Off-Policy Learning (Part 1)

[Safe and Efficient Off-Policy Reinforcement Learning]
Munos, R., Stepleton, T., Harutyunyan, A. and Bellemare, M., NeurIPS 2016

[The Reactor: A fast and sample-efficient Actor-Critic agent for Reinforcement Learning]
Gruslys, A., Azar, M.G., Bellemare, M.G. and Munos, R., ICLR 2018

HaoChih, Lin

2019-April-09

Safe and Efficient Off-Policy Reinforcement Learning

Retrace(λ) is a convergent off-policy multi-step algorithm extending the DQN agent

Safe and Efficient Off-Policy Reinforcement Learning

The Retrace algorithm comes with the theoretical guarantee that in finite state and action spaces, repeatedly updating our current estimate Q produces a sequence of Q functions which converges to Q^π for a fixed π or to Q* if we consider a sequence of policies π which become increasingly greedy w.r.t. the Q estimate

Preliminary (Off-policy)

- Learning the state (action) value function for a policy π:

$$Q^{\pi}(x, a) = \mathbb{E}_{\pi}[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots | x_0 = x, a_0 = a]$$

- You can learn optimal control if it is a greedy policy to the current estimate Q(x; a) e.g. Q-learning
- <u>On-policy:</u> learning from data collected by π
- <u>Off-policy:</u> learning from data collected by $\mu \neq \pi$
- Off-policy methods have advantages:
    - Sample-efficient (e.g. experience replay)
    - Exploration by μ

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Preliminary (Off-policy)

Off-policy Learning

- **Target policy** (π) : deterministic (optimal greedy)
- **Behavior policy** (μ) : stochastic (exploratory)
- **Assumption of coverage:** π(a|s) > 0 implies μ(a|s) > 0

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (Importance Sampling)

Off-policy Learning

- **Target policy** (π) : deterministic (optimal greedy)
- **Behavior policy** (μ) : stochastic (exploratory)
- **Assumption of coverage:** π(a|s) > 0 implies μ(a|s) > 0
- **Importance sampling:**

$$
\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}
$$

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (Importance Sampling)

Off-policy Learning

- **Target policy** (π) : deterministic (optimal greedy)
- **Behavior policy** (μ) : stochastic (exploratory)
- **Assumption of coverage:** π(a|s) > 0 implies μ(a|s) > 0
- **Importance sampling:**

$$
\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}
$$

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

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (Importance Sampling)

Off-policy Learning

- **Target policy** (π) : deterministic (optimal greedy)
- **Behavior policy** (μ) : stochastic (exploratory)
- **Assumption of coverage:** π(a|s) > 0 implies μ(a|s) > 0
- **Importance sampling:**

$$
\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1})\cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}
$$

$$
\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)\cancel{p(S_{k+1}|S_k, A_k)}}{\prod_{k=t}^{T-1} \mu(A_k|S_k)\cancel{p(S_{k+1}|S_k, A_k)}}
$$

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (Importance Sampling)

Off-policy Learning

- **Target policy** (π) : deterministic (optimal greedy)
- **Behavior policy** (μ) : stochastic (exploratory)
- **Assumption of coverage:** π(a|s) > 0 implies μ(a|s) > 0
- **Importance sampling:**

$$
\begin{aligned}
\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&= \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&= \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}
$$

$$
\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)\cancel{p(S_{k+1}|S_k, A_k)}}{\prod_{k=t}^{T-1} \mu(A_k|S_k)\cancel{p(S_{k+1}|S_k, A_k)}} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}.
$$

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (Importance Sampling)

Importance Sampling

- **Usage:**
  - Wanted: the expected returns (values) under the target policy: $v_\pi(S_t)$

Preliminary (Importance Sampling)

Importance Sampling

- **Usage:**
  - Wanted: the expected returns (values) under the target policy: $v_\pi(S_t)$
  - Got: Returns Gt based on the wrong (behavior) policy: $G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

$$\mathbb{E}[G_t|S_t] = v_b(S_t)$$

Preliminary (Importance Sampling)

Importance Sampling

- **Usage:**
  - Wanted: the expected returns (values) under the target policy: $v_\pi(S_t)$
  - Got: Returns Gt based on the wrong (behavior) policy: $G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

    $$\mathbb{E}[G_t|S_t] = v_b(S_t)$$

  - Solution: introduce the importance sampling (for discrepancy correction):

    $$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t] = v_\pi(S_t)$$

    The ratio ϱ(t:T-1) transforms the returns to have the right expected value

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (Importance Sampling)

Importance Sampling

- **Usage:**
  - Wanted: the expected returns (values) under the target policy: $v_\pi(S_t)$
  - Got: Returns Gt based on the wrong (behavior) policy: $G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

$$\mathbb{E}[G_t|S_t] = v_b(S_t)$$

  - Solution: introduce the importance sampling: $\rho_{t:T-1} \doteq \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}.$

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t] = v_\pi(S_t)$$

The ratio ϱ(t:T-1) transforms the returns to have the right expected value

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (Importance Sampling)

