Lecture 4: Model-Free Prediction

CS60077 : REINFORCEMENT LEARNING

Autumn 2023

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

Predictions

What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?

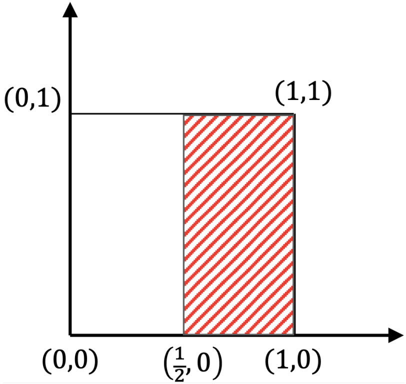


Figure: $\mathbb{P}(area) = \frac{1}{2}$

Predictions (2)

What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?

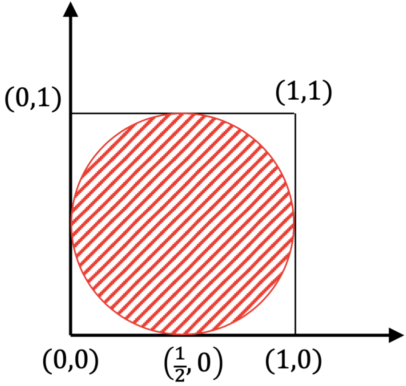


Figure: $\mathbb{P}(area) = \pi\left(\frac{1}{2}\right)^2$

Predictions (3)

What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?

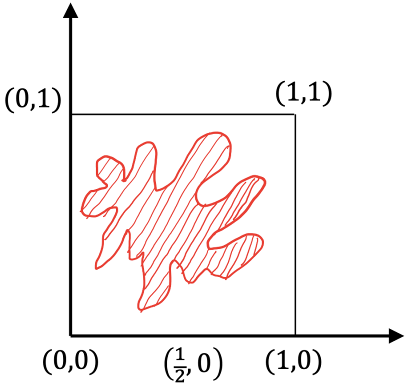


Figure: $\mathbb{P}(area) = ?$

Predictions and Monte-Carlo

What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?

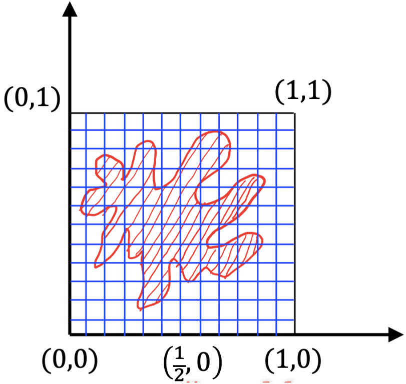


Figure: $\mathbb{P}(area) = \frac{\#\text{ red boxes}}{\#\text{ blue boxes}}$

Predictions and Monte-Carlo (2)

What is the probability that a dart thrown uniformly at random in the unit square will hit the red area?

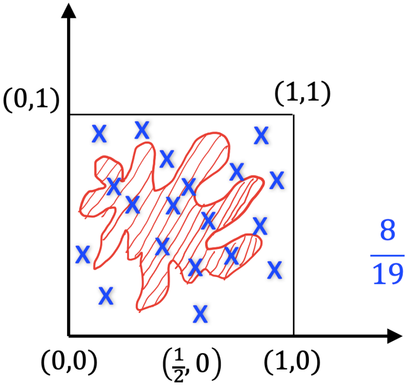


Figure: $\mathbb{P}(area) = \frac{\#\text{ darts in red area}}{\#\text{ total darts}}$

Monte-Carlo for Expectation Calculation

Compute: $\mathbb{E}\left[f(x)\right] = \int f(x)p(x)dx$

- Draw N i.i.d samples, $x_1, x_2, \ldots, x_N$ from probability density $p(x)$
- Approximate $p(x) \approx \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i}(x)$, where $\delta_{x_i}(x)$ is the impulse at x_i on x-axis

