Safe and efficient off-policy reinforcement learning

Remi Munos, Tom Stepleton, Anna Harutyunyan, Marc Bellemare
Two desired properties of a RL algorithm:

- **Off-policy learning**
  - use memory replay
  - do exploration

- **Use multi-steps returns**
  - propagate rewards rapidly
  - avoid accumulation of approximation/estimation errors

Ex: Q-learning (and DQN) is off-policy but does not use multi-steps returns. Policy gradient (and A3C) use returns but are on-policy.

Both properties are important in deepRL. Can we have both simultaneously?
Off-policy reinforcement learning

Behavior policy $\mu(a|x)$, target policy $\pi(a|x)$

Observe trajectory $\{x_0 = x, a_0 = a, r_0, \ldots, x_t, a_t, r_t, \ldots\}$

where $a_t \sim \mu(\cdot|x_t), r_t = r(x_t, a_t)$ and $x_{t+1} \sim p(\cdot|x_t, a_t)$

Goal:
- Policy evaluation: $Q^\pi(x, a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | x_0 = x, a_0 = a, \pi\right]$
- Optimal control: $Q^*(x, a) = \max_{\pi} Q^\pi(x, a)$
Off-policy credit assignment problem

Behavior policy $\mu(a|x)$
Target policy $\pi(a|x)$

Can we use the TD $\delta_t$ to estimate $Q^\pi(x_s, a_s)$ for all $s \leq t$?
Importance sampling

$$\Delta Q(x, a) = \gamma^t \left( \prod_{1 \leq s \leq t} \frac{\pi(a_s | x_s)}{\mu(a_s | x_s)} \right) \delta_t$$

Reweight the trace by the product of IS ratios
Importance sampling

$$\Delta Q(x, a) = \gamma^t \left( \prod_{1 \leq s \leq t} \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)} \right) \delta_t$$

Unbiased estimate of $Q^\pi$
Importance sampling

\[ \Delta Q(x, a) = \gamma^t \left( \prod_{1 \leq s \leq t} \frac{\pi(a_s | x_s)}{\mu(a_s | x_s)} \right) \delta_t \]

Unbiased estimate of \( Q^\pi \)

Large (possibly infinite) variance

$Q^\pi(\lambda)$ algorithm

[Harutyunyan, Bellemare, Stepleton, Munos, 2016]

$$\Delta Q(x, a) = (\gamma \lambda)^t \delta_t$$

Cut traces by a constant $\lambda^t$
$Q^\pi(\lambda)$ algorithm

[Harutyunyan, Bellemare, Stepleton, Munos, 2016]

\[
\Delta Q(x, a) = (\gamma \lambda)^t \delta_t
\]

works if \( \|\pi - \mu\|_1 \leq \frac{1 - \gamma}{\lambda \gamma} \)
**Q^\pi_\lambda** algorithm

[Harutyunyan, Bellemare, Stepleton, Munos, 2016]

\[ \Delta Q(x, a) = (\gamma \lambda)^t \delta_t \]

works if \( \|\pi - \mu\|_1 \leq \frac{1 - \gamma}{\lambda \gamma} \)

may not work otherwise

Not safe!
Tree backup TB(\(\lambda\)) algorithm

[Precup, Sutton, Singh, 2000]

\[\Delta Q(x, a) = \lambda^t \prod_{1 \leq s \leq t} \pi(a_s | x_s) \delta_t\]

Reweight the traces by the product of target probabilities
Tree backup TB(\(\lambda\)) algorithm

[Precup, Sutton, Singh, 2000]

\[
\Delta Q(x, a) = \lambda^t \prod_{1 \leq s \leq t} \pi(a_s | x_s) \delta_t
\]

works for arbitrary policies \(\pi\) and \(\mu\)
Tree backup TB(\(\lambda\)) algorithm

[Precup, Sutton, Singh, 2000]

\[
\Delta Q(x, a) = \lambda^t \prod_{1 \leq s \leq t} \pi(a_s | x_s) \delta_t
\]

- works for arbitrary policies \(\pi\) and \(\mu\)
- cut traces unnecessarily when on-policy

Not efficient!
General off-policy return-based algorithm:

\[
\Delta Q(x, a) = \sum_{t \geq 0} \gamma^t \left( \prod_{1 \leq s \leq t} c_s \right) \left( r_t + \gamma \mathbb{E}_\pi Q(x_{t+1}, \cdot) - Q(x_t, a_t) \right)_{\delta_t}
\]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Trace coefficient:</th>
<th>Problem:</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>( c_s = \frac{\pi(a_s</td>
<td>x_s)}{\mu(a_s</td>
</tr>
<tr>
<td>( Q^\pi(\lambda) )</td>
<td>( c_s = \lambda )</td>
<td>not safe (off-policy)</td>
</tr>
<tr>
<td>( TB(\lambda) )</td>
<td>( c_s = \lambda \pi(a_s</td>
<td>x_s) )</td>
</tr>
</tbody>
</table>
Off-policy policy evaluation:

