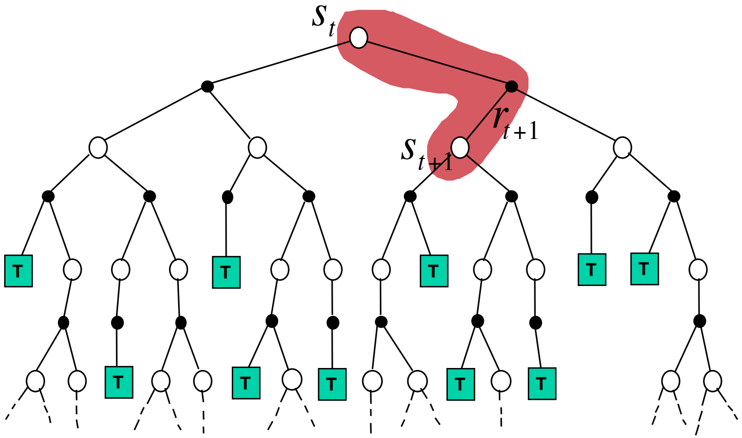
Monte-Carlo Backup

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



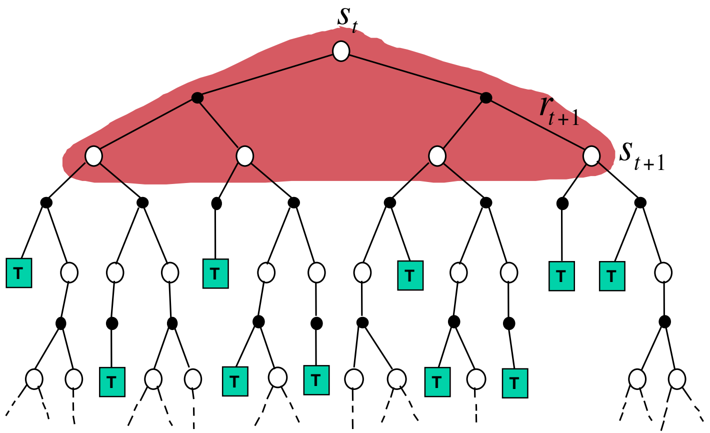
Temporal-Difference Backup

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



Dynamic Programming Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$



Bootstrapping and Sampling

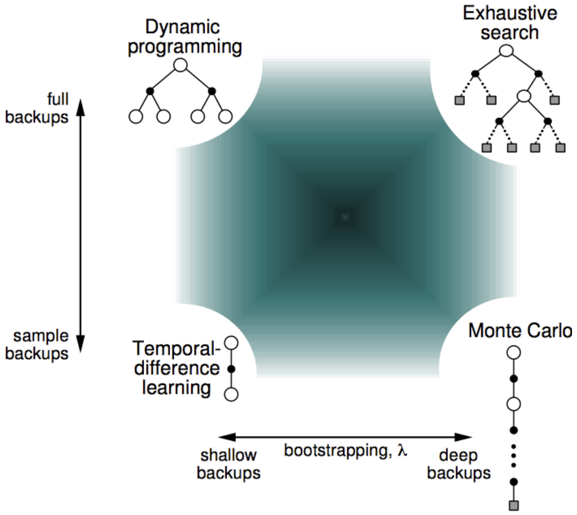
- Bootstrapping: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps

bootstrap is when you do not use the actual returns rather you use your own estimate of returns

- Sampling: update samples an expectation
  - MC samples
  - DP does not sample
  - TD samples

in DP you have to know the environment dynamics and you in turn perform and full width search over the next step

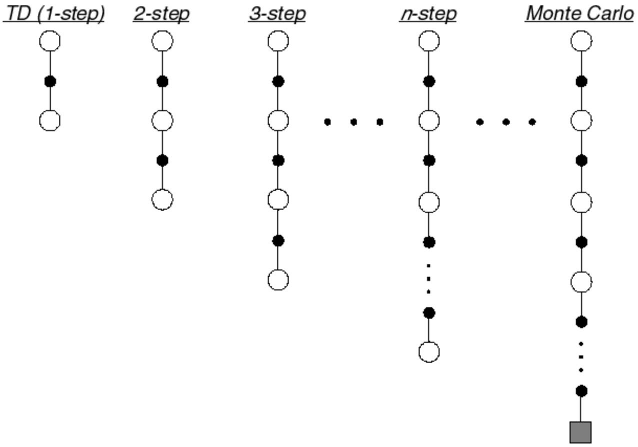
Unified View of Reinforcement Learning



n-Step Prediction

instead of looking one step ahead and update that value to our original state, we can look ahead n steps ahead and update our original state. As this n increases we tend to approach Monte Carlo