Importance Sampling

- **Problem of variances:**
  - Example: an episodes has 100 steps and γ = 0.
    The return from time 0 will then be just G0 = R1

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

Preliminary (Importance Sampling)

Importance Sampling

- **Problem of variances:**
  - Example: an episodes has 100 steps and γ = 0.
    The return from time 0 will then be just G0 = R1
  - Its importance sampling ratio will be a product of 100 factors:

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

$$\frac{\pi(A_0|S_0)}{\mu(A_0|S_0)} \frac{\pi(A_1|S_1)}{\mu(A_1|S_1)} \cdots \frac{\pi(A_{99}|S_{99})}{\mu(A_{99}|S_{99})}$$

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (Importance Sampling)

Importance Sampling

- **Problem of variances:**
  - Example: an episodes has 100 steps and γ = 0. The return from time 0 will then be just G0 = R1

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

  - Its importance sampling ratio will be a product of 100 factors:

$$\frac{\pi(A_0|S_0)}{\mu(A_0|S_0)} \frac{\pi(A_1|S_1)}{\mu(A_1|S_1)} \cdots \frac{\pi(A_{99}|S_{99})}{\mu(A_{99}|S_{99})}$$

  - But it is really only necessary to scale by the first factor. The other 99 factors are irrelevant, but they add enormously to its variance.

Preliminary (N-steps Returns)

N-step TD Prediction

- Monte Carlo Return:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T,$$

- One Step Return:

$$G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1}),$$

- N steps Return:

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

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (N-steps Returns)

N-step TD Prediction

- Monte Carlo Return:

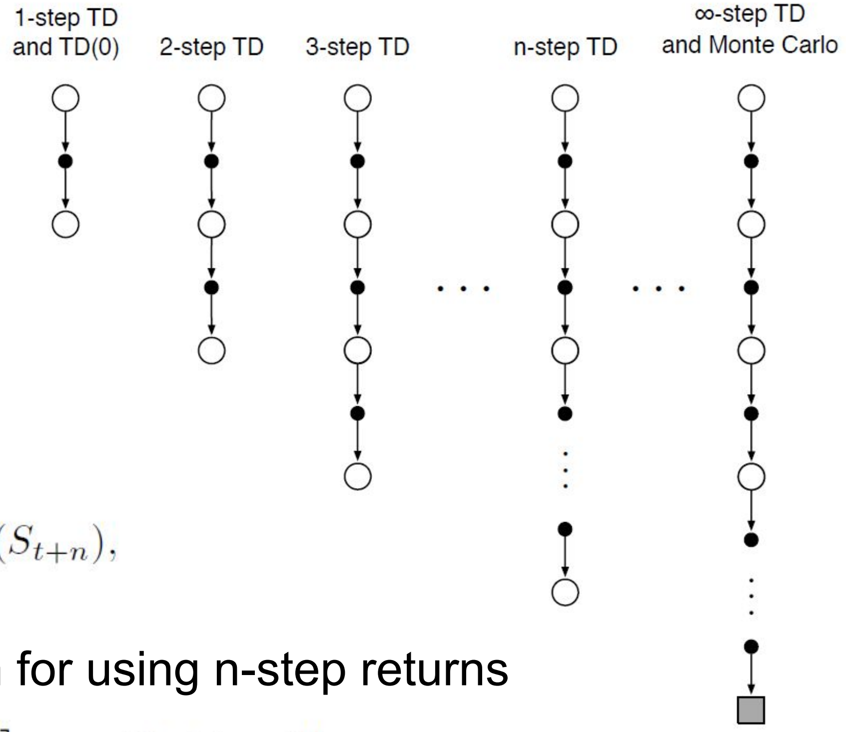
$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T,$$

- One Step Return:

$$G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1}),$$

- N steps Return:

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



- The natural state-value learning algorithm for using n-step returns

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\left[G_{t:t+n} - V_{t+n-1}(S_t)\right], \qquad 0 \le t < T,$$

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (λ-steps Returns)

The λ-return

An alternative way of moving smoothly between Monte Carlo and one-step TD methods

$$G_t^\lambda \doteq (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}.$$

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_{t:t+n} + \boxed{\lambda^{T-t-1}G_t,}$$

Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

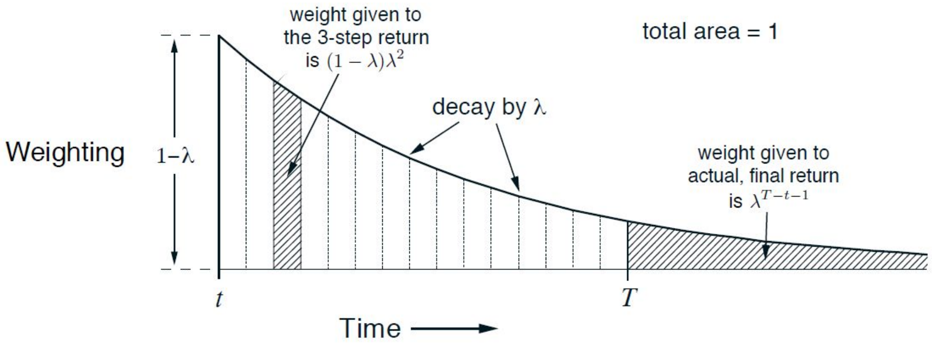
Preliminary (λ-steps Returns)

