Reinforcement Learning
Lecture 3

Georg Martius
today substituted by René Geist

Distributed Intelligence / Autonomous Learning Group, Uni Tübingen, Germany

Oct 29, 2024

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Overview

- Last time: *known* MDP
  - Markov Decision Process (MDP)
  - reduction of MDP to Markov Reward Proc. (MRP) via policy
  - estimate the value function: policy evaluation
  - control: policy iteration and value iteration
- Today: *unknown* MDP
  - want to estimate the value function (prediction)
  - no model (model-free)
- Next time: Solve an *unknown* MDP

Model-free Prediction

Value function – Recap and Problem

Value function: Expected return when starting from state s and following policy π

$$v_\pi(s) = \mathbb{E}[G_t | S_t = s]$$

Why do we need the value function again? Because it is captures what we want to optimize! Ideal for optimizing behavior.

With known MDP:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

recursive definition!

Problem in unknown MDP? We have no $\mathcal{R}_s^a$ and $\mathcal{P}_{ss'}^a$!

Need to **estimate** v from experience!

Exploration and Exploitation

How to collect this experience?

- The agent should discover a good policy
- Needs to experience relevant transitions in the environment
- Should not lose too much reward along the way
- Explore around *promising* solutions
- Exploitation exploits known information to maximize reward
- Exploration finds more information about the environment
- In RL we the agent need to do both: explore as well as exploit

[Slide adapted from David Silver]

Examples

- Robot Control
  Exploitation Do the movement you know works best
  Exploration Try a different movement
- Playing Go
  Exploitation Play the move you believe is best
  Exploration Play an experimental move
- Control of plasma in tokamak
  Exploitation Play Create magnetic field that worked best so far
  Exploration Play Try a different magnetic field
- Restaurant Selection
  Exploitation Go to your favorite restaurant
  Exploration Try a new restaurant

[Slide adapted from David Silver]

Two Approaches

Monte Carlo Estimation

Follows the global definition:

$$v_\pi(s) = \mathbb{E}[G_t | S_t = s]$$

- learns from complete episodes of experience
- Simplest idea: value = mean return

Temporal Difference Learning

Follows the recursive definition:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

- learns from incomplete episodes, by bootstrapping
- updates a guess towards a guess

Both learn **directly** from **experience**.

Monte-Carlo Value Estimation (Policy Evaluation)

Goal: learn v_π from episodes of experience under policy π

- Given: episodes of experience: all episodes $\mathcal{T}_\pi$, one episode $\tau \in \mathcal{T}_\pi$:

$$\tau = S_1, A_1, R_2, S_2, A_2, R_3 \ldots, S_T \sim \pi$$

- Return: total discounted reward:

$$G_t(\tau) = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T$$

- Value function is the expected return:

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

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

Notation: S_t, A_t, R_t are here specific states/actions/rewards at time t.
For consistency with book.

Monte-Carlo Value Estimation (Policy Evaluation)

Need to compute empirical mean return:

- Sum of all trajectories
- Compute return for state s and divide by the number of times s was visited $N(s)$

$$V(s) = \frac{1}{N(s)} \sum_{\tau \in \mathcal{T}_\pi} G_t(\tau) [\![S_t = s]\!]$$

$V(s) \to v_\pi(s)$ as $N(s) \to \infty$.

MC Value Estimation

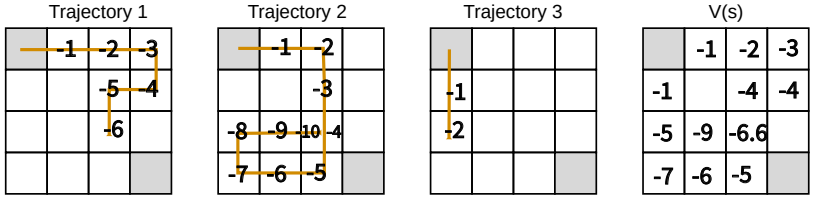
$N(s) = 0$ and $W(s) = 0$, $\forall s \in \mathcal{S}$
For all τ and for t in episode τ:

1. Increment counter $N(S_t) \leftarrow N(S_t) + 1$
2. Increment total return $W(s_t) \leftarrow W(S_t) + G_t$
3. Value estimate is updated to mean return $V(S_t) = W(S_t)/N(S_t)$

Example: Gridworld

$V(s) = W(s)/N(s)$

Assume the following trajectories and the corresponding returns G ($\gamma = 1$).



