Foundations of Reinforcement Learning

Model-free RL: Monte Carlo and temporal difference (TD) learning

Yuejie Chi

Department of Electrical and Computer Engineering

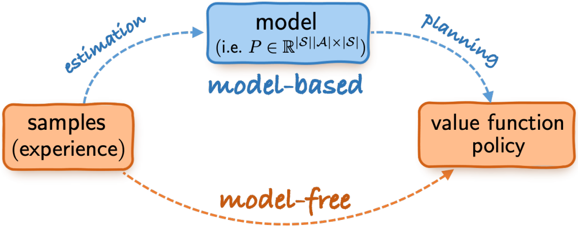
Carnegie Mellon University

Spring 2023

Many materials of this lecture are adapted/stolen from David Silver's online lecture.

Outline

Two approaches to RL



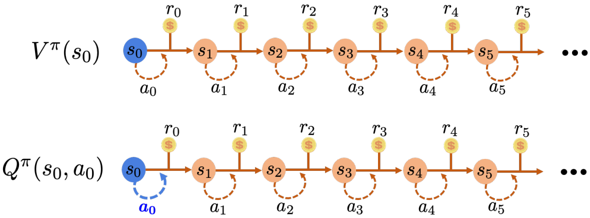
Model-based approach ("plug-in")

1. build an empirical estimate $\widehat{P}$ for P
2. planning based on empirical $\widehat{P}$

Model-free approach

— learning w/o constructing model explicitly

Value function and Q-function



Value function of policy π: cumulative discounted reward

$$\forall s \in \mathcal{S}: \quad V^\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s\right]$$

Q-function of policy π:

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}: \quad Q^\pi(s,a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, s_0 = s, a_0 = a\right]$$

Recap: Bellman's consistency equation

- V^π / Q^π: value / action-value function under policy π

Bellman's consistency equation

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^\pi(s,a)\right]$$

$$Q^\pi(s,a) = \underbrace{\mathbb{E}[r(s,a)]}_{\text{immediate reward}} + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)}\left[\underbrace{V^\pi(s')}_{\text{next state's value}} \right]$$

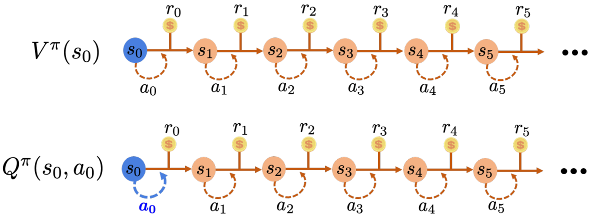
The value/Q function can be decomposed into two parts:

- immediate reward $\mathbb{E}\left[r(s,a)\right]$
- discounted value of at the successor state $\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}V(s')$

Richard Bellman

Monte Carlo policy evaluation

Monte Carlo policy evaluation



Monte Carlo (MC) learns directly from experience by replacing the expectation by empirical means.

- Sample trajectories according to π.
- Calculate the value using empirical means.

Monte Carlo policy evaluation

Consider a trajectory rolled out by following policy π:

$$s_0, a_0, r_0, s_1, a_1, r_1, \dots,$$

The **return** or **reward-to-go** from time t is

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \dots.$$

- $V^\pi(s) = \mathbb{E}[G_t | s_t = s]$;

Idea: to evaluate state s, average the reward-to-gos from time-steps that visit state s over many trajectories.

$$V(s) \approx \frac{\sum_{t:s_t=s} G_t}{\sum_{t:s_t=s} 1}$$

First-visit versus Every-visit

First-visit Monte Carlo:
For each episode, at the first time-step t that state s is visited in an episode.