- Let TD target look n steps into the future



n-Step Return

- Consider the following n-step returns for $n = 1, 2, \infty$:

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

one step ahead reward

two step ahead discounted reward

twice discounted estimated return from that point onwards

in MC we looked into real rewards and we did not estimate at any point in time

- Define the n-step return

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

n steps of real reward by interacting with environment

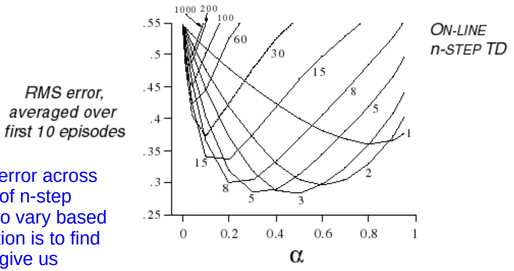
after n steps we are going to use our value function as a proxy for all the upcoming rewards

- n-step temporal-difference learning

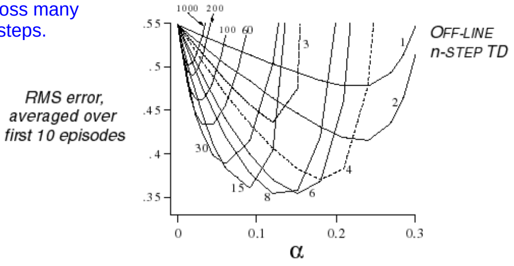
$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

here we start with an estimate V(St), we observe n values of that reward Gt(n) by interacting with the env plus our estimated value for the remainder, this generates an error signal, based on which we update our value function

Large Random Walk Example



here we observe the error across different alpha and # of n-step ahead. Results tend to vary based on on|off line. Motivation is to find an algorithm that will give us consistent results across many different ways and n-steps.



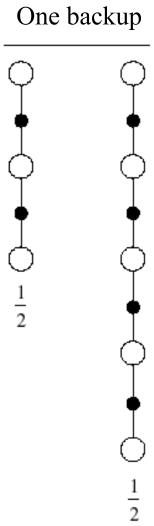
Averaging *n*-Step Returns

here we look ahead two steps and four steps and we adjust our TD target so as to reflect the average of the said look-aheads

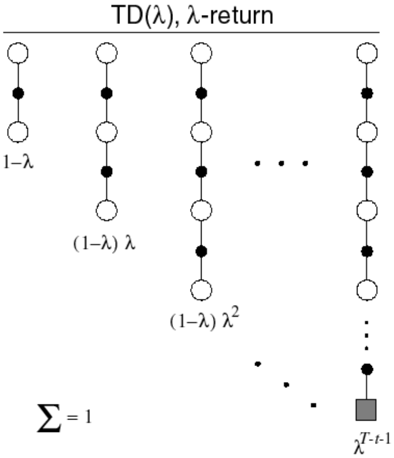
- We can average *n*-step returns over different *n*
- e.g. average the 2-step and 4-step returns

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

- Combines information from two different time-steps
- Can we efficiently combine information from all time-steps?



λ-return



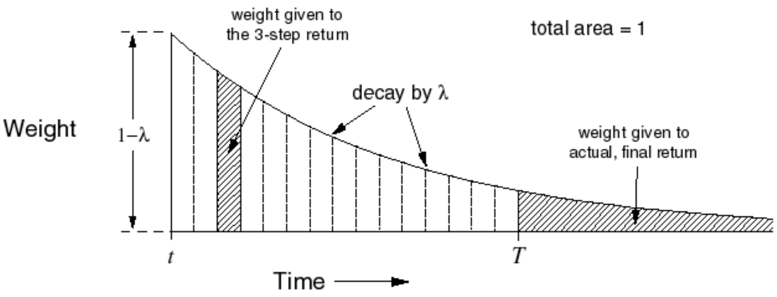
- The λ-*return* G_t^λ combines all n-step returns $G_t^{(n)}$
- Using weight $(1 - \lambda)\lambda^{n-1}$

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

- Forward-view TD(λ)

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

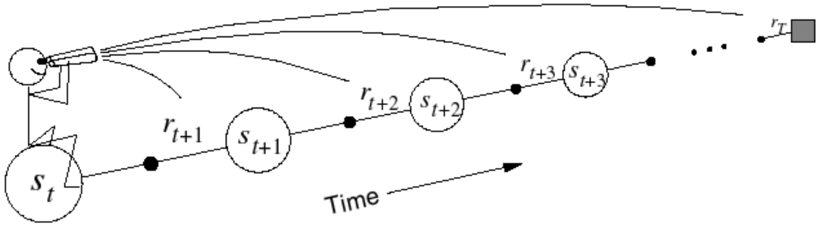
TD(λ) Weighting Function


weight given to the 3-step return
total area = 1
decay by λ
Weight
1−λ
weight given to actual, final return
t
T
Time

