MACHINE INTELLIGENCE COMMUNITY

Summary of Reinforcement Learning: An Introduction Chapter 7: n-step Boostrapping

Justin Chen[*] chenjus@bu.edu

Tyrone Hou[*] tyroneh@bu.edu

January 22, 2018

Introduction

- **Goals**:
  - Combine Monte Carlo methods and one-step temporal-difference (TD) methods
- **n-step TD methods** - Generalization between both Monte Carlo methods and temporal-difference methods
- One-step TD methods - The same time step determines how often the action can be changed and the time interval over which bootstrapping is done
- In practice, we want to be able to quickly update the action value to account for changes, but bootstrapping performs over long timescales with significant changes
- n-step methods enable bootstrapping over multiple steps

7.1 n-step TD Prediction

- **What space of methods bridge the gap between Monte Carlo and TD methods?**
  - **Monte Carlo methods** estimate v_π for policy π using the *full return* from a given state.
  - **TD methods** bootstrap with *one-step* updates using current reward, but *proxy* the rest of the rewards using the current value estimate for the next state.
- **n-step TD methods** do *n-step updates* with n actual rewards, proxy the rest of the return with v_π for the state n steps later. These generalize both Monte Carlo and one-step TD methods into one spectrum.
- *update target* - Quantity towards which a value estimate is moved during learning

Method	Target
Monte Carlo	$G_t \doteq R_{t+1} + \gamma R_{t+2} + \dots \gamma^{T-t-1} R_T$
one-step TD	$G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1})$
n-step TD	$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \dots \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$

  - In any n-step methods, the target *approximates* the full return using n rewards. If $t+n \geq T$ then $G_{t:t+n} \doteq G_t$.

- **n-step return update rule**

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha \Big[G_{t:t+n} - V_{t+n-1}(S_t) \Big], \quad 0 \leq t < T \tag{1}$$

[*]These authors contributed equally to this work

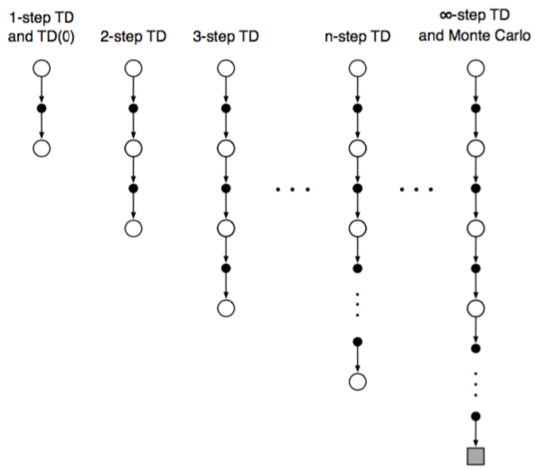

Figure 1: Backup diagrams for n-step methods

- Since this equation needs knowledge of rewards from n future steps, no updates can occur during the *first* $n-1$ steps of each episode.
- During the *last* $n-1$ steps the full return is always composed of less than n rewards, so we don't need to approximate using the value estimate. Instead we make updates using the full return.

n-step TD for estimating $V \approx v_\pi$

Initialize $V(s)$ arbitrarily, $\forall s \in \mathcal{S}$
Parameters: step size $\alpha \in (0,1]$, $n \in \mathbb{Z}^+$
All store/access operations (for S_t and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
    If t < T, then:
      Choose action using $\pi(\cdot|S_t)$
      Observe and store next state and reward pair S_{t+1} and R_{t+1}
      If S_{t+1} is terminal, then $T \leftarrow t+1$
    $\tau \leftarrow t - n + 1$ (where an update occurs to the state estimate at time τ)
    If $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha[G - V(S_\tau)]$
Until $\tau = T - 1$

7.2 n-step Sarsa

- How can n-step methods be used not just for prediction, but also for control?
- *n-step Sarsa* - Sarsa that utilizes bootstrapping over n-steps

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)], 0 \leq t < T$$

  - Values for all other states remain unchanged
  - $Q_{t+n}(s, a) = Q_{t+n-1}(s, a)$ for all $s \neq S_t$ or $a \neq A_t$

