Multi-step Bootstrapping

Jennifer She

Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto

February 7, 2017

Multi-step Bootstrapping

- Generalization of Monte Carlo methods and one-step TD methods
  - Includes methods that lie in-between these two extremes
  - Methods based on sample episodes of states, actions and rewards

Multi-step Bootstrapping

- Generalization of Monte Carlo methods and one-step TD methods
  - Includes methods that lie in-between these two extremes
  - Methods based on sample episodes of states, actions and rewards
- Time intervals for making updates and bootstrapping are no longer the same
  - Enables bootstrapping to occur over longer time intervals

Prediction Problem (Policy Evaluation)

- Given a fixed policy π, estimate the state-value function v_π

Prediction Problem (Policy Evaluation)

- Given a fixed policy π, estimate the state-value function v_π
- Monte Carlo update

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

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 Rt + 3 + ... + \gamma^{T-t-1} R_T$$

  - Updates of the state-value estimates happen at the end of each episode
  - G_t is the complete return of an episode after S_t
  - No bootstrapping involved (does not use other estimations)

Prediction Problem (Policy Evaluation)

- One-step TD update

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

  - Updates happen one step later, bootstrapping using $V_t(S_{t+1})$
  - $R_{t+1} + \gamma V_t(S_{t+1})$ approximates G_t

n-step TD Prediction

- Approximate G_t by looking ahead n steps
  - Bootstrap using $V_{t+n-1}(S_{t+n})$

$$G_t^{(n)} = \begin{cases} R_{t+1} + \gamma R_{t+2} + ... + \gamma^n V_{t+n-1}(S_{t+n}) & 0 \leq t < T - n \\ G_t & t + n \geq T \end{cases}$$

  - Incorporate discounted rewards up to R_{t+n}
- $G_t^{(n)}$ is called the n-step return

n-step TD Prediction

- Approximate G_t by looking ahead n steps
  - Bootstrap using $V_{t+n-1}(S_{t+n})$

$$G_t^{(n)} = \begin{cases} R_{t+1} + \gamma R_{t+2} + ... + \gamma^n V_{t+n-1}(S_{t+n}) & 0 \leq t < T - n \\ G_t & t + n \geq T \end{cases}$$

  - Incorporate discounted rewards up to R_{t+n}
- $G_t^{(n)}$ is called the n-step return
- $G_t^{(1)}$ for one-step TD
- $G_t^{(T)}$ for Monte Carlo

n-step TD Prediction

- For $n > 1$, $V_{t+n-1}(S_{t+n})$ involves future rewards and value functions not available at time between t and $t+1$

n-step TD Prediction

- For $n > 1$, $V_{t+n-1}(S_{t+n})$ involves future rewards and value functions not available at time between t and $t+1$
  - Must wait until time $t+n$ to update $V(S_t)$

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha(G_t^{(n)} - V_{t+n-1}(S_{t+1})) \quad 0 \leq t < T$$

n-step TD Prediction

- For $n > 1$, $V_{t+n-1}(S_{t+n})$ involves future rewards and value functions not available at time between t and $t+1$
  - Must wait until time $t+n$ to update $V(S_t)$

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha(G_t^{(n)} - V_{t+n-1}(S_{t+1})) \quad 0 \leq t < T$$

- No updates during the first $n-1$ time steps
- $n-1$ updates at the end of the episode using G_t