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

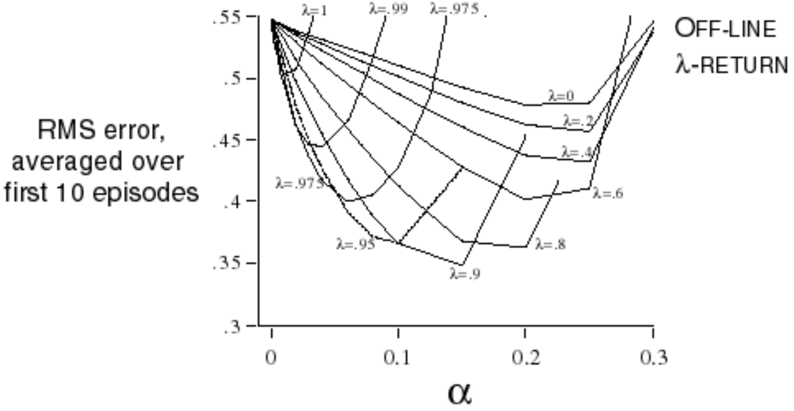
geometric decay of weights.
Geometric decay is memoryless, which means you can perform TD(0) and TD(λ) with the same effort

Forward-view TD(λ)



- Update value function towards the λ-return
- Forward-view looks into the future to compute G_t^λ
- Like MC, can only be computed from complete episodes

Forward-View TD(λ) on Large Random Walk

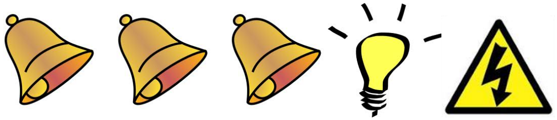


TD(λ) appears to be more robust on the changes of the characteristics

Backward View TD(λ)

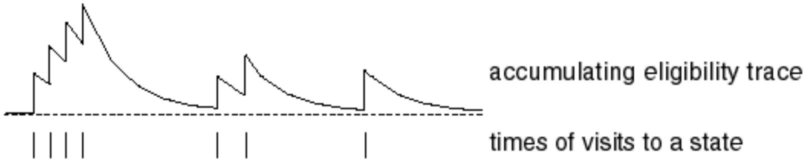
- Forward view provides theory
- Backward view provides mechanism
- Update online, every step, from incomplete sequences

Eligibility Traces



- Credit assignment problem: did bell or light cause shock?
- Frequency heuristic: assign credit to most frequent states
- Recency heuristic: assign credit to most recent states
- *Eligibility traces* combine both heuristics

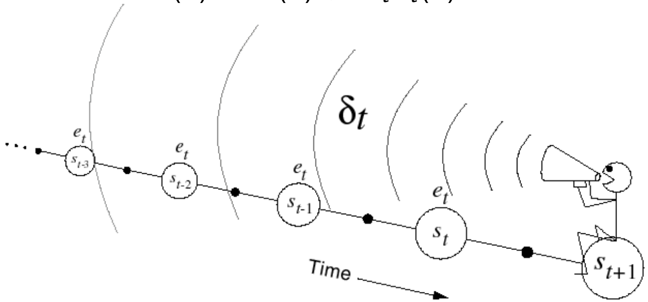
$$E_0(s) = 0$$
$$E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$$



Backward View TD(λ)

- Keep an eligibility trace for every state s
- Update value $V(s)$ for every state s
- In proportion to TD-error δ_t and eligibility trace $E_t(s)$

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



TD(λ) and TD(0)

- When $\lambda = 0$, only current state is updated

$$\begin{aligned} E_t(s) &= \mathbf{1}(S_t = s) \\ V(s) &\leftarrow V(s) + \alpha \delta_t E_t(s) \end{aligned}$$

- This is exactly equivalent to TD(0) update

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

TD(λ) and MC

- When $\lambda = 1$, credit is deferred until end of episode
- Consider episodic environments with offline updates
- Over the course of an episode, total update for TD(1) is the same as total update for MC

Theorem

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

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

MC and TD(1)

- Consider an episode where s is visited once at time-step k,
- TD(1) eligibility trace discounts time since visit,

$$\begin{aligned} E_t(s) &= \gamma E_{t-1}(s) + \mathbf{1}(S_t = s) \\ &= \begin{cases} 0 & \text{if } t < k \\ \gamma^{t-k} & \text{if } t \geq k \end{cases} \end{aligned}$$