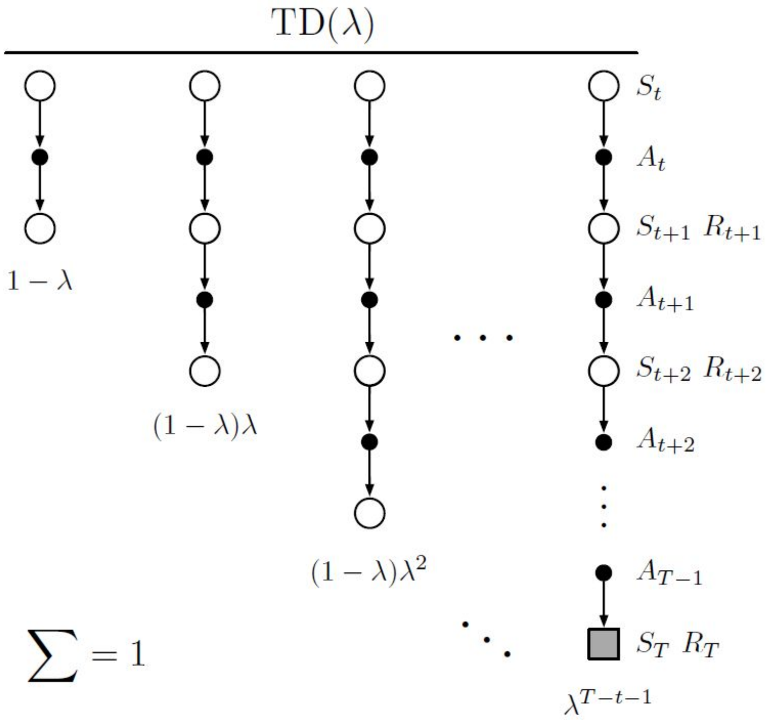
The λ-return

An alternative way of moving smoothly between Monte Carlo and one-step TD methods

$$G_t^\lambda \doteq (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}.$$

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_{t:t+n} + \boxed{\lambda^{T-t-1}G_t},$$





Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Preliminary (λ-steps Returns)

The λ-return

- The λ-return could be written as:

$$R_t^\lambda = \underbrace{(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}R_t^{(n)}}_{\text{Until termination}} + \underbrace{\lambda^{T-t-1}R_t}_{\text{After termination}}$$

- If <u>λ = 1</u>, you get MC return:

$$R_t^\lambda = (1-1)\sum_{n=1}^{T-t-1}1^{n-1}R_t^{(n)} + 1^{T-t-1}R_t = R_t$$

- If <u>λ = 0</u>, you get TD(0):

$$R_t^\lambda = (1-0)\sum_{n=1}^{T-t-1}0^{n-1}R_t^{(n)} + 0^{T-t-1}R_t = R_t^{(1)}$$

Ref: http://www-anw.cs.umass.edu/~barto/courses/cs687/Chapter%207.pdf
Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Preliminary (λ-steps Returns)

The λ-return

- The λ-return could be written as:

$$R_t^\lambda = \underbrace{(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}R_t^{(n)}}_{\text{Until termination}} + \underbrace{\lambda^{T-t-1}R_t}_{\text{After termination}}$$

- If <u>λ = 1</u>, you get MC return:

$$R_t^\lambda = (1-1)\sum_{n=1}^{T-t-1}1^{n-1}R_t^{(n)} + 1^{T-t-1}R_t = R_t$$

- If <u>λ = 0,</u> you get TD(0):

$$R_t^\lambda = (1-0)\sum_{n=1}^{T-t-1}0^{n-1}R_t^{(n)} + 0^{T-t-1}R_t = R_t^{(1)}$$

Questions:

- Can we apply to Q learning?
  - <u>Policy evaluation</u>:
    estimate Q^π from samples collected by μ
  - <u>Control</u>:
    estimate Q* from samples collected by μ
- Possible solution
  - Watkins's Q(λ) [Watkins 1989] method
  - Cut off traces whenever a non-greedy action is taken
  - Converges to Q* under a mild assumption (first proved in Retrace paper)

Ref: http://www-anw.cs.umass.edu/~barto/courses/cs687/Chapter%207.pdf
Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Preliminary (λ-steps Returns)