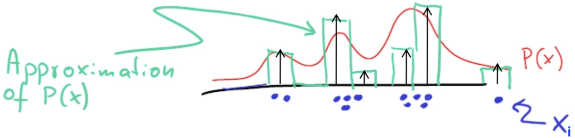


Image taken from: *Nando de Freitas: MLSS 08*

$$
\begin{aligned}
\mathbb{E}\left[f(x)\right] &= \int f(x)p(x)dx \approx \int f(x)\frac{1}{N}\sum_{i=1}^{N}\delta_{x_i}(x)dx \\
&= \frac{1}{N}\sum_{i=1}^{N}\underbrace{\int f(x)\delta_{x_i}(x)dx}_{f(x_i)} = \frac{1}{N}\sum_{i=1}^{N} f(x_i)
\end{aligned}
$$

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

Monte-Carlo Policy Evaluation

- 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:

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

- 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

First-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- The first time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- By law of large numbers, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

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 total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- Again, $V(s) \to v_\pi(s)$ as $N(s) \to \infty$

MRP Evaluation using Monte-Carlo: An Example

Estimation of values from 'experience' without knowing the model.

- Suppose, we have the following 5 samples/episodes.
- What is the estimated value of $V(S_1)$?
  - After 3 episodes?
  - After 4 episodes?

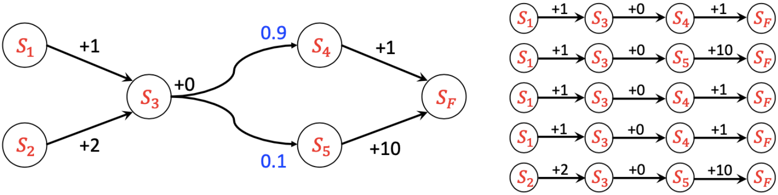


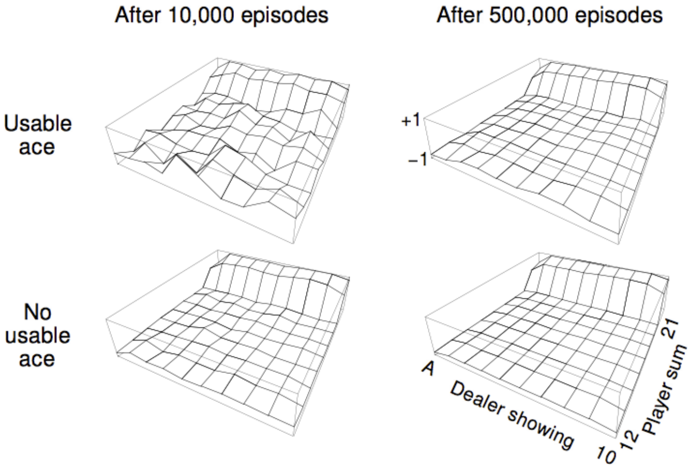
Figure: Samples (in right) drawn from the (unknown) Model (in left)

- After 3 episodes: $\frac{(1+0+1)+(1+0+10)+(1+0+1)}{3} = 5$
- After 4 episodes: $\frac{(1+0+1)+(1+0+10)+(1+0+1)+(1+0+1)}{4} = 4.25$

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

Incremental Mean

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

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

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.

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

Learning Rate and Incremental Monte-Carlo

- Monte-Carlo Update with incremental learning rate (α_T):

$$V_T(S_t) \leftarrow V_{T-1}(S_t) + \alpha_T\Big(G_t - V_{T-1}(S_t)\Big)$$

- If we visit a state (S_t) infinitely often (following policy π to draw samples), the value estimate for that state is going to converge to its true value, meaning $\lim_{T\to\infty} V(S_t) = V^\pi(S_t)$, provided the learning rate obeys the following two conditions:

(i) $\sum_{T=1}^{\infty} \alpha_T = \infty$ (ii) $\sum_{T=1}^{\infty} \alpha_T^2 < \infty$

- For example, if we take $\alpha_T = \frac{1}{T}$, then

$$\begin{aligned}\sum_{T=1}^{\infty} \alpha_t &= 1 + \frac{1}{2} + \left(\frac{1}{3}+\frac{1}{4}\right) + \left(\frac{1}{5}+\frac{1}{6}+\frac{1}{7}+\frac{1}{8}\right) + \frac{1}{9} + \cdots \\ &> 1 + \frac{1}{2} + \underbrace{\left(\frac{1}{4}+\frac{1}{4}\right)}_{\frac{1}{2}} + \underbrace{\left(\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}\right)}_{\frac{1}{2}} + \frac{1}{16} + \cdots \quad = \quad \infty\end{aligned}$$