Observe how the values for states that are multiple times visited become averages

Example: Blackjack

- Goal: get as close as possible to 21 points but not above
- Counting: Face-cards: 10, Ace: 1 or 11, other: cards their value
- start with two cards, dealer has one card open

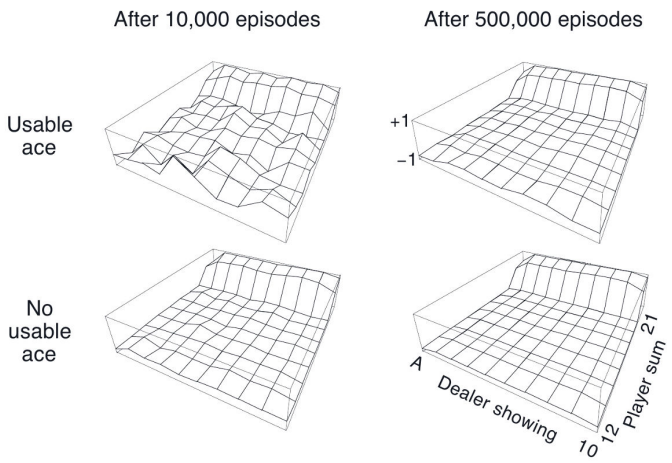
Example: Blackjack

Here one player against dealer.

- Two actions:
  1. stick: Stop receiving cards and terminate
  2. twist: Take another card (without replacement)
- Rewards 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
- Rewards for twist:
  - -1 if sum of cards > 21 (*goes bust* and terminate)
  - 0 otherwise
- Transitions: automatically twist if sum < 12
- States (200 of them)
  - Sum of cards (12 – 21)
  - Dealer's shown card (ace – 10)
  - Useable Ace? (player has an ace that can be counted as 11)

MC - Value estimation for Blackjack

- Policy: stick if sum of cards ≥ 20, otherwise twist
- Dealer: stick if sum of cards ≥ 17, otherwise twist



Why Monte-Carlo and not Dynamic programming Policy evaluation?

- System is known, so we could do DP

  Discuss for 3 minutes with your neighbor
- Computing the probabilities $\mathcal{P}$ is difficult and error prone
- Simulating the environment can be easy (for games it usually is easy)

So MC can sometimes be prefered also in known systems

Incremental Update of Mean

Remember: MC Value Estimation

$N(s) = 0$ and $W(s) = 0$, $\forall s \in \mathcal{S}$
For all τ and for t in episode τ:

1. Increment counter $N(S_t) \leftarrow N(S_t) + 1$
2. Increment total return $W(s_t) \leftarrow W(S_t) + G_t$
3. Value estimate is updated to mean return $V(S_t) = W(S_t)/N(S_t)$

Let's do this in a way that we can feed one trajectory τ at a time and update $V(s)$ incrementally, without storing W.

Talk to your peers and compute an incremental update for a mean μ_k from a sequence $x_1, x_2, \ldots$. Derive how to update μ_k from μ_{k-1} and x_k:

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

Incremental Update of Mean

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

Incremental Monte-Carlo Updates

Incremental every-visit MC policy evaluation

- For every episode τ
  1. For every t
  2. $N(S_t) \leftarrow N(S_t) + 1$
  3. $V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$

In non-stationary problems: use a running mean, i.e. forget old episodes:
$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$

Temporal-Difference Learning

Goal: learn v_π from episodes of experience under policy π

- Idea: make value function locally consistent (minimize temporal difference)
- Given: episodes of experience $\mathcal{T}_\pi$
- Update value $V(S_t)$ towards *estimated* return: $R_{t+1} + \gamma V(S_{t+1})$
- Value function is the expected return: Simplest algorithm: TD(0)

$$V(S_t) \leftarrow V(S_t) + \alpha\Big(\overbrace{\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{TD target}} - V(S_t)}^{\text{TD error: } \delta_t}\Big)$$

Compare: Incremental every-visit Monte-Carlo:

- Update value $V(S_t)$ towards *actual* return G_t

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

MC vs. TD

α-MC: $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$
TD(0): $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

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

What happens if we have finite experience?

