Lecture 4: Model-Free Prediction

Tuesday, August 9, 2022 12:10 PM

Lecture 3: Planning w/ Dynamic Programming for a KNOWN MDP

This lecture: Model-Free prediction (w/ unknown MDP)
- estimate value function of unknown MDP

Next lecture: Model-Free control
- Optimise value function of unknown MDP

Monte-Carlo Learning

- Learn directly from episodes of experience
- don't need knowledge of MDP transitions
- only works for episodic MDPs (all episodes must terminate)

Goal: learn v_π from episodes of experience under policy π

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

- Monte-Carlo evaluation uses empirical mean return instead of expected value

- Version 1 of Monte Carlo learning

First-Visit Monte-Carlo Policy Evaluation

- For a state s on the first time-step t that s is visited
  - Increment counter $N(s) \leftarrow N(s) + 1$
  - Increment total return $S(s) \leftarrow S(s) + G_t$

$N(s) \rightarrow$ number of episodes in which ... visited we have visited s

Note that we are adding G_t to $S(s)$, G_t is TOTAL return from the time we saw the state

- Value is estimated by mean return $V(s) = \frac{S(s)}{N(s)}$
- Law of Large Numbers
  - ↳ $V(s) \to v_\pi(s)$ as $N(s) \to \infty$

Version 2 of Monte Carlo learning

Every - visit Monte-Carlo Evaluation

- Exact same algo as first-visit, except don't just update on first-visit, update every time you see a specific state

Incremental Mean Formula

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

$$= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right)$$

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

previous mean

$$= \frac{1}{k}x_k + \frac{1}{\cancel{k}}\cancel{k}\mu_{k-1} - \frac{1}{k}\mu_{k-1}$$

$$\mu_k = \mu_{k-1} + \frac{1}{k}\underbrace{\left(x_k - \mu_{k-1}\right)}_{\text{error term}}$$

} move mean in direction of error

Incremental Monte-Carlo Updates

- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, \dots S_T$
- For each state S_t w/ return G_t

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

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

- For non-stationary problems, it can be helpful to track a running mean i.e; forget old episodes

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

TD-Learning

Temporal-Difference:
- Learn directly from episodes of experience
- model free: no knowledge of MDP transitions/rewards
- learns from incomplete episodes, by bootstrapping
  - ↳ updates guess before a guess

Goal: learn V_π online from experience under policy π

Incremental, every-visit Monte-Carlo

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

Simplest T-D algorithm TD(0)

Update value $V(S_t)$ toward estimated return

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

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

$R_{t+1} + \gamma V(S_{t+1})$ is TD-target

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

- Estimates are grounded by end-of-episode

Advantages and Disadvantages

- TD can learn before knowing final outcome
  MC must wait till end of episode

- TD can learn w/o final outcome

Bias/Variance Trade-Off

- Return $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$ is unbiased estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is unbiased estimate of $v_\pi(S_t)$
  oracle value (true value function)

- TD target $R_{t+1} + \gamma V(S_{t+1})$ is biased estimate of $v_\pi(S_t)$
  - has much less variance than the return
  - Return depends on many random actions, transitions, rewards
  - TD target depends on one random action, transition, reward

MC

- high variance, zero bias
- good convergence
- not sensitive to initial value
- very simple

TD

- Usually more efficient than MC
- TD(0) converges to $v_\pi(s)$
  - (but not always w/ function approximation)
- more sensitive to initial value

- MC and TD converge: $V(s) \to V_\pi(s)$ as experience $\to \infty$
- But what about batch solutions
  - $s_1^1, a_1^1, r_2^1, \ldots s_{T_1}^1$ ← episode 1

    ⋮
  - $s_1^K, a_1^K, r_2^K \ldots s_{T_K}^K$ ← episode K

- MC converges to solution w/ minimum mean-squared error
  - Best fit to observed returns

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

- TD(0) converges to solution of max-likelihood Markov model

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

Recap

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

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

$$V(S_t) \leftarrow E_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$ Dynamic Programming (Need to know environment dynamics)