Properties of Learning Rate

- A generalization of harmonic series is p-series (or hyperharmonic series), defined as $\sum_{n=1}^{\infty} \frac{1}{n^p}$ $(p \in \mathbb{R}^+)$
- Few choices of α_T for a converging algorithm are:

α_T	$\sum_{T=1}^{\infty} \alpha_T$	$\sum_{T=1}^{\infty} \alpha_T^2$	Algo Convergence
$\frac{1}{T^2}$	$< \infty$	$< \infty$	No
$\frac{1}{T}$	∞	$< \infty$	Yes
$\frac{1}{T^{\frac{2}{3}}}$	∞	$< \infty$	Yes
$\frac{1}{T^{\frac{1}{2}}}$	∞	∞	No

Temporal-Difference Learning

- 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

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 $R_{t+1} + \gamma V(S_{t+1})$

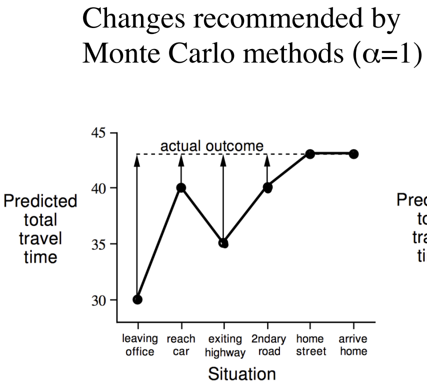
$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$

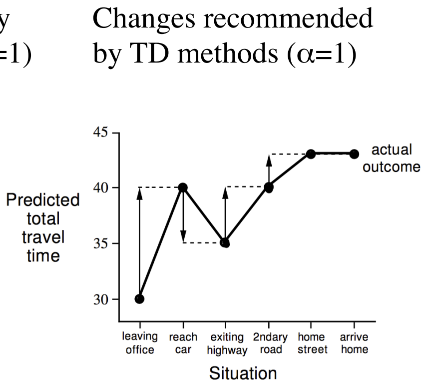
  - $R_{t+1} + \gamma V(S_{t+1})$ is called the *TD target*
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the *TD error*

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





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

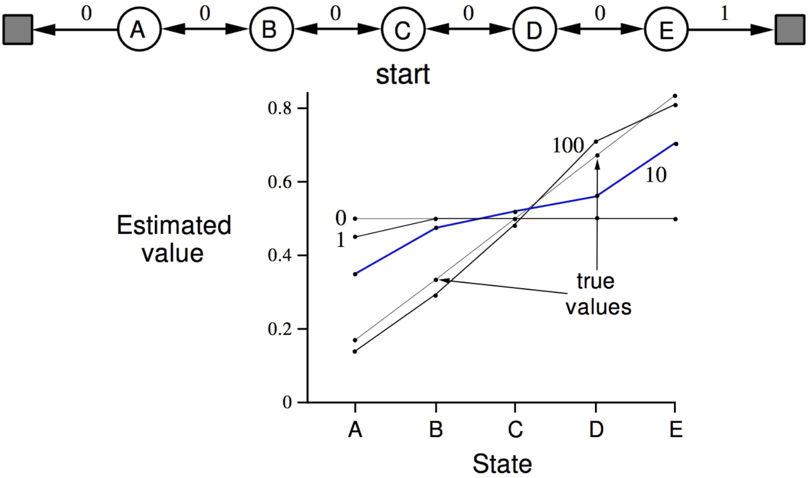
Bias/Variance Trade-Off

- Return $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$ is *unbiased* estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is *unbiased* estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimate of $v_\pi(S_t)$
- 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