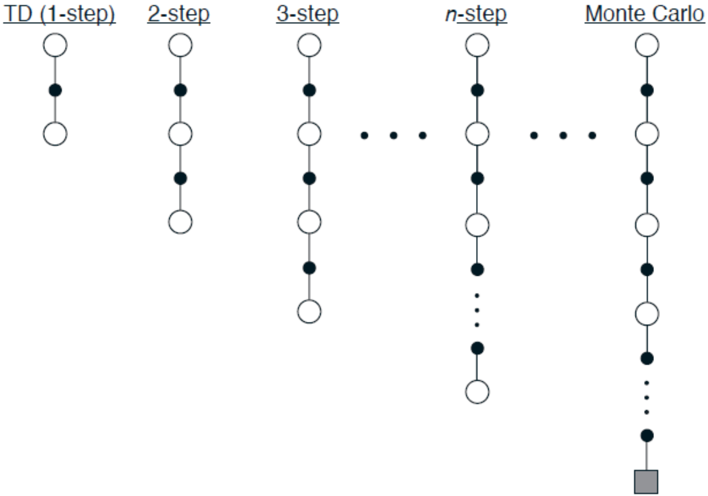
n-step TD Prediction

- For $n > 1$, $V_{t+n-1}(S_{t+n})$ involves future rewards and value functions not available at time between t and $t+1$
  - Must wait until time $t+n$ to update $V(S_t)$

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha(G_t^{(n)} - V_{t+n-1}(S_{t+1})) \quad 0 \leq t < T$$

- No updates during the first $n-1$ time steps
- $n-1$ updates at the end of the episode using G_t
- Still considered TD methods ($n < T$)
  - Involves changing an earlier estimate based on how it differs from a later estimate

n-step TD Prediction



n-step TD Prediction

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

Initialize $V(s)$ arbitrarily, $s \in \mathcal{S}$
Parameters: step size $\alpha \in (0, 1]$, a positive integer n
All store and 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 $t = 0, 1, 2, \dots$:
  | If $t < T$, then:
  | Take an action according to $\pi(\cdot|S_t)$
  | 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 $T \leftarrow t + 1$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose state's 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: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_\tau^{(n)})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha \left[G - V(S_\tau)\right]$
  Until $\tau = T - 1$

n-step TD Prediction

- The expected n-step return is guaranteed to be a better estimate of v_π than V_{t+n-1} in the worst case

$$\max_s |\mathbb{E}[G_t^{(n)} | S_t = s] - v_\pi(s)| \leq \gamma^n \max_s |V_{t+n-1}(s) - v_\pi(s)|$$

n-step TD Prediction

- The expected n-step return is guaranteed to be a better estimate of v_π than V_{t+n-1} in the worst case

$$\max_s |\mathbb{E}[G_t^{(n)}|S_t = s] - v_\pi(s)| \leq \gamma^n \max_s |V_{t+n-1}(s) - v_\pi(s)|$$

  - All n-step TD methods converge to correct predictions under appropriate technical conditions

Example

- Random walk starting from state C
- Rewards are all 0 except when following the right arrow from state E
- True state-values from A to E are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}$

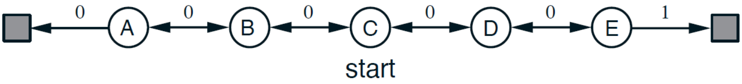
Example

- Random walk starting from state C
- Rewards are all 0 except when following the right arrow from state E
- True state-values from A to E are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}$
- Initialize with $V(s) = 0.5, \forall s$

Example

- Random walk starting from state C
- Rewards are all 0 except when following the right arrow from state E
- True state-values from A to E are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}$
- Initialize with $V(s) = 0.5, \forall s$
- Suppose the first episode goes from C to the right, through D and E

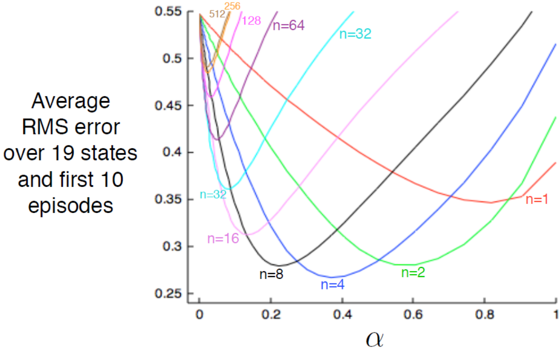
Example



- Random walk starting from state C
- Rewards are all 0 except when following the right arrow from state E
- True state-values from A to E are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}$
- Initialize with $V(s) = 0.5, \forall s$
- Suppose the first episode goes from C to the right, through D and E
- At the end of the episode
  - For a one-step method, only $V(E)$ incremented towards 1
  - For a two-step method, both $V(D)$ and $V(E)$ incremented towards 1
  - For $n \geq 3$, all $V(C)$, $V(D)$ and $V(E)$ incremented towards 1