**Theorem 1:** Assume finite state space. Generate trajectories according to behavior policy $\mu$. Update all states along trajectories according to:

$$Q_{k+1}(x, a) = Q_k(x, a) + \alpha_k \sum_{t>0} \gamma^t (c_1 \ldots c_t) (r_t + \gamma \mathbb{E}_\pi Q_k(x_{t+1}, \cdot) - Q_k(x_t, a_t))$$

Assume all states visited infinitely often. Under usual SA assumptions,

If $0 \leq c_s \leq \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)}$ then $Q_k \rightarrow Q^\pi$ a.s.

Sufficient conditions for a **safe** algorithm (works for any $\mu$ and $\pi$)
Off-policy return-based operator

Lemma:
Assume the traces satisfy $0 \leq c_s \leq \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)}$.

Then the off-policy return-based operator:

$$\mathcal{R}Q(x, a) = Q(x, a) + \mathbb{E}_\mu \left[ \sum_{t \geq 0} \gamma^t (c_1 \ldots c_t) (r_t + \gamma \mathbb{E}_\pi Q(x_{t+1}, \cdot) - Q(x_t, a_t)) \right]$$

is a contraction mapping (whatever $\mu$ and $\pi$) and $Q^\pi$ is its fixed point.
Proof [part 1]

\[ R_Q(x, a) = Q(x, a) + \mathbb{E}_\mu \left[ \sum_{t \geq 0} \gamma^t (c_1 \ldots c_t) (r_t + \gamma \mathbb{E}_\pi Q(x_{t+1}, \cdot) - Q(x_t, a_t)) \right] \]

\[ = \mathbb{E}_\mu \left[ \sum_{t \geq 0} \gamma^t (c_1 \ldots c_t) (r_t + \gamma [\mathbb{E}_\pi Q(x_{t+1}, \cdot) - c_{t+1} Q(x_{t+1}, a_{t+1})]) \right] \]

Thus

\[ (R_{Q_1} - R_{Q_2})(x, a) = \mathbb{E}_\mu \left[ \sum_{t \geq 0} \gamma^{t+1} (c_1 \ldots c_t) \left( \mathbb{E}_\pi (Q_1 - Q_2)(x_{t+1}, \cdot) - c_{t+1} (Q_1 - Q_2)(x_{t+1}, a_{t+1}) \right) \right] \]

\[ = \mathbb{E}_\mu \left[ \sum_{t \geq 0} \gamma^{t+1} (c_1 \ldots c_t) \sum_a \left( \pi(a|x_{t+1}) - \mu(a|x_{t+1}) c_{t+1}(a) \right) (Q_1 - Q_2)(x_{t+1}, a) \right] \]

which is a linear combination weighted by non-negative coefficients which sum to...
Proof [part 2]

Sum of the coeff. \[= \mathbb{E}_\mu \left[ \sum_{t \geq 0} \gamma^{t+1} (c_1 \ldots c_t) \sum_a (\pi(a|x_{t+1}) - \mu(a|x_{t+1})c_{t+1}(a)) \right] \]
\[= \mathbb{E}_\mu \left[ \sum_{t \geq 0} \gamma^{t+1} (c_1 \ldots c_t)(1 - c_{t+1}) \right] \]
\[= \gamma - (1 - \gamma) \mathbb{E}_\mu \left[ \sum_{t \geq 1} \gamma^t (c_1 \ldots c_t) \right] \]
\[\in [0, \gamma] \]

Thus \[\|RQ_1 - RQ_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty \]
Tradeoff for trace coefficients $C_S$

- **Contraction coefficient of the expected operator**

  $$\eta := \gamma - (1 - \gamma) \mathbb{E}_\mu \left[ \sum_{t \geq 1} \gamma^t (c_1 \cdots c_t) \right] \in [0, \gamma]$$

  $\eta = \gamma$ when $C_S = 0$ (one-step Bellman update)

  $\eta = 0$ when $C_S = 1$ (full Monte-Carlo rollouts)

- **Variance of the estimate** (can be infinite for $C_S = \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)}$)

  Large $C_S$ uses multi-steps returns, but large variance

  Small $C_S$ low variance, but do not use multi-steps returns
Our recommendation:

Use \textbf{Retrace}(\lambda) defined by

\[ c_s = \lambda \min \left( 1, \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)} \right) \]

Properties:

- Low variance since \( c_s \leq 1 \)

- Safe (off policy): cut the traces when needed \( c_s \in \left[ 0, \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)} \right] \)