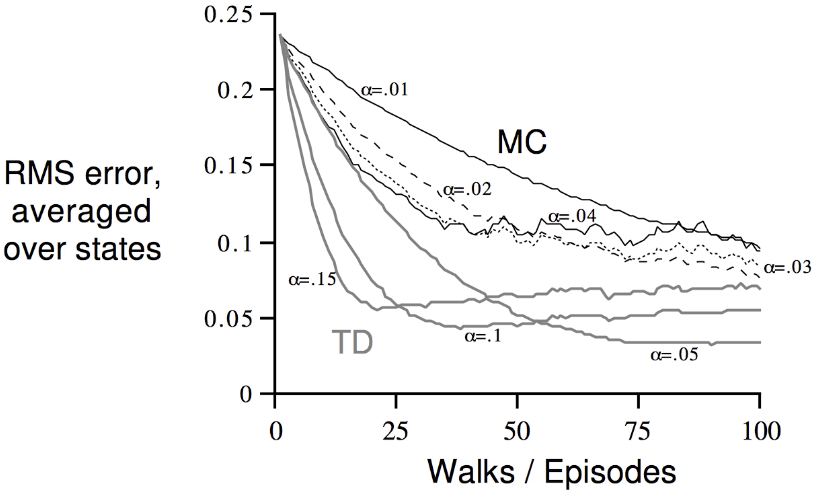
Advantages and Disadvantages of MC vs. TD (2)

- 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 Walk: MC vs. TD



Batch MC and TD

- MC and TD converge: $V(s) \rightarrow v_\pi(s)$ as experience $\rightarrow \infty$
- 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

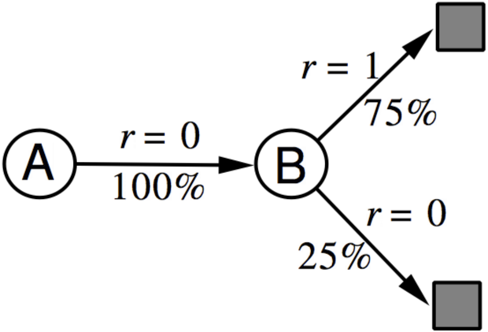
$\text{A}, 0, \text{B}, 0$
$\text{B}, 1$
$\text{B}, 1$
$\text{B}, 1$
$\text{B}, 1$
$\text{B}, 1$
$\text{B}, 1$
$\text{B}, 0$

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

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

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

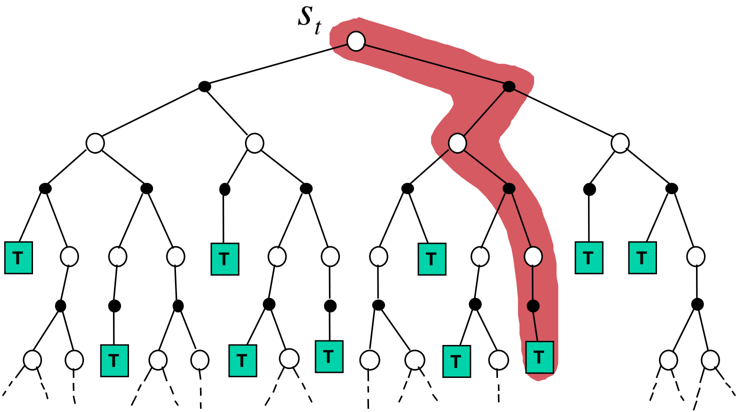
  - In the AB example, $V(A) = 0.75$

Advantages and Disadvantages of MC vs. TD (3)