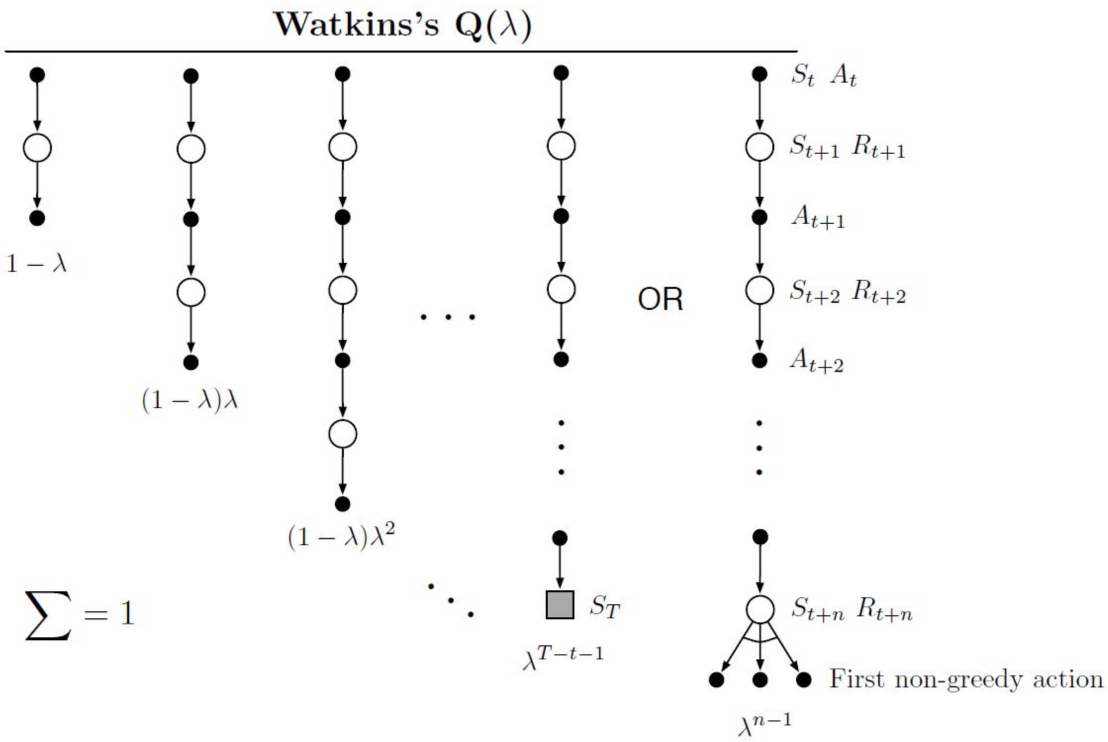
Watkins's Q(λ)

Classic multi-step algorithm for off-policy control

This approach is an off-policy eligibility trace which updates more than one Q-value per step.

This can result in a significant increase in the speed of learning at a cost to stability

unproven of convergence until <u>Retrace </u>(2016, ~30 years)



Ref: Sutton, R.S. and Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Safe and Efficient Off-Policy Reinforcement Learning

Retrace

Safe and Efficient Off-Policy Reinforcement Learning

- Proposes a new off-policy multi-step RL method: **Retrace(λ)**

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Safe and Efficient Off-Policy Reinforcement Learning

- Proposes a new off-policy multi-step RL method: **Retrace(λ)**
- Theoretical advantages
  - It converges for any π, μ (**safe**)
  - It makes the best use of samples if π and μ are close to each other (**efficient**)
  - Its variance is lower than importance sampling

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Safe and Efficient Off-Policy Reinforcement Learning

- Proposes a new off-policy multi-step RL method: **Retrace(λ)**
- Theoretical advantages
  - It converges for any π, μ (**safe**)
  - It makes the best use of samples if π and μ are close to each other (**efficient**)
  - Its variance is lower than importance sampling
- Empirical evaluation
  - On Atari 2600, it beats one-step Q-learning (DQN) and the existing multi-step methods (Q*(λ), Tree-Backup)

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Safe and Efficient Off-Policy Reinforcement Learning

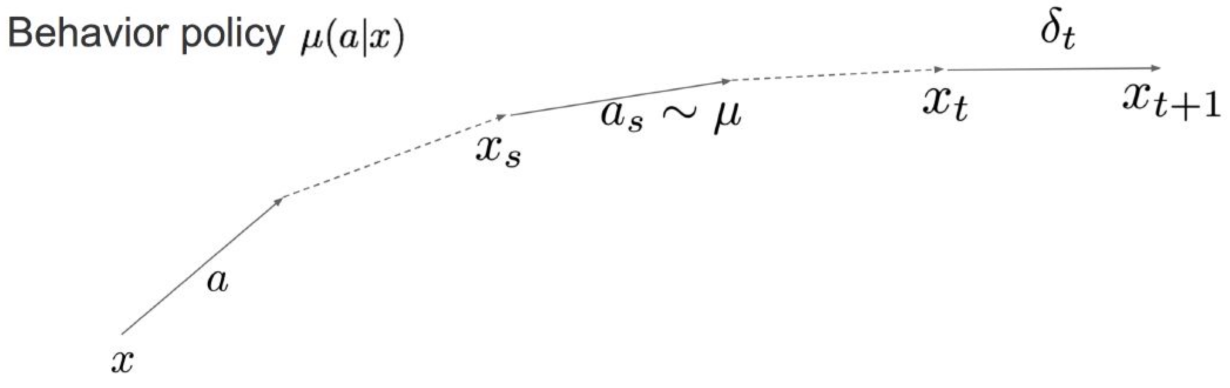
- Proposes a new off-policy multi-step RL method: **Retrace(λ)**
- Theoretical advantages
  - It converges for any π, μ (**safe**)
  - It makes the best use of samples if π and μ are close to each other (**efficient**)
  - Its variance is lower than importance sampling
- Empirical evaluation
  - On Atari 2600, it beats one-step Q-learning (DQN) and the existing multi-step methods (Q*(λ), Tree-Backup)
- Proves the convergence of Watkins's Q(λ) for the first time