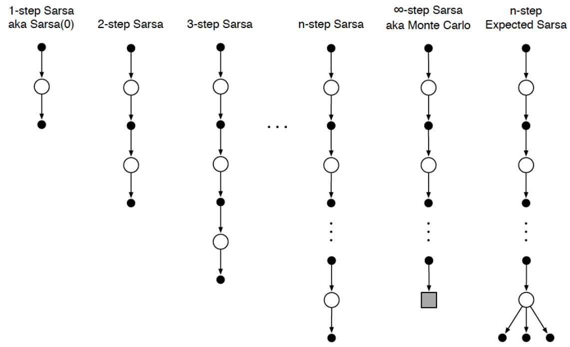

Figure 2: Backup diagrams for the spectrum of n-step methods for state-action values

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ϵ-greedy with respect to Q, or to a fixed given policy
Parameters: step size $\alpha \in (0,1]$, small $\epsilon > 0$, a positive integer n
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq terminal$
  Select and store an action $A_0 \tilde{\pi}(\dot{|}S_0)$
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
  | If t<T, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t+1$
  | else:
  | Select and store and action A_{t+1} $\pi(\dot{|}S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(A_\tau, S_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\dot{|}S_\pi)$ is ϵ-greedy w.r.t. Q
Until $\tau = T - 1$

- Idea: switch states for actions or state-action pairs, and then proceed with ϵ-greedy
- **n-step Returns** - Update targets that use n rewards over n time steps
- **n-step Expected Sarsa** is defined by the same equation as Sarsa, but the return is

$$G_{t:t+n} \doteq R_{t+1} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n \sum_a \pi(a|S_{t+n}) Q_{t+n-1}(S_{t+n}, a),$$

$$\text{for all } n \text{ and } t \text{ such that} n \geq 1 and 0 \leq t \leq T - n$$

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}),$$

$$n \geq 1, 0 \leq t < T - n, \text{ with } G_{t:t+n} \doteq G_t \text{ if } t + n \geq T$$

7.3 n-step Off-policy Learning by Importance Sampling

- Similar to importance sampling methods for Monte Carlo that compare target policy π and behavior policy b. Here we are only interested in the relative probabilities of the n actions taken in an n-step update.
- The product of the ratio of probabilities is called the **importance sampling ratio** ρ:

$$\rho_{t:t+h} \doteq \prod_{k=t}^{min(h,T-1)} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

  - If π never takes an action a, any n step return generated with action a is given zero weight.
  - If π takes action a much more often than b does, the entire return is given a much larger weight.

- **Simple off-policy n-step TD update rule**

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\rho_{t:t+n-1}[G_{t:t+n} - V_{t+n-1}(S_t)], \quad 0 \leq t < T$$

- **off-policy n-step Sarsa update rule**

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha\rho_{t+1:t+n-1}[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)], \quad 0 \leq t < T$$

  - ρ starts at $t+1$ instead of t. We don't care about probability to take action A_t. We only want what happens to the return *given* we already selected the action.

- **off-policy n-step Expected Sarsa udpate rule**

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha\rho_{\mathbf{t+1:t+n-2}}[\mathbf{G_{t:t+n}} - Q_{t+n-1}(S_t, A_t)], \quad 0 \leq t < T$$

  - Uses $\rho_{t+1:t+n-2}$ instead of $\rho t+1 : t+n-1$
  - Expected Sarsa definition of expected return G_t (see 7.2)
  - No need to importance sample *last time step* since we use the expected value and account for probability of taking each action under π.

Off-policy n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π
Important changes in bold

Input: arbitrary ϵ-*soft* behavior policy b
Initialize $Q(s, a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π as ϵ-greedy w.r.t. Q, or as fixed policy
Parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$, $n \in \mathbb{Z}^+$
All store/access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
    Initialize and store $S_0 \neq$ terminal
    Select and store $A_0 \sim b(\cdot|S_0)$
    $T \leftarrow \infty$
    For $t = 0, 1, 2, \ldots$:
        If t $<$ T, then:
            Take action A_t
            Observe and store next state and reward pair S_{t+1} and R_{t+1}
            If S_{t+1} is terminal, then $T \leftarrow t+1$
        else:
            Select and store action $A_{t+1} \sim b(\cdot|S_{t+1})$
        $\tau \leftarrow t - n + 1$ (where an update occurs to the state estimate at time τ)
        If $\tau \geq 0$:
            $\rho \leftarrow \prod_{\mathbf{i}=\tau+\mathbf{1}}^{\mathbf{min}(\tau+\mathbf{n-1},\mathbf{T-1})} \frac{\pi(\mathbf{A_t}|\mathbf{S_t})}{\mathbf{b}(\mathbf{A_t}|\mathbf{S_t})}$