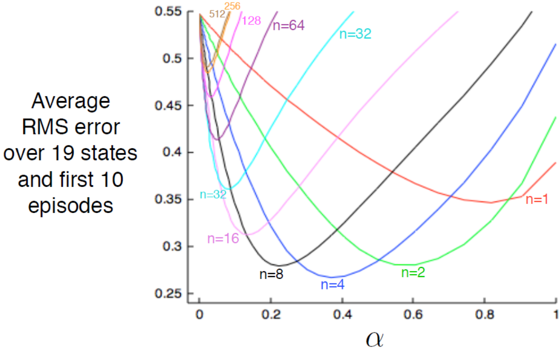
Example

- Empirical comparison for a similar problem
- Random walk with 19 states
- All rewards are 0 except the left-most being -1



Example

- Empirical comparison for a similar problem
- Random walk with 19 states
- All rewards are 0 except the left-most being -1



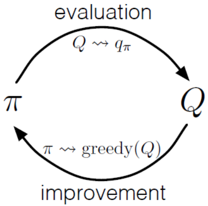
- An intermediate value of *n* works best

Control Problem (Policy Evaluation + Policy Improvement)

- Find an optimal policy π_*

Control Problem (Policy Evaluation + Policy Improvement)

- Find an optimal policy π_*
- Alternate estimating action-value function q_π (evaluation) and updating policy π (improvement)



  - Estimate q_π instead of v_π because we need this information to decide the next π

Control Problem (On-Policy)

Evaluation step

- Monte Carlo evaluation

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

- Sarsa (one-step on-policy TD) evaluation

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) - Q_t(S_t, A_t))$$

  - $R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1})$ approximates G_t

Control Problem (On-Policy)

Evaluation step

- Monte Carlo evaluation

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

- Sarsa (one-step on-policy TD) evaluation

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) - Q_t(S_t, A_t))$$

  - $R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1})$ approximates G_t

Improvement step

- ϵ-greedy (or any other ϵ-soft policy) helps maintain exploration

$$A^* \leftarrow \text{argmax}_a Q(S_t, a)$$

$\forall a \in A(S_t)$,

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \epsilon + \epsilon/|A(S_t)| & a = A^* \\ \epsilon & a \neq A^* \end{cases}$$

n-step Sarsa

- Modification to evaluation step
- Similar to prediction, approximate G_t with

$$G_t^{(n)} = \begin{cases} R_{t+1} + \gamma R_{t+2} + ... + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}) & 0 \leq t < T - n \\ G_t & t + n \geq T \end{cases}$$

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha(G_t^{(n)} - Q_{t+n-1}(S_t, A_t)) \quad 0 \leq t < T$$

n-step Sarsa

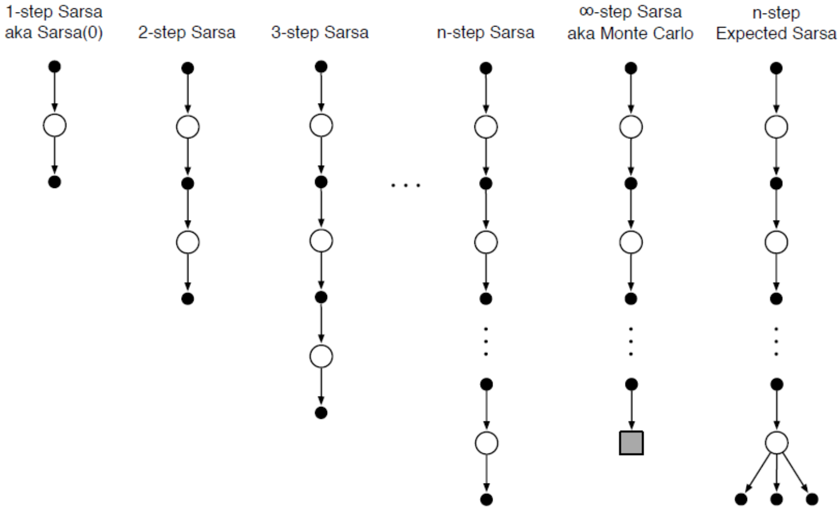
- Modification to evaluation step
- Similar to prediction, approximate G_t with