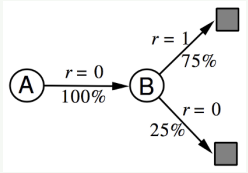
$$\tau^1, \tau^2, \dots \tau^K$$

Batch methods: repeatedly go through the data (sample $k \in [1, K]$)

AB Example

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

episode	transitions
1	A, 0; B, 0
2	B, 1
3	B, 1
4	B, 1
5	B, 1
6	B, 1
7	B, 1
8	B, 0



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

MC: $V(A) = 0 \quad V(B) = \frac{3}{4}$

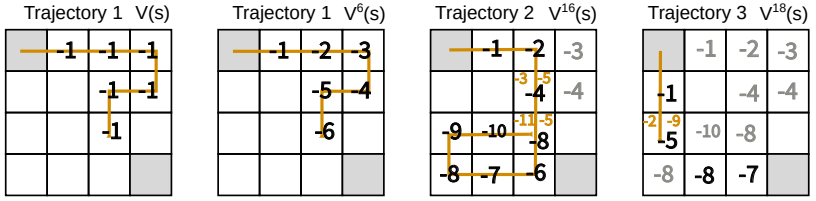
TD: $V(A) = \frac{3}{4} \quad V(B) = \frac{3}{4}$

Example: Gridworld

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

Assume the following **trajectories** as before.

- we update V offline a few times before adding a new trajectory (all data)
- $\gamma = 1$
- Computation not quite accurate, not α, but average over different paths:



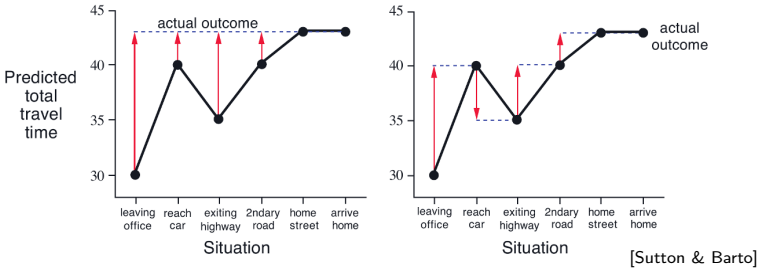
- observe how TD uses intermediate values
- What would happen with only first 2 trajectories and iterated updates?

Example: Driving Home

α-MC: $V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$
TD(0): $V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$

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

updates of Monte Carlo method ($\alpha = 1$) updates by TD methods ($\alpha = 1$)



[Sutton & Barto]

Bias/Variance Trade-Off

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

MC vs. TD

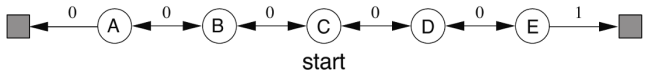
TD

- can update at every step
  - works also in non-episodic case
- low variance, some bias
- TD(0) converges to $v_\pi(s)$ (not always with function approximation)
- more sensitive to initial value
- usually more efficient than MC

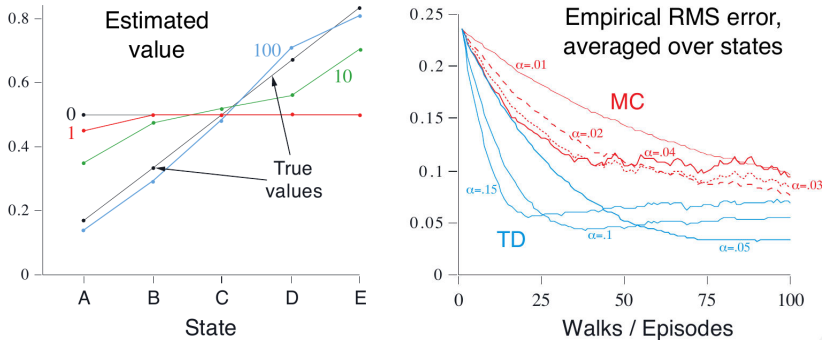
MC

- only updates after episode ends
  - only work in episodic case
- high variance, zero bias
- good convergence properties (also with function approximation)
- not very sensitive to initial value
- very simple to understand and use

Random Walk Example



One run of TD(0) ($\alpha = 0.1$) Learning curves



Initialization: $V(s) = 0.5$

What do Batch MC and TD converge to?