$G \leftarrow \sum_{i=\tau+1}^{min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
If $\tau + n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$
$\mathbf{Q}(\mathbf{S}_\tau, \mathbf{A}_\tau) \leftarrow \mathbf{Q}(\mathbf{S}_\tau, \mathbf{A}_\tau) + \alpha\rho[\mathbf{G} - \mathbf{Q}(\mathbf{S}_\tau, \mathbf{A}_\tau)]$
If π is being learned, change $\pi(\cdot|S_\tau)$ to be ϵ-greedy w.r.t updated Q
Until $\tau = T - 1$

7.4 *Per-reward Off-policy Methods

- More efficient to use per-reward importance sampling
- Recursively defined n-step return:

$$G_{t:h} = R_{t+1} + \gamma G_{t+1:h}$$

- First reward and next state must be weight be importance sampling ratio at time t by $\rho_t = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
- Off-policy n-step return:

$$G_{t:h} \doteq \rho_t(R_{t+1} + \gamma G_{t+1:h}) + (1 - \rho_t)V_{h-1}(S_t), t < h \leq T, \text{where} G_{t:t} \doteq V_{t-1}(S_t)$$

  - Using policy b to try to model target π
  - If π would not select action at timestep t, then same as weighing b by ρ_t but this can cause high variance because of the resulting zeros
  - Importance sampling ratio has expected value of one and is not correlated with the estimate

7.5 Off-policy Learning Without Importance Sampling: The n-step Tree Backup Algorithm

- How to do off-policy learning without importance sampling? Extend the idea of n-step Expected Sarsa to create the **tree backup algorithm**
- The update equation backs up the current value estimate for state-action pair (S_t, A_t) using the action nodes of the tree below (the filled in circles).
- There are two 'half-steps' for the update, corresponding to two types of nodes:
  1. **Leaf nodes** - Accounts for estimated probability and Q-value of actions *not taken*. Each leaf node a contributes weight $\pi(a|S_{t+j})$ to the target, the probability of action a occurring under target policy π.
  2. **Center nodes** - Uses reward to account for actions actually taken at each time step. Each center node A_{t+i} weights *all* the values from the next level with $\pi(A_{t+i}|S_{t+i})$

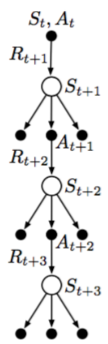

Figure 3: 3-step tree backup update

- **One-step return target** is just Expected Sarsa:

$$G_{t:t+1} \doteq R_{t+1} + \gamma \sum_a \pi(a|S_{t+1})Q(S_{t+1}, A_{t+1}), \quad t < T-1$$
$$G_{T-1:T} \doteq R_T$$

- **n-step return target** - Recursive definition with one-step return as base case.

$$G_{t:t+n} \doteq R_{t+1}+\gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1})Q_{t+n-1}(S_{t+1}, a)+\gamma\pi(A_{t+1}|S_{t+1})G_{t+1:t+n}, \;\; t+1 < T, n > 1$$

- **n-step tree backup update rule** - Same as n-step Sarsa

$$Q_{t+n}(S_t, A_t) = Q_{t+n-1}(S_t, A_t) + \alpha[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)], \quad 0 \leq t < T$$

n-step Tree Backup for estimating $Q \approx q_\pi$, or $Q \approx q_\pi$ for a given π
Initialize $Q(s, a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ϵ-greedy w.r.t. Q, or as fixed policy
Parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$, $n \in \mathbb{Z}^+$
All store/access operations can take their index mod n