$$G_t^{(n)} = \begin{cases} R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}) & 0 \leq t < T - n \\ G_t & t + n \geq T \end{cases}$$

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha(G_t^{(n)} - Q_{t+n-1}(S_t, A_t)) \quad 0 \leq t < T$$

- Expected Sarsa
  - Replace $Q_{t+n-1}(S_{t+n}, A_{t+n})$ with $\mathbb{E}[Q_{t+n-1}(S_{t+n}, A_{t+n})|S_{t+n}] = \sum_a \pi(a|S_{t+n}) Q_{t+n-1}(S_{t+n}, a)$
  - Moves deterministically in same direction Sarsa moves in expectation
  - Requires more computation but eliminates variance from sampling A_{t+n}

n-step Sarsa


1-step Sarsa
aka Sarsa(0)
2-step Sarsa
3-step Sarsa
n-step Sarsa
∞-step Sarsa
aka Monte Carlo
n-step
Expected Sarsa

n-step Sarsa

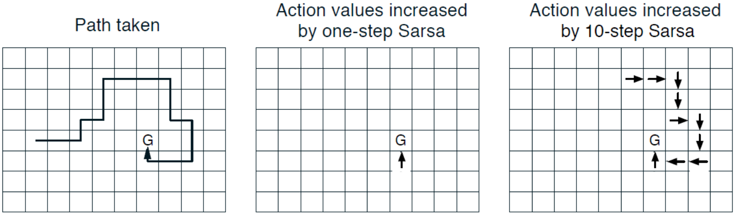
n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π

Initialize $Q(s,a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or to a fixed given policy
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, 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 $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (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 $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_\tau^{(n)})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

Example

- Gridworld scenario where rewards at all states are 0 except a positive reward on the square G
- Initialize $V(s) = 0, \forall s$
- Suppose you take a path on the first episode, and you end at G



- At the end of the episode
  - One-step method only strengthens the last state-actions pair in the path for the next policy
  - n-step method strengthens the last n state-actions pairs in the path for the next policy

Control Problem (Off-Policy)

- Learn the value for one policy π while following another policy μ
  - π often greedy and μ exploratory (ex. ϵ-greedy)
  - Requires that $\pi(a|s) > 0$ implies $\mu(a|s) > 0$

Control Problem (Off-Policy)

- Learn the value for one policy π while following another policy μ
  - π often greedy and μ exploratory (ex. ϵ-greedy)
  - Requires that $\pi(a|s) > 0$ implies $\mu(a|s) > 0$
- Importance sampling (Monte Carlo)
  - Step size takes into account the difference between π and μ using relative probability of all the subsequent actions

$$V(S_t) \leftarrow V(S_t) + \alpha\rho_t^T(G_t - V(S_t))$$

  - ρ_t^T is the importance sampling ratio

$$\rho_t^T = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)P(S_{k+1}|S_k, A_k)}{\mu(A_k|S_k)P(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}$$

Off-Policy Learning by Importance Sampling

- In n-step methods, returns are constructed over n steps
  - Interested in the relative probability of just those n actions
  - Incorporate ρ_t^{t+n} (in place of ρ_t^T) into TD

$$\rho_t^{t+n} = \prod_{k=t}^{\min(t+n, T-1)} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}$$

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \rho_t^{t+n} (G_t^{(n)} - V_{t+n-1}(S_t)) \quad 0 \leq t < T$$