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

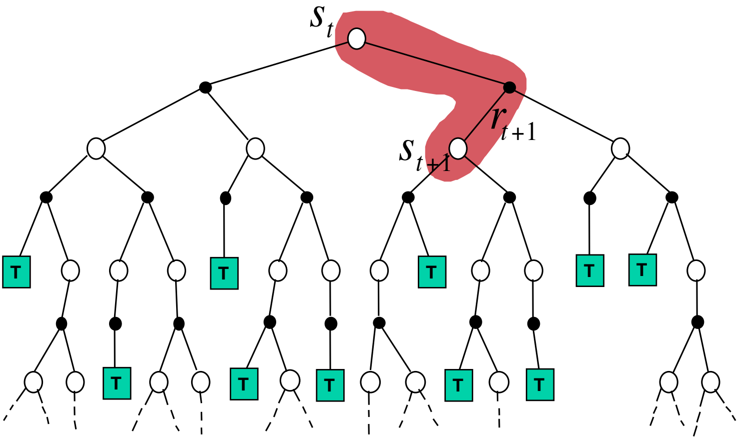
Monte-Carlo Backup

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



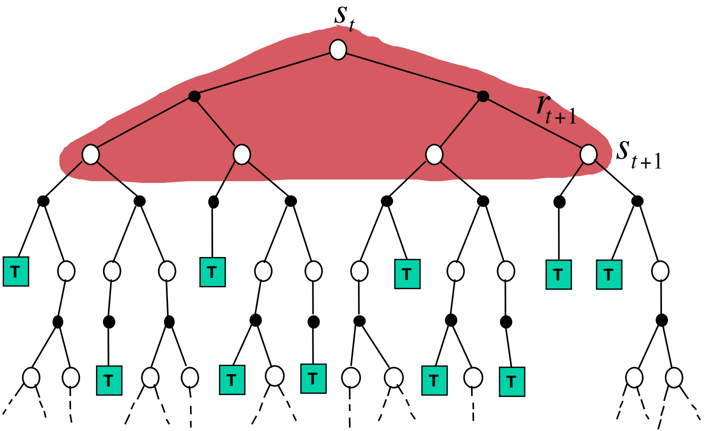
Temporal-Difference Backup

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



Dynamic Programming Backup

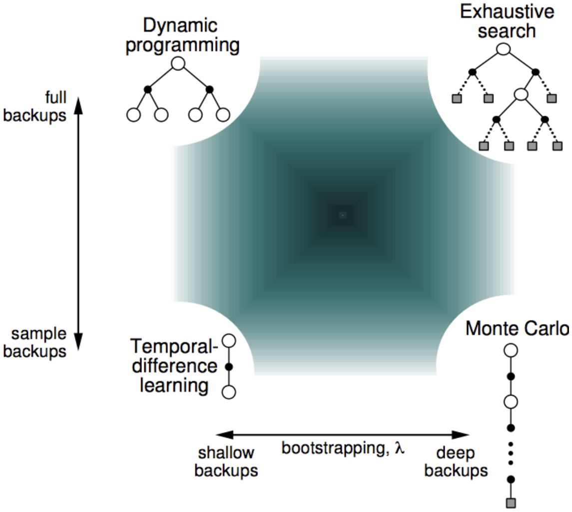
$$V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$



Bootstrapping and Sampling

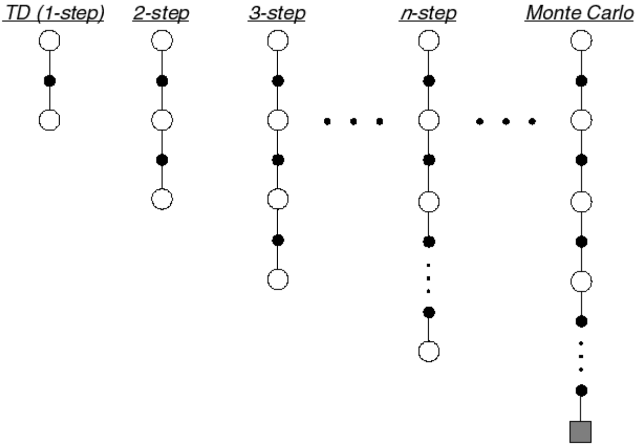
- Bootstrapping: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- Sampling: update samples an expectation
  - MC samples
  - DP does not sample
  - TD samples

Unified View of Reinforcement Learning



n-Step Prediction

- Let TD target look n steps into the future



n-Step Return

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

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

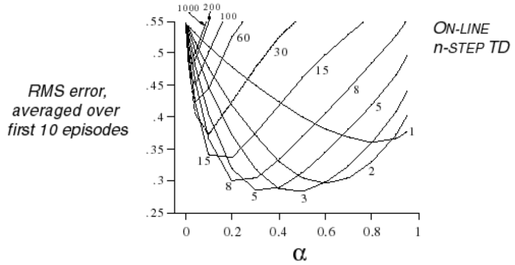
- Define the n-step return

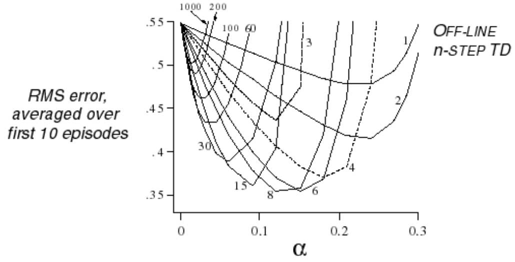
$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n-step temporal-difference learning