- Increase the counter $N(s) \leftarrow N(s) + 1$
- Increase the total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return: $V(s) = S(s)/N(s)$

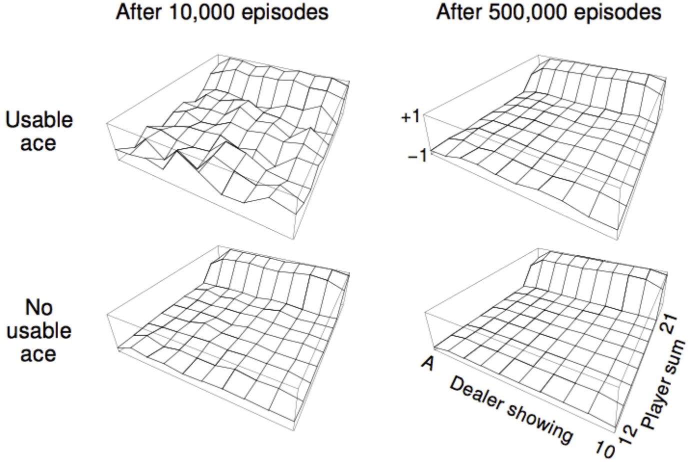
Less bias, more variance

Every-visit Monte Carlo:
For each episode, at the every time-step t that state s is visited in an episode.

- Increase the counter $N(s) \leftarrow N(s) + 1$
- Increase the total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return: $V(s) = S(s)/N(s)$

More bias, less variance

Example: blackjack Monte-Carlo value estimation



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

Incremental Monte Carlo update

The Monte-Carlo value update can be done in an incremental manner to facilitate implementation.

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

$$V(s_t) \leftarrow V(s_t) + \underbrace{\frac{1}{N(s_t)} (G_t - V(s_t))}_{\text{incremental update}}$$

The value $V(s_t)$ is updated towards the actual return G_t.

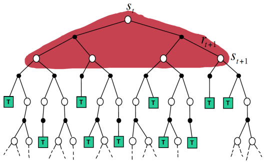
This motivates a more general scheme, which is beneficial specially in non-stationary problems, that one simply does

$$V(s_t) \leftarrow V(s_t) + \alpha (G_t - V(s_t)),$$

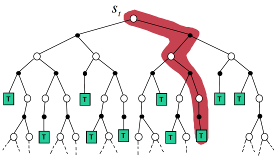
where α is the learning rate to enable more flexible trade-off between past and future (e.g., forgetting faster when $\alpha > \frac{1}{N(s_t)}$).

Dynamic programming versus Monte Carlo

Monte Carlo does not require nor use the Markovian structure.



Dynamic programming
bootstrapping



Monte Carlo
sampling

Caveat of Monte Carlo methods:

- Must wait until the episode to end to calculate the reward-to-go.
- Can only be applied to MDPs when each episode terminates.
- Generally incurs a high variance, but consistent under mild conditions.

Temporal difference (TD) learning

Temporal difference (TD) learning

"If one had to identify one idea as central and novel to RL, it would undoubtedly be TD learning."

Richard Sutton

Temporal difference (TD) learning

- combines dynamic programming and Monte Carlo, by bootstrapping and sampling simultaneously
- learns from incomplete episodes, and does not require the episode to terminate
- "updates a guess towards a guess"

TD learning for value evaluation

- In Monte Carlo, updating the value towards the return:

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

- Instead, TD updates $V(S_t)$ towards estimated return $r_t + \gamma V(s_{t+1})$

$$V(s_t) \longleftarrow V(s_t) + \alpha \underbrace{\left(\underbrace{r_t + \gamma V(s_{t+1})}_{\text{TD target}} - V(s_t) \right)}_{\text{TD error}}$$

  - TD target $r_t + \gamma V(s_{t+1})$: sampling + bootstrapping
  - TD error $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

TD-learning as stochastic approximation

Stochastic approximation [Robbins and Monro, 1951] for solving Bellman equation

$$V = \mathcal{T}^\pi(V),$$

where the Bellman operator $\mathcal{T}^\pi : \mathbb{R}^{|\mathcal{S}|} \mapsto \mathbb{R}^{|\mathcal{S}|}$ is defined as

$$\forall V \in \mathbb{R}^{|\mathcal{S}|} : \qquad \mathcal{T}^\pi(V) = r^\pi + \gamma P^\pi V.$$