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

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

- 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

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

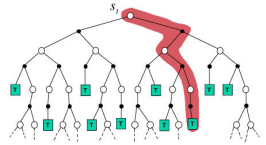
$$\hat{\mathcal{R}}_{s}^{a} = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}\left(s_t^k, a_t^k = s, a\right) r_t^k$$

TD exploits Markov property
- more efficient in Markov environments

MC does not exploit Markov property
- more effective in non-Markov environments

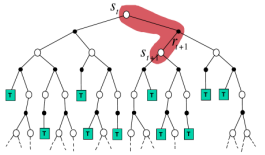
Backup Diagrams

Monto Carlo



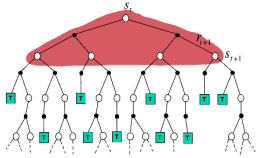
$V(S_t) \leftarrow$
$V(S_t) + \alpha\left(G_t - V(S_t)\right)$

Temporal Difference



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

Dynamic Programming



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

Bootstrapping: update uses estimates

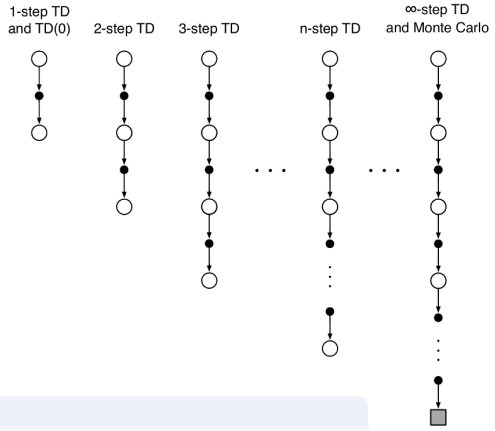
No	Yes	Yes

Sampling: update "samples" an expectation

Yes	Yes	No

n-step Backup

TD with n steps look into the future



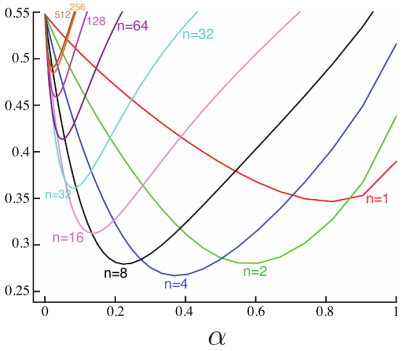
n-step return

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

n-step TD

Random walk (with 19 states)

Average RMS error over 19 states and first 10 episodes



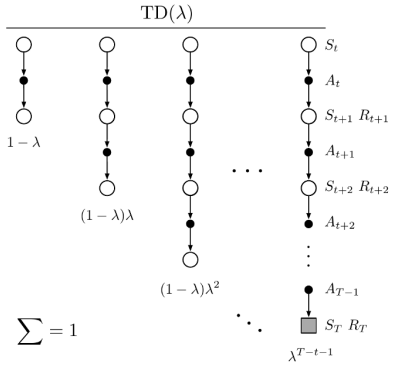
n-step TD methods (with data from 10 episodes)

intermediate value of n can be better than either extremes (TD(0) and MC)

What about combining n steps in TD? $\rightarrow$ TD(λ)

Idea: combine information from all n-step updates

How could we do that?