Repeat (for each episode):
    Initialize and store $S_0 \neq$ terminal
    Select and store $A_0 \sim b(\cdot|S_0)$
    Store $Q(S_0, A_0)$ as Q_0
    $T \leftarrow \infty$
    For $t = 0, 1, 2, \ldots$:
        If t < T, then:
            Take action A_t
            Observe and store next state and reward pair S_{t+1} and R_{t+1}
            If S_{t+1} is terminal:
                $T \leftarrow t + 1$
                Store $R - Q_t$ as δ_t
        else:
            Store $R + \gamma \sum_a \pi(a|S_{t+1})Q(S_{t+1}, a) - Q_t$ as δ_t
            Select arbitrarily and store action A_{t+1}
            Store $Q(S_{t+1}, A_{t+1})$ as Q_{t+1}
            Store $\pi(A_{t+1}|S_{t+1})$ as π_{t+1}
        $\tau \leftarrow t - n + 1$ (where an update occurs to the state estimate at time τ)
        If $\tau \geq 0$:
            $Z \leftarrow 1$
            $G \leftarrow Q_\tau$
            For $k = \tau, \ldots, min(\tau + n - 1, T - 1)$:
                $G \leftarrow G + Z\delta_k$
                $Z \leftarrow \gamma Z \pi_{k+1}$
            $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$
            If π is being learned, ensure $\pi(\cdot|S_\tau)$ is ϵ-greedy w.r.t updated $Q(S_\tau, \cdot)$
Until $\tau = T - 1$

7.6 *A Unifying Algorithm: n-step $\mathcal{Q}(\sigma)$

- Two ends of a spectrum: Sample action (Sarsa) or consider expectation over all actions (tree-backup algorithm)

- Expected Sarsa would sample all steps expect for the last step

$$\underset{\text{Always sample}}{\text{Sarsa}} \longleftrightarrow \underset{\text{Never sample}}{\text{Tree-backup}}$$

- In between are methods that sample on some time steps and use expected transitions on others.
- Let each step have $\sigma \in [0, 1]$, a degree of sampling at time t.
- Sarsa has $\sigma = 0$ for all time steps, Tree backup has $\delta = 1$ for all steps.

$$\underset{\text{Full sampling}}{\sigma_t = 0} \quad \longleftrightarrow \quad \underset{\text{No sampling}}{\sigma_t = 1}$$

- The n-step algorithm using σ is called $\mathbf{Q}(\sigma)$

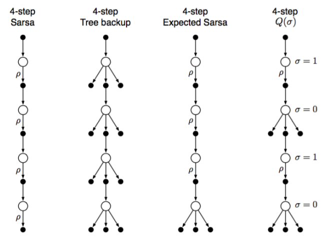

Figure 4: 4 types of n-step backup diagrams covered so far

- How do we define generalize n-step return using δ?
- Remember for n-step Sarsa

$$G_{t:t+n} = Q_{t-1}(S_t, A_t) + \sum_{k=t}^{min(t+n-1, T-1)} \gamma^{k-t}[R_{k+1} + \gamma Q_k(S_{k+1}, A_{k+1}) - Q_{k-1}(S_k, A_k)]$$

- where we define $\delta_k = R_{k+1} + \gamma Q_k(S_{k+1}, A_{k+1}) - Q_{k-1}(S_k, A_k)$ as the TD error.
- We can generalize TD error to 'slide' with δ_k to take into account either more sample or expected values.

$$\begin{aligned}
\delta_k &\doteq R_{t+1} + \gamma[\sigma_{t+1} Q_t(S_{t+1}, A_{t+1}) + (1 - \sigma_{t+1}\bar{Q}_{t+1}] - Q_{t-1}(S_t, A_t) \\
\bar{Q}_t &\doteq \sum_a \pi(a|S_t) Q_{t-1}(S_t, a)
\end{aligned}$$

- Then we define n-step return for $Q(\sigma)$ for the on-policy case as:

$$\begin{aligned}
G_{t:t+1} &\doteq R_{t+1} + \gamma[\sigma_{t+1} Q_t(S_{t+1}, A_{t+1})_(1 - \sigma_{t+1}\bar{Q}_{t+1}] \qquad = \delta_t + Q_{t-1}(S_t, A_t) \\
G_{t:t+2} &\doteq R_{t+1} + \gamma[\sigma_{t+1} Q_t(S_{t+1}, A_{t+1}) + (1 - \sigma_{t+1}\bar{Q}_{t+1}] \\
&\quad - \gamma(1 - \sigma_{t+1}\pi(A_{t+1}|S_{t+1}) Q_t(S_{t+1}, A_{t+1}) \\
&\quad + \gamma(1 - \sigma_{t+1})\pi(A_{t+1}|S_{t+1})[R_{t+2} + \gamma[\sigma_{t+2} Q_t(S_{t+2}, A_{t+2}) + (1 - \sigma_{t+2}\bar{Q}_{t+2}]] \\
&\quad - \gamma\sigma_{t+1} Q_t(S_{t+1}, A_{t+1}) \\
&\quad + \gamma\sigma_{t+1}[R_{t+2} + \gamma[\sigma_{t+2} Q_t(S_{t+2}, A_{t+2}) + (1 - \sigma_{t+2}\bar{Q}_{t+2}]] \\
&\quad = Q_{t-1}(S_t, A_t) + \delta_t \\
&\quad + \gamma(1 - \sigma_{t+1})\pi(A_{t+1}|S_{t+1})\delta_{t+1} \\
&\quad + \gamma\sigma_{t+1}\delta_{t+1} \\
&\quad = Q_{t-1}(S_t, A_t) + \delta_t + \gamma[(1 - \sigma_{t+1})\pi(A_{t+1}|S_{t+1}) + \sigma_{t+1}]\delta_{t+1} \\
G_{t:t+n} &\doteq Q_{t-1}(S_t, A_t) + \sum_{k=t}^{min(t+n-1, T-1)} \delta_k \prod_{i=t+1}^{k} \gamma[(1 - \sigma_i)\pi(A_i|S_i) + \sigma_i]
\end{aligned}$$