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

On-policy multi-step methods

TD(λ)



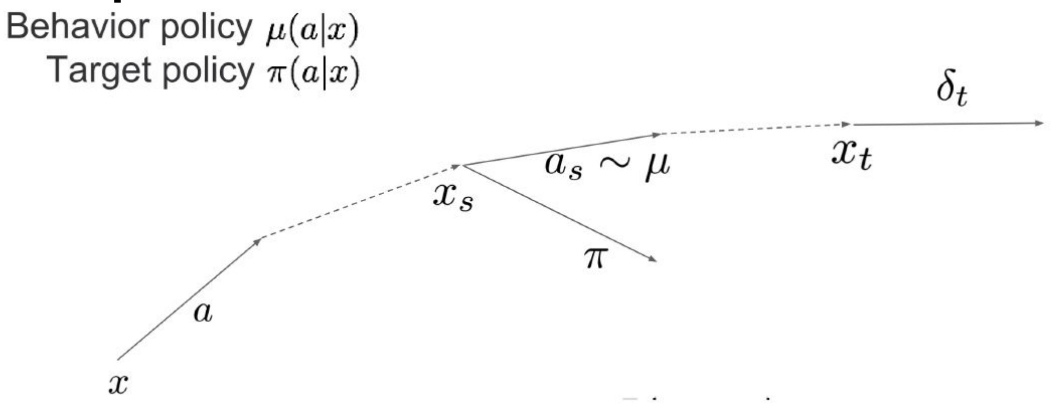
From the presentation by the authors: https://ewrl.files.wordpress.com/2016/12/munos.pdf

- A popular multi-step algorithm for on-policy policy evaluation
- $\Delta_t Q(x, a) = (\gamma\lambda)^t \delta_t$ where λ ∈ [0,1] is chosen to balance bias and variance
- Multi-step methods have advantages:
  - Rewards are propagated rapidly
  - Bias introduced by bootstrapping is reduced

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Off-policy multi-step methods

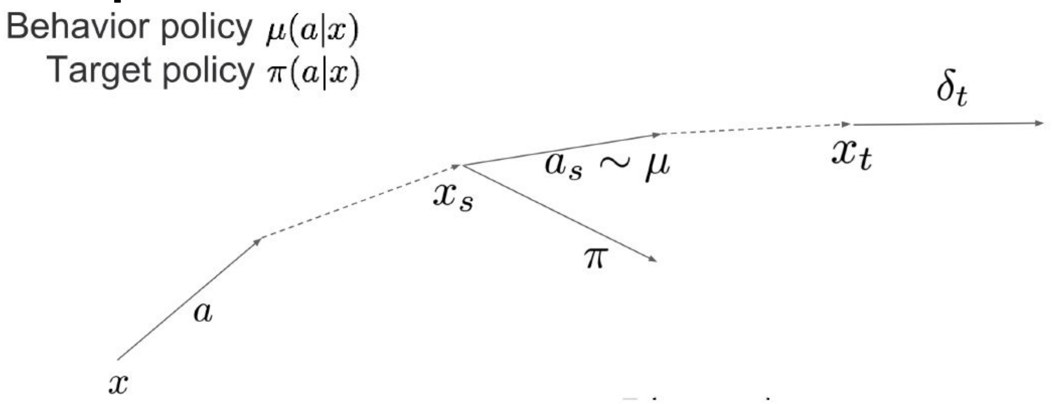


From the presentation by the authors: https://ewrl.files.wordpress.com/2016/12/munos.pdf

- $\delta_t = r_t + \gamma \mathbb{E}_{\pi} Q(x_{t+1}, \cdot) - Q(x_t, a_t)$
- Can you use δ_t to estimate $Q^{\pi}(x_t, a_t)$ for all s ≤ t?
  - Three methods mentioned in the paper:

Retrace

Off-policy multi-step methods



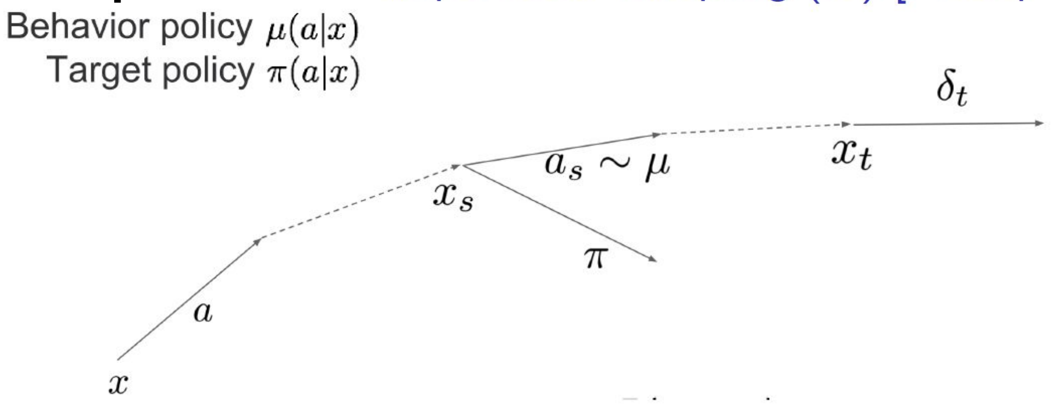
From the presentation by the authors: https://ewrl.files.wordpress.com/2016/12/munos.pdf