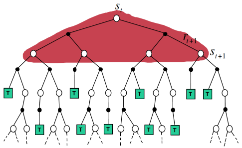
- Access a stochastic realization of $\mathcal{T}^\pi(V)$:

$$\mathcal{T}_t^\pi(V)(s_t) = r_t + \gamma V(s_{t+1})$$

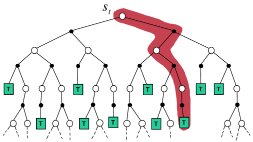
- Update $V(s_t)$ by a weighted combination of old and new:

$$\begin{aligned} V(s_t) &\leftarrow (1-\alpha)V(s_t) + \alpha \mathcal{T}_t^\pi(V)(s_t) \\ &= V(s_t) + \alpha \underbrace{\Big[r_t + \gamma V(s_{t+1}) - V(s_t) \Big]}_{\text{temporal difference}}, \quad t \geq 0 \end{aligned}$$

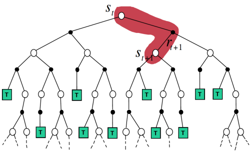
DP versus MC versus TD



Dynamic programming
bootstrapping



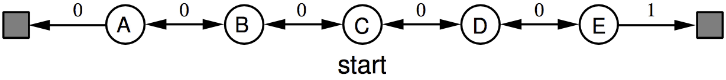
Monte Carlo
sampling

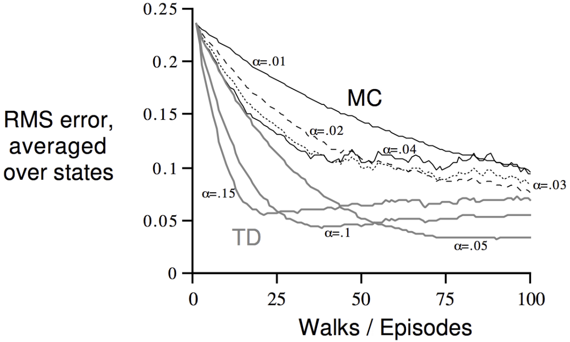


TD learning
bootstrapping+sampling

- TD has much lower variance than MC because of bootstrapping.
- TD learn on-the-fly because of bootstrapping.

Example: random walk

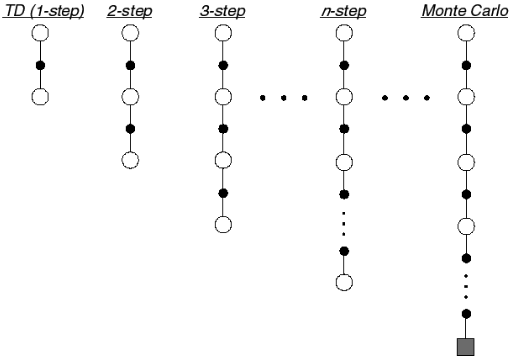




n-step TD

Let the TD target look n steps into the future

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}\big[r_t + \gamma V^\pi(s_{t+1}) | s_t = s\big] && \text{(one-step bootstrap)} \\
&= \mathbb{E}\big[r_t + \gamma r_{t+1} + \gamma^2 V^\pi(s_{t+2}) | s_t = s\big] && \text{(two-step bootstrap)} \\
&= \cdots
\end{aligned}
$$



n-step TD

The n-step return:

$$G_t^{(n)} = r_t + \gamma r_{t+1} + \ldots + \gamma^n V(s_{t+n})$$

- $n = 1$: TD target
- $n = \infty$: MC target

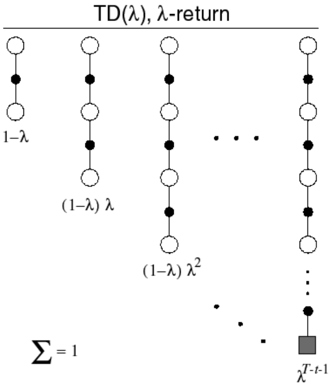
The n-step TD learning:

$$V(s_t) \longleftarrow V(s_t) + \alpha \underbrace{\left(G_t^{(n)} - V(s_t)\right)}_{\text{TD error}}$$

- Mix-and-match: combine information over different n as the TD target, e.g. using
$$\frac{1}{2}G_t^{(2)} + \frac{1}{2}G_t^{(3)}.$$

From n-step TD to TD(λ)

Can we efficiently combine information from all time-steps?