Categorization

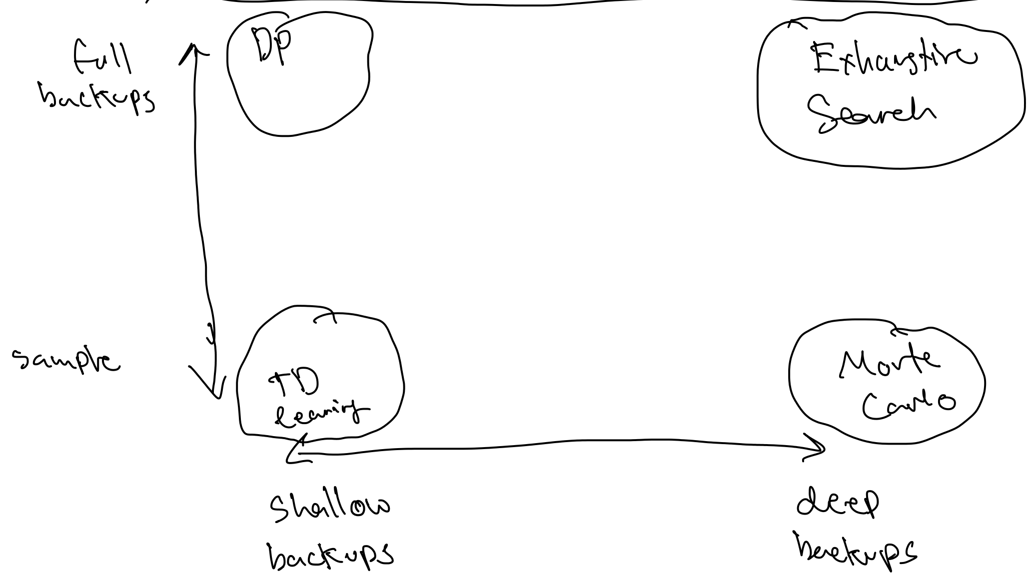
Bootstrap: update involves an estimate

- MC does not bootstrap
- DP bootstraps
- TD bootstraps

Sampling: update samples an expectation

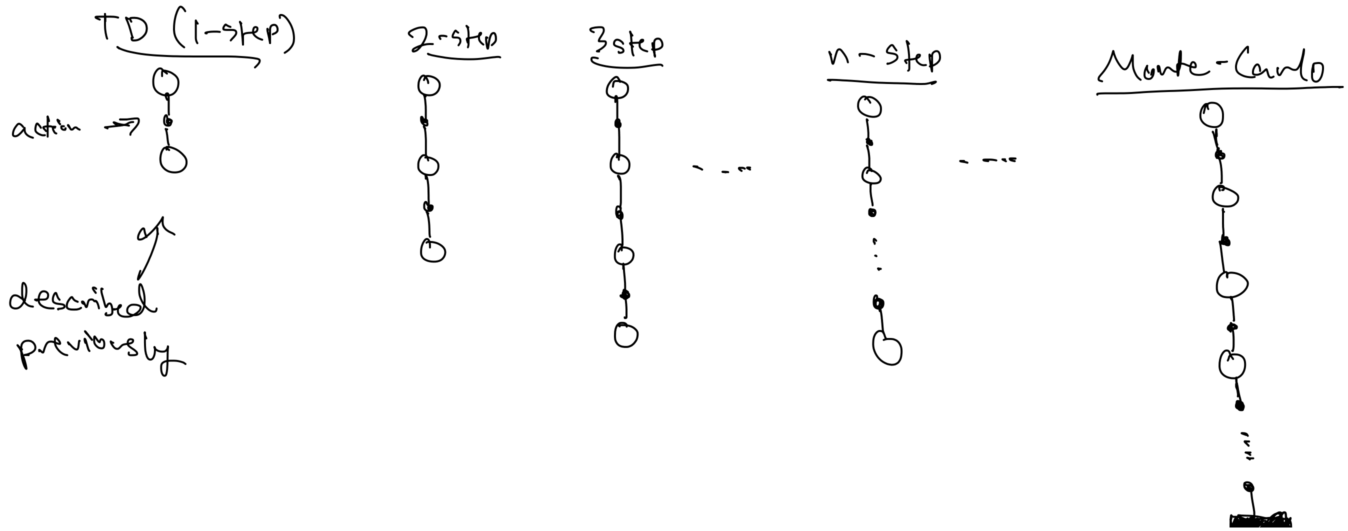
- MC samples
- DP does not sample
- TD samples

Unified view of Policy Evaluation



TD(λ) - mix of TD-learning and Monte-Carlo updates

- TD target looks n steps into the future



[redacted]

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

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

$\vdots$

$n=\infty \qquad G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$

n-step return

$$G_t^{(n)} = \underbrace{R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n}}_{\text{real}} + \underbrace{\gamma^n V(S_{t+n})}_{\text{estimate}}$$

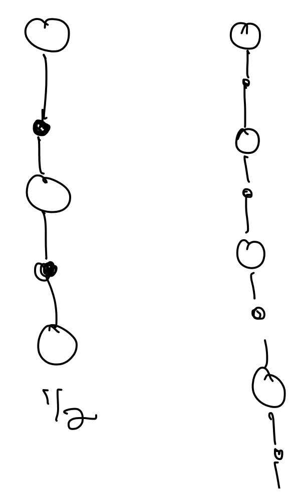
n-step temporal-difference learning

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

Averaging n-Step returns

- average n-step returns over different n
- e.g. average 2-step and 4-step return $\longrightarrow$