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

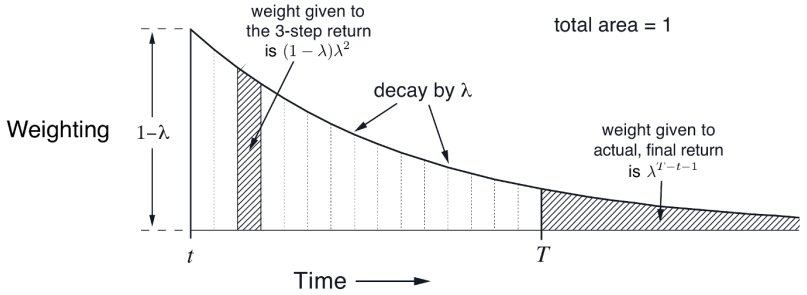
$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$$

- Forward-view TD(λ)

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

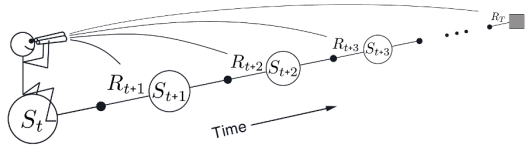
n-step return: $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1}R_{t+n} + \gamma^n V(S_{t+n})$

TD(λ) Weighting



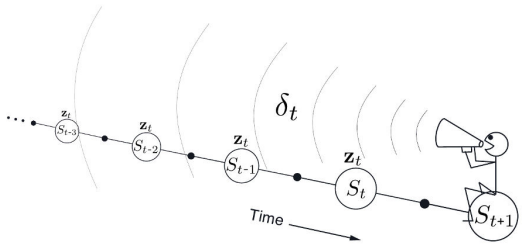
$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$$

Forward-view and Backward View of TD(λ)



- Forward-view: looks into the future to compute G_t^λ
- Can only be computed at the end of the episode (like MC)

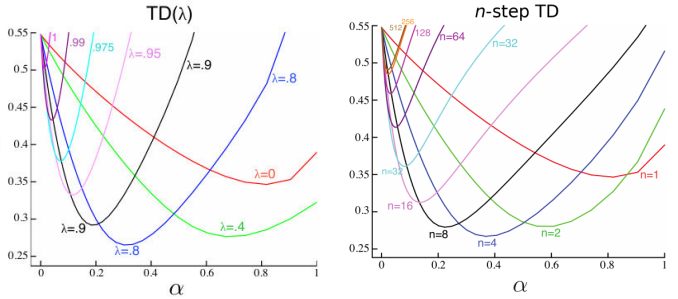
There is an effiient way to implement the Backward view using Eligibility Traces (later)



TD(λ)

Example: random walk with 19 states

RMS error at the end of the episode over the first 10 episodes



Intermediate values of λ are best (as before with n-step)

TD(λ) and TD(0)

Reminder: TD(λ), λ-returns, n-step returns:
$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^\lambda - V(S_t)\right)$
$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$
$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^n V(S_{t+n})$

For $\lambda = 0$: only current state is updated:

$$\begin{aligned} G_t^0 &= (1-0)0^0 G_t^{(1)} \\ &= R_{t+1} + \gamma^1 V(S_{t+1}) \\ V(S_t) &\leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_{t+1})) \end{aligned}$$

Equivalent to TD(0):

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

TD(λ) and MC

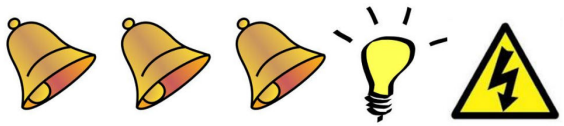
For $\lambda \to 1$ the credit is deferred to the end of the episode

- We 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

Backward View TD(λ)

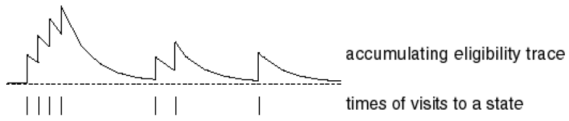
- Forward view is impractical but provides the theory
- Backward view provides mechanism/practical implementation
  - update online from incomplete sequences
- However, most modern algorithms are updating values offline and keep a memory of past interactions
- Backward view is less important anymore, but good to understand nevertheless