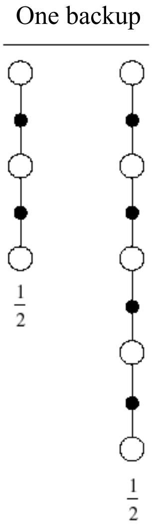
$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

Large Random Walk Example





Averaging n-Step Returns

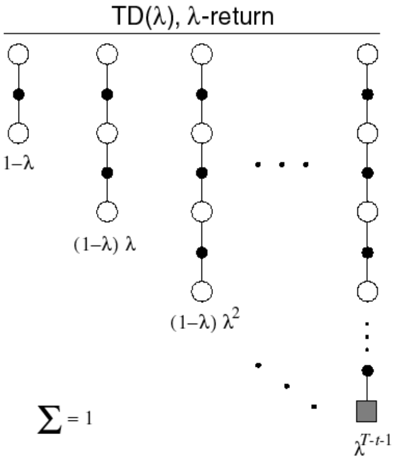


- We can average n-step returns over different n
- e.g. average the 2-step and 4-step returns

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

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

λ-return



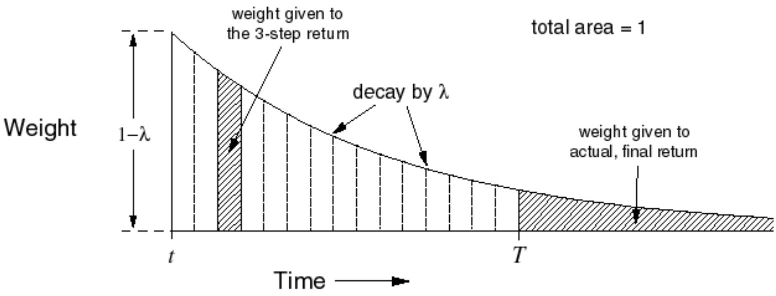
- The *λ-return* G_t^λ combines all *n*-step returns $G_t^{(n)}$
- Using weight $(1-\lambda)\lambda^{n-1}$

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

- Forward-view TD(λ)

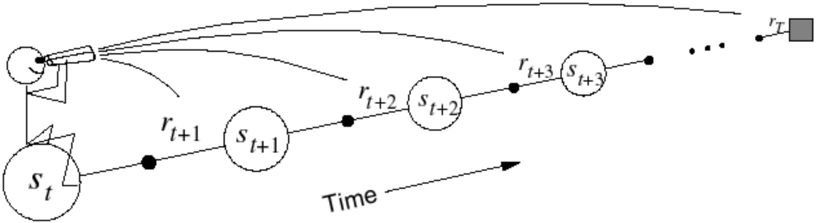
$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^\lambda - V(S_t)\right)$$

TD(λ) Weighting Function



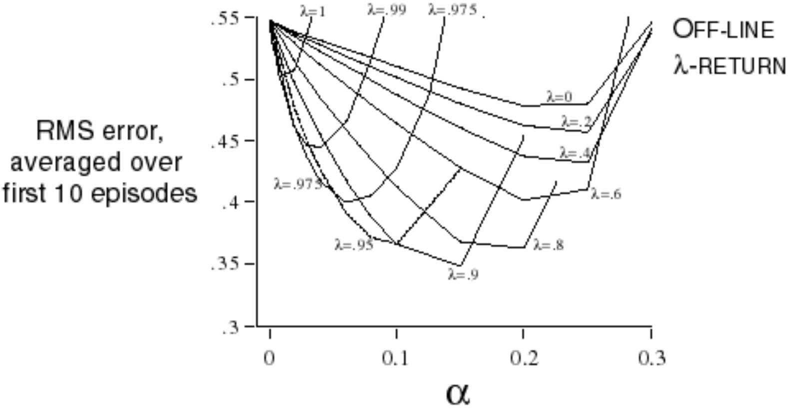
$$G_t^{\lambda} = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

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

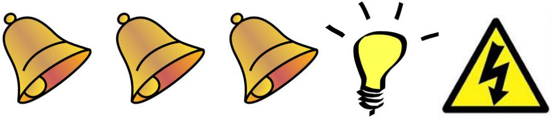
Forward-View TD(λ) on Large Random Walk



Backward View TD(λ)