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

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

where $\lambda \in [0,1]$.

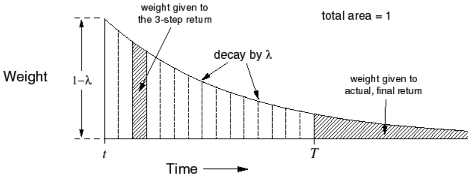
The forward-view TD(λ):

$$V(s_t) \leftarrow V(s_t) + \alpha\,\underbrace{\left(G_t^\lambda - V(s_t)\right)}_{\text{TD error}}$$

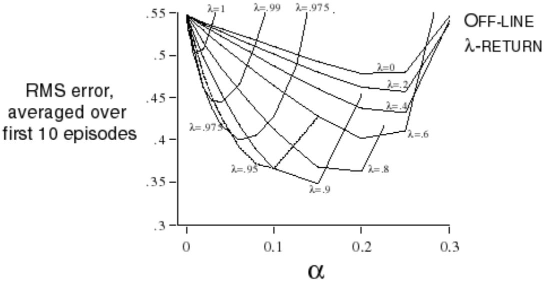
Update towards the λ-return.

Forward-view TD(λ)

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



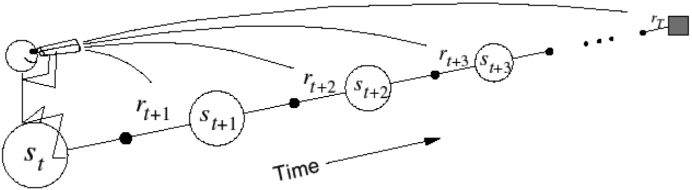
Example: forward-view TD(λ) on random walk



From forward-view TD(λ) to backward-view TD(λ)

Forward-view TD(λ):

- Like MC, requires the episode to terminate to compute G_t^λ.

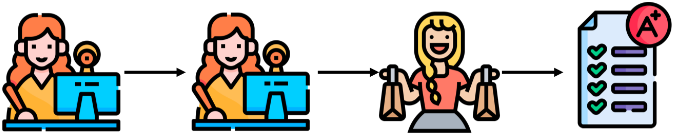


Question

Can we have TD(λ) run on-the-fly?

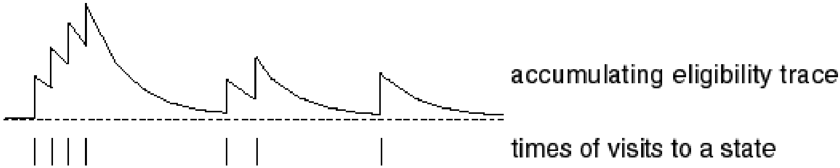
Eligibility traces

Credit assignment: most frequent or most recent



Eligibility traces combine both heuristics:

$$E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbb{I}(s_t = s), \qquad \text{with} \quad E_0(s) = 0$$

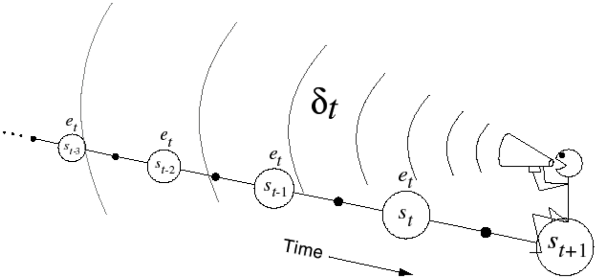


Backward-view TD(λ)

- Keep an eligible trace for every state s
- Update value $V(s)$ for **every state** s, in proportional to TD-error $\delta_t = r_{r+1} + \gamma V(s_{t+1}) - V(s_t)$ and eligibility trace $E_t(s)$:

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

When $\lambda = 0$, $E_t(s) = \mathbb{I}(s_t = s)$, and it reduces to TD$(0)$, the basic TD.