Backward View – 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

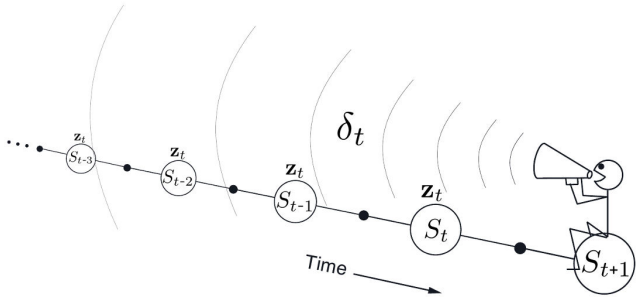
$$\begin{aligned} E_0(s) &= 0 \\ E_t(s) &= \gamma\lambda E_{t-1}(s) + [\![S_t = s]\!] \end{aligned}$$



Backward View – Eligibility Traces

- Keep an eligibility trace for every state s
  (how much did s contribute to the current situation)
- Update value $V(s)$ for every state s
- current TD-error δ_t scaled by eligibility $E_t(s)$

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



TD(λ) and MC

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

Forward-backward Equivalence

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

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

- Online updates:
  - TD(λ) updates are applied online at each step within episode
  - Forward and backward-view TD(λ) are slightly different

TD(1) and MC

Advanced content for self study

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

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

- TD(1) updates accumulate error online

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

- By end of episode it accumulates total error

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

TD(1) and MC – Online

Advanced content for self study

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

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

TD(λ) and MC

Advanced content for self study

- TD(1) is roughly equivalent to Monte-Carlo (MC)
- 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

TD(λ): Telescoping sums trick can be also used here

Forward and backward view of TD(λ) are the same: yield total λ-return

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

- Backward TD(λ) updates online
- For multiple visits of s, $E_t(s)$ accumulates many TD-errors

Online and Offline updates

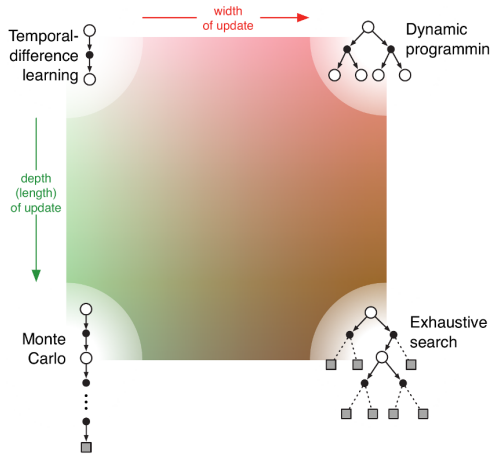
Advanced content for self study

- Offline updates:
  - Updates are accumulated within episode but applied in batch at the end of episode
- Online updates:
  - TD(λ) updates are applied online at each step within episode
  - Forward and backward-view TD(λ) are slightly different

Offline updates	$\lambda = 0$	$\lambda \in (0, 1)$	$\lambda = 1$
Backward view	$\mathrm{TD}(0)$	$\mathrm{TD}(\lambda)$	$\mathrm{TD}(1)$
	$\Vert$	$\Vert$	$\Vert$
Forward view	$\mathrm{TD}(0)$	Forward $\mathrm{TD}(\lambda)$	MC
Online updates	$\lambda = 0$	$\lambda \in (0, 1)$	$\lambda = 1$
Backward view	$\mathrm{TD}(0)$	$\mathrm{TD}(\lambda)$	$\mathrm{TD}(1)$
	$\Vert$	$\not\Vert$	$\not\Vert$
Forward view	$\mathrm{TD}(0)$	Forward $\mathrm{TD}(\lambda)$	MC
	$\Vert$	$\Vert$	$\Vert$
Exact Online	$\mathrm{TD}(0)$	Exact Online $\mathrm{TD}(\lambda)$	Exact Online $\mathrm{TD}(1)$

Equality in terms of total update at end of episode

Overview of TD, MC, DP and Exhaustive search



Summary

- Value estimation is central in Reinforcement Learning
- It is a non-trivial problem as we have to fight stochasticity
- Two different ways:
  - Monte Carlo (MC)
    - high variance, zero bias
    - simple to understand and use
  - Temporal Difference (TD)
    - low variance, some bias
    - usually more efficient than MC
- combination of both: n-step returns
- TD(λ): weighted version of all n-step returns