- $\delta_t = r_t + \gamma \mathbb{E}_\pi Q(x_{t+1}, \cdot) - Q(x_t, a_t)$
- Can you use δ_t to estimate $Q^\pi(x_t, a_t)$ for all s ≤ t?
  - Three methods mentioned in the paper:
    Importance Sampling (IS) [Precup et al. 2000] $Q^\pi(\lambda)$ [Harutyunyan et al. 2016]
    Tree-Backup (TB) [Precup et al. 2000]

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Off-policy multi-step methods: Importance Sampling (IS) [Precup et al. 2000]



From the presentation by the authors: https://ewrl.files.wordpress.com/2016/12/munos.pdf

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

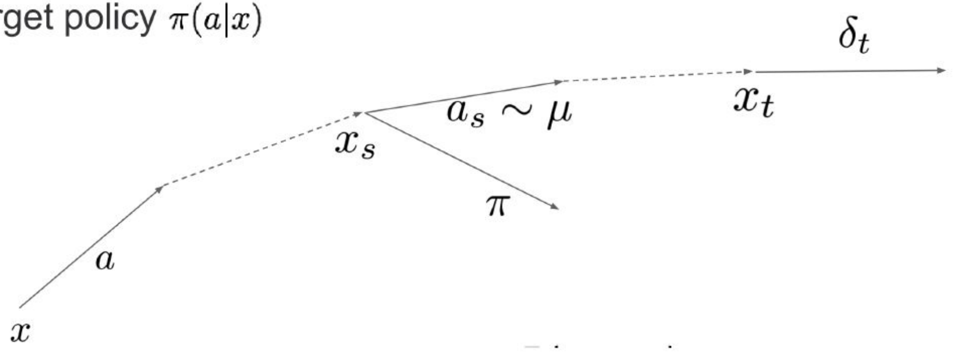
- <u>Pros:</u> Unbiased estimate of Q^π
- <u>Cons:</u> Large variance since $\frac{\pi(a_s|x_s)}{\mu(a_s|x_s)}$ is not bounded

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Off-policy multi-step methods: Importance Sampling (IS) [Precup et al. 2000]

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



From the presentation by the authors: https://ewrl.files.wordpress.com/2016/12/munos.pdf

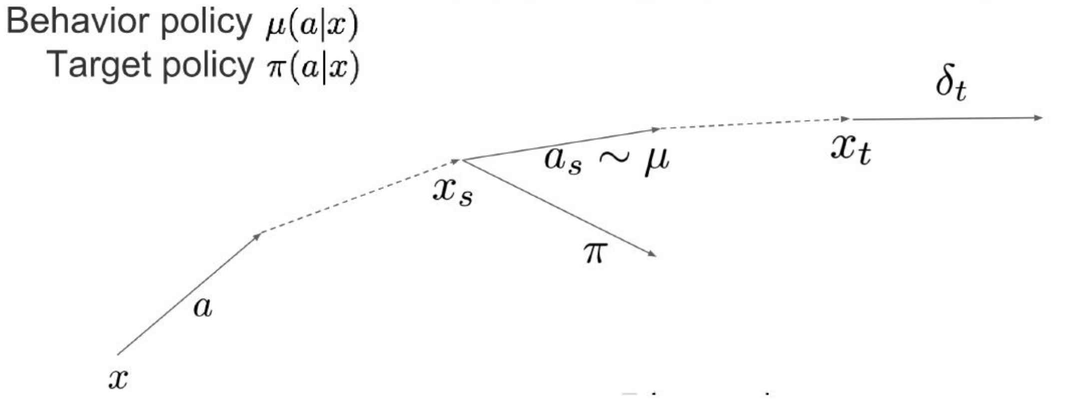
- $\Delta_t Q(x,a) = \gamma^t \left(\prod_{1 \le s \le t} \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)}\right)\delta_t$ **Reweight the trace by the product of IS ratios**

- Pros: Unbiased estimate of Q^{π}
- Cons: Large variance since $\frac{\pi(a_s|x_s)}{\mu(a_s|x_s)}$ is not bounded (not efficient)

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Off-policy multi-step methods: $Q^{\pi}(\lambda)$ [Harutyunyan et al. 2016]