Off-Policy Learning by Importance Sampling

- In n-step methods, returns are constructed over n steps
  - Interested in the relative probability of just those n actions
  - Incorporate ρ_t^{t+n} (in place of ρ_t^T) into TD

$$\rho_t^{t+n} = \prod_{k=t}^{\min(t+n, T-1)} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}$$

$$V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \rho_t^{t+n} (G_t^{(n)} - V_{t+n-1}(S_t)) \quad 0 \leq t < T$$

- If any $\pi(A_k|S_k) = 0$, then $\rho_t^{t+n} = 0$ and return would be totally ignored
- If any $\pi(A_k|S_k) >> \mu(A_k|S_k)$, then ρ_t^{t+n} increases weight given to return, which compensates for action being rarely selected under μ

Off-Policy Learning by Importance Sampling

- Evaluation step
  - ρ_{t+1}^{t+n} replaces ρ_t^{t+n} because requires no further sampling of A_t
  - A_t already determined

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1}^{t+n}(G_t^{(n)} - Q_{t+n-1}(S_t, A_t)) \quad 0 \leq t < T$$

Off-Policy Learning by Importance Sampling

- Evaluation step
  - ρ_{t+1}^{t+n} replaces ρ_t^{t+n} because requires no further sampling of A_t
  - A_t already determined

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha\rho_{t+1}^{t+n}(G_t^{(n)} - Q_{t+n-1}(S_t, A_t)) \quad 0 \leq t < T$$

- Expected Sarsa
  - ρ_{t+1}^{t+n-1} replaces ρ_{t+1}^{t+n} because requires no sampling of A_{t+n}
  - Expected value all actions on $(t+n)$th step into account