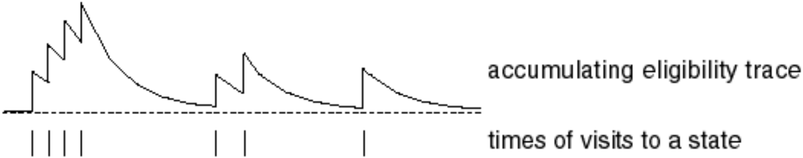
- Forward view provides theory
- Backward view provides mechanism
- Update online, every step, from incomplete sequences

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

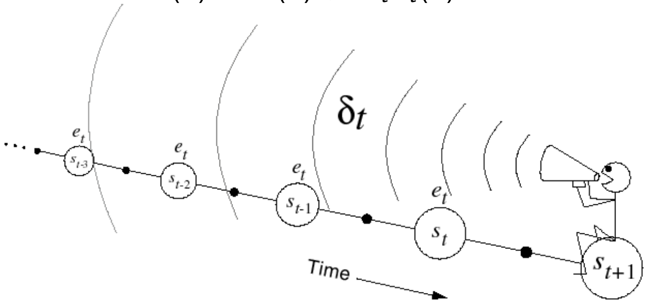
$$E_0(s) = 0$$
$$E_t(s) = \gamma \lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$$



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 $E_t(s)$

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



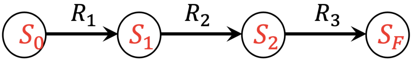
TD(1)

Algorithm 1: TD(1)

(Initialization) Episode Number $T \leftarrow 1$;
repeat
  foreach $s \in \mathcal{S}$ **do**
    initialize $E(s) \leftarrow 0$; `//` $E(s)$ `is 'eligibility' of state` s
    $V_T(s) \leftarrow V_{T-1}(s)$; `// same as previous episode`
  end
  $t \leftarrow 1$;
  repeat
    After state transition, $s_{t-1} \xrightarrow{R_t} s_t$,
      $E(s_{t-1}) = E(s_{t-1}) + 1$; `// update state eligibility`
    foreach $s \in \mathcal{S}$ **do**
      $V_T(s) \leftarrow V_{T-1}(s) + \alpha_T \Big(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}) \Big) E(s)$;
      $E(s) \leftarrow \gamma E(s)$;
    end
    $t \leftarrow t + 1$;
  until *this episode terminates*;
  $T \leftarrow T + 1$;
until *all episodes are done*;

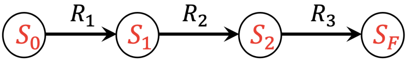
TD(1) Example

- Initial Eligibility Values:



0 0 0

- Change in Eligibility Values after $s_0 \rightarrow s_1$ Transition:



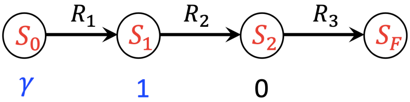
1 0 0

- Looping through all states and Applying TD update $\left(R_1 + \gamma V_{T-1}(s_1) - V_{T-1}(s_0)\right)$ proportional to eligibility and learning rate of all states

$$\begin{aligned} V_T(s_0) &= \alpha_T \left(R_1 + \gamma V_{T-1}(s_1) - V_{T-1}(s_0)\right), \\ V_T(s_1) &= 0, \qquad V_T(s_2) = 0 \end{aligned}$$

TD(1) Example (2)

- Change in Eligibility Values after $s_1 \rightarrow s_2$ Transition:

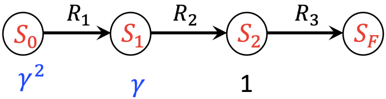


- Applying TD update of $\Big(R_2 + \gamma V_{T-1}(s_2) - V_{T-1}(s_1)\Big)$:

$$
\begin{aligned}
V_T(s_0) &= \alpha_T \Big(R_1 + \cancel{\gamma V_{T-1}(s_1)} - V_{T-1}(s_0)\Big) \\
&\quad + \gamma \alpha_T \Big(R_2 + \gamma V_{T-1}(s_2) - \cancel{V_{T-1}(s_1)}\Big) \\
&= \alpha_T \Big(R_1 + \gamma R_2 + \gamma^2 V_{T-1}(s_2) - V_{T-1}(s_0)\Big) \\
V_T(s_1) &= \alpha_T \Big(R_2 + \gamma V_{T-1}(s_2) - V_{T-1}(s_1)\Big), \\
V_T(s_2) &= 0
\end{aligned}
$$