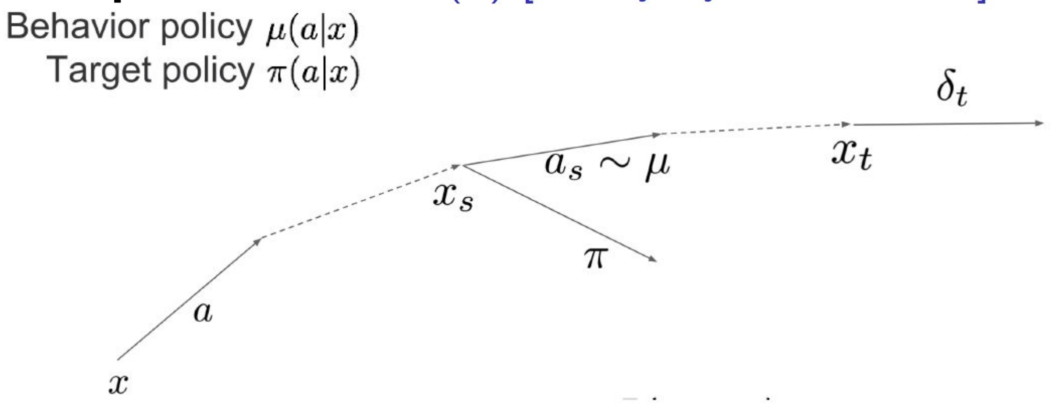
From the presentation by the authors: https://ewrl.files.wordpress.com/2016/12/munos.pdf

- $\Delta_t Q(x, a) = (\gamma\lambda)^t \delta_t$
- Pros: Convergent if π and μ are sufficiently close to each other or λ is sufficiently small: $\lambda < \frac{1-\gamma}{\gamma\epsilon}$, where $\epsilon := max_x \|\pi(\cdot|x) - \mu(\cdot|x)\|_1$
- Cons: Not convergent otherwise

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Off-policy multi-step methods: $Q^{\pi}(\lambda)$ [Harutyunyan et al. 2016]



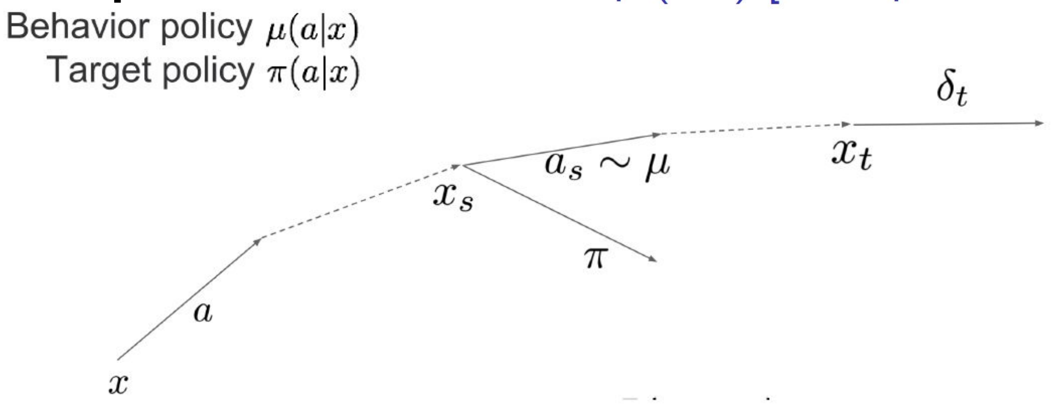
From the presentation by the authors: https://ewrl.files.wordpress.com/2016/12/munos.pdf

- $\Delta_t Q(x, a) = (\gamma\lambda)^t \delta_t$ **Cut traces by a constant λ^t**
- Pros: Convergent if π and μ are sufficiently close to each other or λ is sufficiently small: $\lambda < \frac{1-\gamma}{\gamma\epsilon}$, where $\epsilon := max_x \|\pi(\cdot|x) - \mu(\cdot|x)\|_1$
- Cons: Not convergent otherwise (not safe)

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Off-policy multi-step methods: Tree-Backup (TB) [Precup et al. 2000]



From the presentation by the authors: https://ewrl.files.wordpress.com/2016/12/munos.pdf

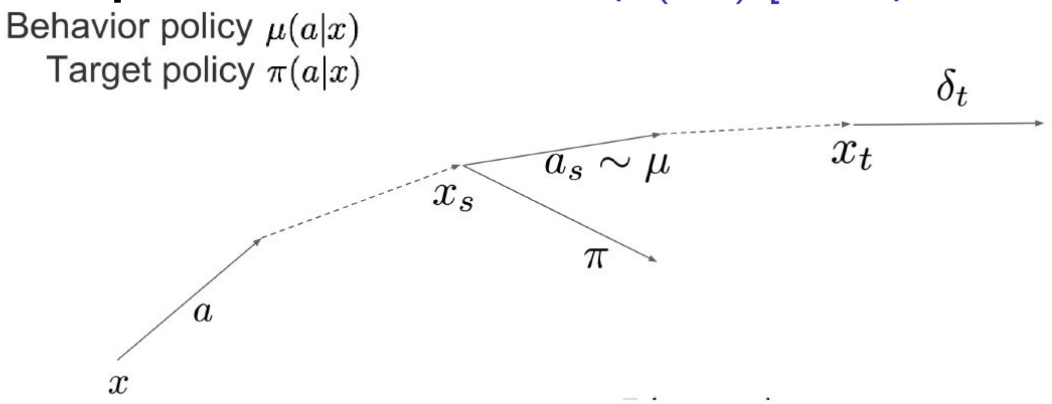
- $\Delta_t Q(x,a) = (\gamma\lambda)^t (\prod_{1 \le s \le t} \pi(a_s|x_s))\delta_t$