- TD(1) updates accumulate error *online*

$$\sum_{t=1}^{T-1} \alpha \delta_t E_t(s) = \alpha \sum_{t=k}^{T-1} \gamma^{t-k} \delta_t = \alpha \left(G_k - V(S_k) \right)$$

- By end of episode it accumulates total error

$$\delta_k + \gamma \delta_{k+1} + \gamma^2 \delta_{k+2} + \ldots + \gamma^{T-1-k} \delta_{T-1}$$

Telescoping in TD(1)

When $\lambda = 1$, sum of TD errors telescopes into MC error,

$$\begin{aligned}
&\delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + ... + \gamma^{T-1-t}\delta_{T-1} \\
= &R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \\
+ &\gamma R_{t+2} + \gamma^2 V(S_{t+2}) - \gamma V(S_{t+1}) \\
+ &\gamma^2 R_{t+3} + \gamma^3 V(S_{t+3}) - \gamma^2 V(S_{t+2}) \\
&\quad \vdots \\
+ &\gamma^{T-1-t} R_T + \gamma^{T-t} V(S_T) - \gamma^{T-1-t} V(S_{T-1}) \\
= &R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} ... + \gamma^{T-1-t} R_T - V(S_t) \\
= &G_t - V(S_t)
\end{aligned}$$

TD(λ) and TD(1)

- TD(1) is roughly equivalent to every-visit Monte-Carlo
- Error is accumulated online, step-by-step
- If value function is only updated offline at end of episode
- Then total update is exactly the same as MC

Telescoping in TD(λ)

For general λ, TD errors also telescope to λ-error, $G_t^\lambda - V(S_t)$

$$\begin{aligned}
G_t^\lambda - V(S_t) = -V(S_t) \quad & + \quad (1-\lambda)\lambda^0 \left(R_{t+1} + \gamma V(S_{t+1})\right) \\
& + \quad (1-\lambda)\lambda^1 \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})\right) \\
& + \quad (1-\lambda)\lambda^2 \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3})\right) \\
& + \quad \ldots \\
= -V(S_t) \quad & + \quad (\gamma\lambda)^0 \left(R_{t+1} + \gamma V(S_{t+1}) - \gamma\lambda V(S_{t+1})\right) \\
& + \quad (\gamma\lambda)^1 \left(R_{t+2} + \gamma V(S_{t+2}) - \gamma\lambda V(S_{t+2})\right) \\
& + \quad (\gamma\lambda)^2 \left(R_{t+3} + \gamma V(S_{t+3}) - \gamma\lambda V(S_{t+3})\right) \\
& + \quad \ldots \\
= \quad & \quad\quad (\gamma\lambda)^0 \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right) \\
& + \quad (\gamma\lambda)^1 \left(R_{t+2} + \gamma V(S_{t+2}) - V(S_{t+1})\right) \\
& + \quad (\gamma\lambda)^2 \left(R_{t+3} + \gamma V(S_{t+3}) - V(S_{t+2})\right) \\
& + \quad \ldots \\
= \delta_t + \gamma\lambda\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \ldots &
\end{aligned}$$

Forwards and Backwards TD(λ)

- Consider an episode where s is visited once at time-step k,
- TD(λ) eligibility trace discounts time since visit,

$$\begin{aligned} E_t(s) &= \gamma\lambda E_{t-1}(s) + \mathbf{1}(S_t = s) \\ &= \begin{cases} 0 & \text{if } t < k \\ (\gamma\lambda)^{t-k} & \text{if } t \geq k \end{cases} \end{aligned}$$

- Backward TD(λ) updates accumulate error *online*

$$\sum_{t=1}^{T} \alpha\delta_t E_t(s) = \alpha \sum_{t=k}^{T} (\gamma\lambda)^{t-k}\delta_t = \alpha\left(G_k^\lambda - V(S_k)\right)$$

- By end of episode it accumulates total error for λ-return
- For multiple visits to s, $E_t(s)$ accumulates many errors

Offline Equivalence of Forward and Backward TD

Offline updates

- Updates are accumulated within episode
- but applied in batch at the end of episode

Onine Equivalence of Forward and Backward TD

Online updates

- TD(λ) updates are applied online at each step within episode
- Forward and backward-view TD(λ) are slightly different
- NEW: Exact online TD(λ) achieves perfect equivalence
- By using a slightly different form of eligibility trace
- Sutton and von Seijen, ICML 2014

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.