TD(1) Example (3)

- Change in Eligibilty Values after $s_2 \to s_F$ Transition:



- Applying TD update of $\Big(R_3 + \gamma V_{T-1}(s_F) - V_{T-1}(s_2)\Big)$:

$$
\begin{aligned}
V_T(s_0) &= \alpha_T\Big(R_1 + \gamma R_2 + \cancel{\gamma^2 V_{T-1}(s_2)} - V_{T-1}(s_0)\Big) \\
&\quad + \gamma^2 \alpha_T \Big(R_3 + \gamma V_{T-1}(s_F) - \cancel{V_{T-1}(s_2)}\Big) \\
&= \alpha_T\Big(R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 V_{T-1}(s_F) - V_{T-1}(s_0)\Big) \\
V_T(s_1) &= \alpha_T\Big(R_2 + \cancel{\gamma V_{T-1}(s_2)} - V_{T-1}(s_1)\Big) + \gamma\alpha_T\Big(R_3 + \gamma V_{T-1}(s_F) - \cancel{V_{T-1}(s_2)}\Big) \\
&= \alpha_T\Big(R_2 + \gamma R_3 + \gamma^2 V_{T-1}(s_F) - V_{T-1}(s_1)\Big) \\
V_T(s_2) &= \alpha_T\Big(R_3 + \gamma V_{T-1}(s_F) - V_{T-1}(s_2)\Big)
\end{aligned}
$$

TD(0)

- TD(1) Update Rule:

$$V_T(s) \leftarrow V_{T-1}(s) + \alpha_T \Big(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}) \Big) E(s)$$

- TD(0) Update Rule:

$$V_T(s_{t-1}) \leftarrow V_{T-1}(s_{t-1}) + \alpha_T \Big(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}) \Big)$$

- What would we expect this outcome to be on average?
  - Here, from some state s_{t-1} and we make a random one-step transition to state s_t without knowing the destination.
  - So, ignoring α_T for the time being, the expected value of the above modified rule is $\mathbb{E}_{s_t}\left[R_t + V_T(s_t)\right]$, which is basically averaging after sampling different possible s_t values.
  - Similar to what maximum likelihood is also doing.

TD(0)

Algorithm 2: TD(0)

(Initialization) Episode Number $T \leftarrow 1$;
repeat
    foreach $s \in \mathcal{S}$ **do**
        initialize $E(s) \leftarrow 0$; `// ` $E(s)$ `is ‘eligibility’ of state` s
        $V_T(s) \leftarrow V_{T-1}(s)$; `// same as previous episode`
    end
    $t \leftarrow 1$;
    repeat
        After state transition, $s_{t-1} \xrightarrow{R_t} s_t$
        for $s = s_{t-1}$ **do**
            $V_T(s) \leftarrow V_{T-1}(s) + \alpha_T \Big(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}) \Big)$;
        end
        $t \leftarrow t + 1$;
    until *this episode terminates*;
    $T \leftarrow T + 1$;
until *all episodes are done*;

TD(λ)

Algorithm 3: TD(λ)

(Initialization) Episode Number $T \leftarrow 1$;
repeat
    foreach $s \in \mathcal{S}$ **do**
        initialize $E(s) \leftarrow 0$; `// E(s) is 'eligibility' of state s`
        $V_T(s) \leftarrow V_{T-1}(s)$; `// same as previous episode`
    end
    $t \leftarrow 1$;
    repeat
        After state transition, $s_{t-1} \xrightarrow{R_t} s_t$,
            $E(s_{t-1}) = E(s_{t-1}) + 1$; `// update state eligibility`
        foreach $s \in \mathcal{S}$ **do**
            $V_T(s) \leftarrow V_{T-1}(s) + \alpha_T \Big(R_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1})\Big) E(s)$;
            $E(s) \leftarrow \lambda \gamma E(s)$;
        end
        $t \leftarrow t + 1$;
    until *this episode terminates*;
    $T \leftarrow T + 1$;
until *all episodes are done*;

TD(λ) and TD(0)

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

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

- 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,

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

- 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

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

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, $G_t^\lambda - V(S_t)$

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

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.

Thank You!

Questions?

The only stupid question is the one you were afraid to ask but never did! – Rich Sutton