Off-Policy Learning by Importance Sampling

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

Input: an arbitrary behavior policy μ such that $\mu(a|s) > 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s,a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or as a fixed given policy
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, 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 ≠ terminal
   Select and store an action A_0 ~ μ(·|S_0)
   T ← ∞
   For t = 0, 1, 2, ... :
   |  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 ← t + 1
   |     else:
   |        Select and store an action A_{t+1} ~ μ(·|S_{t+1})
   |  τ ← t − n + 1    (τ is the time whose estimate is being updated)
   |  If τ ≥ 0:
   |     ρ ← ∏_{i=τ+1}^{min(τ+n−1, T−1)} π(A_i|S_i) / μ(A_i|S_i)        (ρ_{τ+1}^{t+n})
   |     G ← ∑_{i=τ+1}^{min(τ+n, T)} γ^{i−τ−1} R_i
   |     If τ + n < T, then: G ← G + γ^n Q(S_{τ+n}, A_{τ+n})           (G_τ^{(n)})
   |     Q(S_τ, A_τ) ← Q(S_τ, A_τ) + αρ [G − Q(S_τ, A_τ)]
   |     If π is being learned, then ensure that π(·|S_τ) is ε-greedy wrt Q
   Until τ = T − 1
```

Off-Policy Learning by Importance Sampling

- Importance sampling enables off-policy at the cost of increasing the variance of the updates
  - Requires smaller step sizes and thus slower
  - Slower speed inevitable because using less relevant data

Off-Policy Learning by Importance Sampling

- Importance sampling enables off-policy at the cost of increasing the variance of the updates
  - Requires smaller step sizes and thus slower
  - Slower speed inevitable because using less relevant data
- Improvements
  - Autostep method (Mahmood et al, 2012)
  - Invariant updates (Karampatziakis and Langford, 2010)
  - Usage technique (Mahmood and Sutton, 2015)

Off-Policy Learning by Importance Sampling

- Importance sampling enables off-policy at the cost of increasing the variance of the updates
  - Requires smaller step sizes and thus slower
  - Slower speed inevitable because using less relevant data
- Improvements
  - Autostep method (Mahmood et al, 2012)
  - Invariant updates (Karampatziakis and Langford, 2010)
  - Usage technique (Mahmood and Sutton, 2015)
- Off-policy possible without importance sampling?

Control Problem (Off-Policy)

- Expected Sarsa (on-policy, one-step case)

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma \mathbb{E}[Q_t(S_{t+1}, A_{t+1}) | S_{t+1}] - Q_t(S_t, A_t))$$

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a) - Q_t(S_t, A_t))$$

Control Problem (Off-Policy)

- Expected Sarsa (on-policy, one-step case)

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma \mathbb{E}[Q_t(S_{t+1}, A_{t+1}) | S_{t+1}] - Q_t(S_t, A_t))$$

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a) - Q_t(S_t, A_t))$$

- Use a different policy μ to generate behaviour

Control Problem (Off-Policy)

- Expected Sarsa (on-policy, one-step case)

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma\mathbb{E}[Q_t(S_{t+1}, A_{t+1})|S_{t+1}] - Q_t(S_t, A_t))$$

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma\sum_a \pi(a|S_{t+1})Q_t(S_{t+1}, a) - Q_t(S_t, A_t))$$

- Use a different policy μ to generate behaviour
  - Updated values are independent of $\mu(A_{t+1}|S_{t+1})$
  - If π is greedy, this is exactly the Q-learning method

$$\pi(a|S_{t+1}) = \begin{cases} 1 & a = \text{argmax}_{a'} Q(S_{t+1}, a') \\ 0 & \text{otherwise} \end{cases}$$

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma\max_a Q_t(S_{t+1}, a) - Q_t(S_t, A_t))$$

Control Problem (Off-Policy)

- Expected Sarsa (on-policy, one-step case)

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma \mathbb{E}[Q_t(S_{t+1}, A_{t+1})|S_{t+1}] - Q_t(S_t, A_t))$$

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a) - Q_t(S_t, A_t))$$

- Use a different policy μ to generate behaviour
  - Updated values are independent of $\mu(A_{t+1}|S_{t+1})$
  - If π is greedy, this is exactly the Q-learning method

$$\pi(a|S_{t+1}) = \begin{cases} 1 & a = \mathrm{argmax}_{a'} Q(S_{t+1}, a') \\ 0 & \text{otherwise} \end{cases}$$

$$Q_{t+1}(S_t, A_t) \leftarrow Q_t(S_t, A_t) + \alpha(R_t + \gamma \max_a Q_t(S_{t+1}, a) - Q_t(S_t, A_t))$$

  - Possible to form off-policy methods without importance sampling

Off-Policy Learning Without Importance Sampling: The *n*-step Tree Backup Algorithm

- Alternate the incorporation of expected values of future action-value estimates and corrections based on actual steps up to S_{t+n}

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

- Alternate the incorporation of expected values of future action-value estimates and corrections based on actual steps up to S_{t+n}

$$
\begin{aligned}
G_t^{(n)} = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a) \\
- \gamma\pi(A_{t+1}|S_{t+1}) Q_t(S_{t+1}, A_{t+1}) \\