- Efficient (on policy): but only when needed. Note that \( c_s \geq \lambda \pi(a_s|x_s) \)
## Summary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Trace coefficient:</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>( c_s = \frac{\pi(a_s</td>
<td>x_s)}{\mu(a_s</td>
</tr>
<tr>
<td>(Q^\pi(\lambda))</td>
<td>( c_s = \lambda )</td>
<td>not safe (off-policy)</td>
</tr>
<tr>
<td>(TB(\lambda))</td>
<td>( c_s = \lambda \pi(a_s</td>
<td>x_s) )</td>
</tr>
<tr>
<td>Retrace(\lambda)</td>
<td>( c_s = \lambda \min \left(1, \frac{\pi(a_s</td>
<td>x_s)}{\mu(a_s</td>
</tr>
</tbody>
</table>
Retrace(\(\lambda\)) for optimal control

Let \((\mu_k)\) and \((\pi_k)\) sequences of behavior and target policies and

\[
Q_{k+1}(x, a) = Q_k(x, a) + \alpha_k \sum_{t \geq 0} (\lambda \gamma)^t \prod_{1 \leq s \leq t} \min \left( 1, \frac{\pi_k(a_s|x_s)}{\mu_k(a_s|x_s)} \right) \left( r_t + \gamma \mathbb{E}_{\pi} Q_k(x_{t+1}, \cdot) - Q_k(x_t, a_t) \right)
\]

**Theorem 2**

Under previous assumptions (+ a technical assumption)
Assume \((\pi_k)\) are “increasingly greedy” wrt \((Q_k)\)

Then, a.s.,

\[
Q_k \rightarrow Q^*
\]
Remarks

- If \((\pi_k)\) are greedy policies, then \(c_s = \lambda \mathbb{I}\{a_s \in \arg \max_a Q_k(x_s, a)\}\)
  
  \[\rightarrow \text{Convergence of Watkin’s } Q(\lambda) \text{ to } Q^*\]
  (open problem since 1989)

- “Increasingly greedy” allows for smoother traces thus faster convergence

- The behavior policies \((\mu_k)\) do not need to become greedy wrt \((Q_k)\)
  
  \[\rightarrow \text{no GLIE assumption} \text{ (Greedy in the limit with infinite exploration)}\]
  (first return-based algo converging to \(Q^*\) without GLIE)
Retrace for deepRL

Retrace is particularly suitable for deepRL as it is off-policy (useful for exploration and when using memory-replay) and uses multi-steps returns (fast propagation of rewards)

Automatically adjust the length of the return to the degree of “off-policy-ness”

Typical use:
→ **Memory replay** Ex: DQN [Mnih et al., 2015]
  (behavior policy = stored in memory, target policy = current policy)

→ **On-line update** Ex: A3C [Mnih et al., 2016]
  (behavior policy = exploratory policy, target policy = evaluation policy)
Atari 2600 environments: *Retrace vs DQN*

Games:
Asteroids, Defender, Demon Attack, Hero, Krull,
River Raid, Space Invaders, Star Gunner, Wizard of Wor, Zaxxon
Experiments on 60 Atari games

\[ f_a(x) = \frac{1}{60} \left| \{ g : z_{a,g} \geq x \} \right| \]
Comparison of the traces: *Retrace* vs *TB*

**Retrace:**
\[ c_s = \min \left( 1, \frac{\pi(a_s | x_s)}{\mu(a_s | x_s)} \right) \]

**TB:**
\[ c_s = \pi(a_s | x_s) \]
Conclusions

- General update rule for off-policy return-based RL
- Conditions under which an algo is safe and efficient
- We recommend to use Retrace:
  - Converges to $Q^*$
  - Safe: cut the traces when needed
  - Efficient: but only when needed
  - Works for policy evaluation and for control
  - Particularly suited for deepRL

Extensions:
- Works in continuous action spaces
- Can be used in off-policy policy-gradient [Wang et al., 2016]

If you want to know more, come to our poster at NIPS!
References (mentioned in the slides):

- [Harutyunyan, Bellemare, Stepleton, Munos, 2016] $Q(\lambda)$ with Off-Policy Corrections
- [Precup, Sutton, Singh, 2000] Eligibility traces for off-policy policy evaluation
- [Sutton, 1988] Learning to predict by the methods of temporal differences
Temporal credit assignment problem

Behavior policy $\mu(a|x)$

Can a “surprise” $\delta_t$ be used to train $Q(x_s, a_s)$ for all $s \leq t$?
Temporal difference learning TD(\(\lambda\)) [Sutton, 1988]

Behavior policy \(\mu(a|x)\)

\[
\Delta Q(x, a) = (\gamma \lambda)^t \delta_t
\]