- For the off-policy case, we use importance sampling and redefine ρ in terms of σ:

$$\rho_{t:h} \doteq \prod_{k=t}^{min(h,T-1)} \left(\sigma_k \frac{\pi(A_k|S_k)}{b(A_k|S_k)} + 1 - \sigma_k \right)$$

Off-policy n-step $Q(\sigma)$ for estimating $Q \approx q_\pi$, or $Q \approx q_\pi$ for a given π
Input: an arbitrary ϵ-soft behavior policy b
Initialize $Q(s, a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ϵ-greedy w.r.t. Q, or as fixed policy
Parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$, $n \in \mathbb{Z}^+$
All store/access operations can take their index mod n

Repeat (for each episode):
    Initialize and store $S_0 \neq$ terminal
    Select and store $A_0 \sim b(\cdot|S_0)$
    Store $Q(S_0, A_0)$ as Q_0
    $T \leftarrow \infty$
    For $t = 0, 1, 2, \ldots$:
        If t < T:
            Take action A_t
            Observe and store next state and reward pair S_{t+1} and R_{t+1}
            If S_{t+1} is terminal:
                $T \leftarrow t + 1$
                Store $R - Q_t$ as δ_t
        else:
            Select and store action $A_{t+1} \sim b(\cdot|S_{t+1})$
            Select and store σ_{t+1}
            Store $Q(S_{t+1}, A_{t+1})$ as Q_{t+1}
            Store $R + \gamma\sigma_{t+1}Q_{t+1} + \gamma(1 - \sigma_{t+1}) \sum_a \pi(a|S_{t+1})Q(S_{t+1}, a) - Q_t$ as δ_t
            Store $\pi(A_{t+1}|S_{t+1})$ as π_{t+1}
            Store $\frac{\pi(A_t|S_t)}{b(A_t|S_t)}$ as ρ_{t+1}
        $\tau \leftarrow t - n + 1$ (where an update occurs to the state estimate at time τ)
        If $\tau \geq 0$:
            $\rho \leftarrow 1$
            $Z \leftarrow 1$
            $G \leftarrow Q_\tau$
            For $k = \tau, \ldots, min(\tau + n - 1, T - 1)$:
                $G \leftarrow G + Z\delta_k$
                $Z \leftarrow \gamma Z[(1 - \sigma_{k+1})\pi_{k+1} + \sigma_{k+1}]$
                $\rho \leftarrow \rho(1 - \sigma_k + \sigma_k\rho_k)$
            $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho[G - Q(S_\tau, A_\tau)]$
            If π is being learned, ensure $\pi(\cdot|S_\tau)$ is ϵ-greedy w.r.t updated $Q(S_\tau, \cdot)$
Until $\tau = T - 1$

7.7 Summary

- This chapter focused a spectrum of temporal-difference methods with one-step TD at one end and Monte Carlo methods on the other
- Bootstrapping performs better than either extreme
- n-step methods look ahead at the next n rewards, states, and actions before updating which requires *more time and space per timestep* than methods that do not look ahead
- Increased cost of n-steps is an inherent limitation
- Advantage of n-step methods is that they're conceptually clear

References

Sutton, Richard S., and Andrew G. Barto. “Temporal-Difference Learning.” Reinforcement Learning: An Introduction, The MIT Press, 2018, pp. 97-114.