+ \gamma\pi(A_{t+1}|S_{t+1})(R_{t+2} + \gamma \sum_a \pi(a|S_{t+2}) Q_{t+1}(S_{t+2}, a)) \\
- \gamma^2\pi(A_{t+1}|S_{t+1})\pi(A_{t+2}|S_{t+2}) Q_{t+1}(S_{t+2}, A_{t+2}) \\
+ \gamma^2\pi(A_{t+1}|S_{t+1})\pi(A_{t+2}|S_{t+2})(R_{t+3} + \gamma \sum_a \pi(a|S_{t+3}) Q_{t+2}(S_{t+3}, a)) \\
+ \dots \\
+ \gamma^{\min(t+n,T)-1}(\prod_{i=t+1}^{\min(t+n,T)-1} \pi(A_i|S_i)) \\
(R_{\min(t+n,T)} + \gamma \sum_a \pi(a|S_{\min(t+n,T)}) Q_{\min(t+n,T)}(S_{\min(t+n,T)}, a))
\end{aligned}
$$

Off-Policy Learning Without Importance Sampling: The *n*-step Tree Backup Algorithm

- Define "TD error" δ_t to simplify notation

$$\delta_t = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a) - Q_{t-1}(S_t, A_t)$$

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

- Define "TD error" δ_t to simplify notation

$$\delta_t = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a) - Q_{t-1}(S_t, A_t)$$

$$G_t^{(n)} = Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n,T)-1} \delta_k \prod_{i=t+1}^{k} \gamma \pi(A_i|S_i)$$

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha(G_t^{(n)} - Q_{t+n-1}(S_{t+n}, A_{t+n}))$$

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

- Define "TD error" δ_t to simplify notation

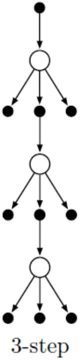
$$\delta_t = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a) - Q_{t-1}(S_t, A_t)$$

$$G_t^{(n)} = Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n,T)-1} \delta_k \prod_{i=t+1}^{k} \gamma \pi(A_i|S_i)$$

$$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha(G_t^{(n)} - Q_{t+n-1}(S_{t+n}, A_{t+n}))$$

- $G_t^{(1)}$ is used for Expected Sarsa

Off-Policy Learning Without Importance Sampling: The *n*-step Tree Backup Algorithm



Off-Policy Learning Without Importance Sampling: The *n*-step Tree Backup Algorithm

n-step Tree Backup for estimating $Q \approx q_*$, 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 with respect to Q, or as a fixed given policy
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n
All store and access operations can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  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 the next reward R; observe and store the next state as S_{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 an action as 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}
    $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    If $\tau \geq 0$:
      $E \leftarrow 1$
      $G \leftarrow Q_\tau$
      For $k = \tau, \ldots, \min(\tau + n - 1, T - 1)$:
        $G \leftarrow G + E\delta_k$
        $E \leftarrow \gamma E \pi_{k+1}$
      $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha \left[G - Q(S_\tau, A_\tau)\right]$
      If π is being learned, then ensure that $\pi(a|S_\tau)$ is ε-greedy wrt $Q(S_\tau, \cdot)$
  Until $\tau = T - 1$

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

- The n-step Tree Backup algorithm is the natural extension of Q-learning to the multi-step case
  - Requires no importance sampling like Q-learning

Off-Policy Learning Without Importance Sampling: The *n*-step Tree Backup Algorithm

- The *n*-step Tree Backup algorithm is the natural extension of Q-learning to the multi-step case
  - Requires no importance sampling like Q-learning
- However, if μ and π vastly differ then $\pi(A_{t+i}|S_{t+i})$ may be small for some i and bootstrapping may span only a few steps even if n is large

Conclusion

- n-step bootstrapping looks ahead to the next n rewards, states and actions, which generalizes Monte Carlo methods and one-step TD methods

Conclusion

- n-step bootstrapping looks ahead to the next n rewards, states and actions, which generalizes Monte Carlo methods and one-step TD methods
  Advantages
    - Intermediate amount of bootstrapping often works better than the two extremes

Conclusion

- n-step bootstrapping looks ahead to the next n rewards, states and actions, which generalizes Monte Carlo methods and one-step TD methods
  Advantages
  - Intermediate amount of bootstrapping often works better than the two extremes

  Disadvantages
  - Requires a delay of n time steps before updating
  - Requires more computation per time step
  - Requires more memory to store variables from the last n time steps

Conclusion

- n-step bootstrapping looks ahead to the next n rewards, states and actions, which generalizes Monte Carlo methods and one-step TD methods
  Advantages
  - Intermediate amount of bootstrapping often works better than the two extremes

  Disadvantages
  - Requires a delay of n time steps before updating
  - Requires more computation per time step
  - Requires more memory to store variables from the last n time steps
- n-step TD policy evaluation
- On-policy control: n-step Sarsa
- Off-policy control:
  - Importance sampling
  - n-step Tree Backup algorithm