One backup



TD (λ), λ-Return

- λ-Return G_t^{λ} combines all n-step ... $G_t^{(n)}$

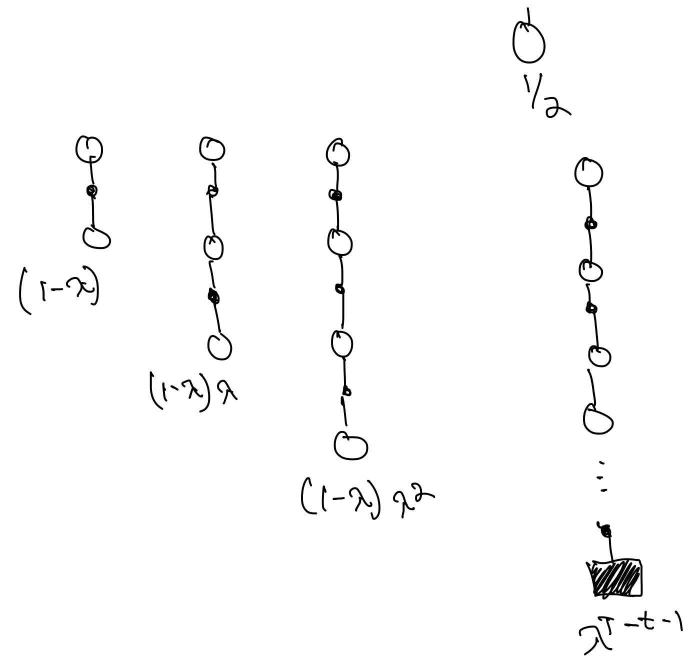
n-step returns $G_t^{(n)}$

- Using weight $(1-\lambda)\lambda^{n-1}$

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

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

☆ initial weight of $(1-\lambda)$ is used to ensure weights sum to one



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

- Forward-view looks into future to compute G_t^{λ}
- Like MC, can only be computed from complete episodes

BUT !!!

- Backward View TD(λ)
  - Forward view provides theory
  - Backward provides mechanism
  - Update online, every step, from incomplete sequences

Credit assignment Problem

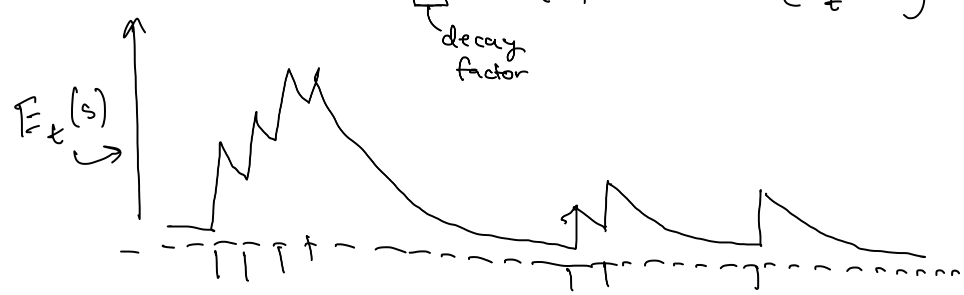
Frequency Heuristic: assign credit to most frequent states

Recency heuristic: assign credit to most recent states

Eligibility Trace

$$E_0(s) = 0$$

$$E_t(s) = \underbrace{\gamma}_{\text{decay factor}} \lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$$



accumulating eligibility trace

times of visit to state

- Keep eligibility trace for every state s
- Update value $V(s)$ for every state S in proportion to TD-Error and eligibity trace

TD-Error $\delta_t = (R_{t+1} + \gamma V(S_{t+1})) - V(S_t)$

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$

α = learning_rate

when $\lambda = 0$, only current state is updated

$$E_t(s) = \mathbf{1}(S_t = s)$$

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$

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

when $\lambda = 1$, credit is deferred until end of episode
- same updates at Monte Carlo

Theorem: Sum of offline updates is identical for forward and backward view TD(λ)

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

Summary of Forward and Backward TD(λ)

	$\lambda = 0$	$\lambda \in (0,1)$	$\lambda = 1$
Forward View	TD(0)	TD(λ)	TD(1)
Backward View	TD(0)	Forward TD (λ)	MC
Exact Online	TD(0)	Exact TD(λ)	Exact TD(1)

TD (λ) is mix between Monte-Car
and TD(0)