Equivalence of forward/backward-view TD(λ)

- Consider episodic environments (episode length T)

Theorem 1

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

$$\underbrace{\sum_{t=1}^{T} \alpha \delta_t E_t(s)}_{\textit{backward updates}} = \underbrace{\sum_{t=1}^{T} \alpha (G_t^\lambda - V(s_t)) \mathbb{I}(s_t = s)}_{\textit{forward updates}}$$

- Forward view provides theory
- Backward view provides mechanism

Forward/backward-view TD(λ)

Consider an episode where s is visited once at time-step k.

- TD(λ) eligibility trace discounts time since visit,

$$E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbb{I}(s_t = s) = \begin{cases} 0, & t < k \\ (\gamma\lambda)^{t-k}, & t \geq k \end{cases}$$

- 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

Telescoping in TD(λ)

TD errors telescope to λ-error (check!),

$$
\begin{aligned}
\delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \cdots \\
&= r_t + \gamma V(s_{t+1}) - V(s_t) \\
&\quad + (\gamma\lambda) r_{t+1} + \gamma(\gamma\lambda) V(s_{t+2}) - \gamma\lambda V(s_{t+1}) \\
&\quad + (\gamma\lambda)^2 r_{t+2} + \gamma(\gamma\lambda)^2 V(s_{t+3}) - (\gamma\lambda)^2 V(s_{t+2}) + \ldots \\
&= -V(s_t) + (1-\lambda)\lambda^0 (r_t + \gamma V(s_{t+1})) \\
&\qquad + (1-\lambda)\lambda^1 (r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2})) \\
&\qquad + (1-\lambda)\lambda^2 (r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V(s_{t+3})) + \cdots \\
&= G_t^\lambda - V(s_t)
\end{aligned}
$$

where

$$
G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1}\left[r_t + \gamma r_{t+1} + \ldots + \gamma^n V(s_{t+n})\right].
$$

Off-policy evaluation via importance sampling

Off-policy evaluation

Sometimes we are interested in evaluating policy π different from behavior policy μ.

- Learn from observing humans or other agents
- Re-use experience generated from old policies
- Learn about optimal policy while following exploratory policy
- Learn about multiple policies while following one policy

Can we adapt our ideas so far to off-policy evaluation?

Importance sampling

Reevaluation an expectation over one distribution to another:

$$\begin{aligned} \mathbb{E}_{X\sim P}[f(X)] &= \sum P(X) f(X) \\ &= \sum Q(X) \frac{P(X)}{Q(X)} f(X) \\ &= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)} f(X)\right] \end{aligned}$$

- The importance weights: $\frac{P(X)}{Q(X)}$
- Allows evaluating a policy π (drawn from P) when sampling from another policy μ (drawn from Q).

Importance sampling for off-policy Monte Carlo

Multiply importance sampling corrections along whole episode:

$$G_t^{\pi/\mu} = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)} \frac{\pi(a_{t+1}|s_{t+1})}{\mu(a_{t+1}|s_{t+1})} \cdots \frac{\pi(a_T|s_T)}{\mu(a_T|s_T)} G_t$$

Update value towards corrected return:

$$V(s_t) \leftarrow V(s_t) + \alpha \left(G_t^{\pi/\mu} - V(s_t) \right)$$

- High variance when $\frac{\pi(a|s)}{\mu(a|s)}$ is large
- Does not apply when $\frac{\pi(a|s)}{\mu(a|s)}$ is zero: behavior policy μ does not cover the target policy π

Importance sampling for off-policy TD

- Weight TD target by importance sampling [Precup et al., 2001]

$$V(s_t) \longleftarrow V(s_t) + \alpha\left(\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}(r_t + \gamma V(s_{t+1})) - V(s_t)\right)$$

- Lower variance than Monte-Carlo importance sampling

References I

Precup, D., Sutton, R. S., and Dasgupta, S. (2001).
Off-policy temporal-difference learning with function approximation.
In *ICML*, pages 417–424.

Robbins, H. and Monro, S. (1951).
A stochastic approximation method.
The annals of mathematical statistics, pages 400–407.