- Pros: Convergent for any π and μ. even if μ is unknown and/or non-Markov
- Cons: $\prod_{1 \le s \le t} \pi(a_s|x_s)$ decays rapidly when near on-policy

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Off-policy multi-step methods: Tree-Backup (TB) [Precup et al. 2000]



From the presentation by the authors: https://ewrl.files.wordpress.com/2016/12/munos.pdf

- $\Delta_t Q(x, a) = (\gamma\lambda)^t (\prod_{1 \leq s \leq t} \pi(a_s|x_s))\delta_t$

Reweight the traces by the product of target probabilities

- Pros: Convergent for any π and μ. even if μ is unknown and/or non-Markov
- Cons: $\prod_{1 \leq s \leq t} \pi(a_s|x_s)$ decays rapidly when near on-policy (not efficient)

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

General off-policy return-based algorithm

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

Retrace

General off-policy return-based algorithm

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

	Definition of c_s	Estimation variance	Guaranteed convergence†	Use full returns (near on-policy)
Importance sampling	$\frac{\pi(a_s\|x_s)}{\mu(a_s\|x_s)}$	High	for any π, μ	yes
$Q^\pi(\lambda)$	λ	Low	for π close to μ	yes
TB(λ)	$\lambda\pi(a_s\|x_s)$	Low	for any π, μ	no

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

General off-policy return-based algorithm

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

	Definition of c_s	Estimation variance	Guaranteed convergence†	Use full returns (near on-policy)
Importance sampling	$\frac{\pi(a_s\|x_s)}{\mu(a_s\|x_s)}$	High	for any π, μ	yes
$Q^\pi(\lambda)$	λ	Low	for π close to μ	yes
TB(λ)	$\lambda\pi(a_s\|x_s)$	Low	for any π, μ	no

- None of the existing methods is perfect (low variance, safe and efficient)
  - Safe: i.e. convergent for any π and μ (Q(λ))
  - Efficient: i.e. using full returns when on-policy (Tree-Backup)

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Proposed Solution: Retrace(λ)

Ref: Remi Munos DeepMind, https://project.inria.fr/paiss/files/2018/07/munos-off-policy-dRL.pdf

Retrace

Proposed Solution: Retrace(λ)

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

Retrace

Proposed Solution: Retrace(λ)

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

Properties:

- Low variance: since $c_s \leq 1$
- Safe (off policy): cut the traces when needed $c_s \in \left[0, \frac{\pi(a_s|x_s)}{\mu(a_a|x_s)}\right]$
- Efficient (on policy): keep the traces near on policy. Note that $c_s \geq \lambda\pi(a_s|x_s)$

Ref: Remi Munos DeepMind, https://project.inria.fr/paiss/files/2018/07/munos-off-policy-dRL.pdf

Retrace

Proposed Solution: Retrace(λ)

	Definition of c_s	Estimation variance	Guaranteed convergence†	Use full returns (near on-policy)
Importance sampling	$\frac{\pi(a_s\|x_s)}{\mu(a_s\|x_s)}$	High	for any π, μ	yes
$Q^\pi(\lambda)$	λ	Low	for π close to μ	yes
TB(λ)	$\lambda\pi(a_s\|x_s)$	Low	for any π, μ	no
Retrace(λ)	$\lambda \min\left(1, \frac{\pi(a_s\|x_s)}{\mu(a_s\|x_s)}\right)$	Low	for any π, μ	yes

Ref: Remi Munos DeepMind, https://project.inria.fr/paiss/files/2018/07/munos-off-policy-dRL.pdf

Retrace

Off-policy policy evaluation

Theorem-1: Assume finite state space. Generate trajectories according to behavior policy μ. Update all states along trajectories according to

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

Ref: Remi Munos DeepMind, https://project.inria.fr/paiss/files/2018/07/munos-off-policy-dRL.pdf

Retrace

Off-policy policy evaluation

Theorem-1: Assume finite state space. Generate trajectories according to behavior policy μ. Update all states along trajectories according to

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

Assume all states visited infinitely often.

Retrace

Off-policy policy evaluation

Theorem-1: Assume finite state space. Generate trajectories according to behavior policy μ. Update all states along trajectories according to

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

Assume all states visited infinitely often.

$$\text{If} \quad 0 \leq c_s \leq \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)} \quad \text{then} \quad Q_k \to Q^\pi$$

Sufficient conditions for a safe algorithm (works for any π and μ)

Retrace

Tradeoff for trace coefficients c_s

- Contraction coefficient of the expected operator

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

  - $\eta = \gamma$ 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 $c_s = \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)}$ case)
  - Large c_s : uses multi-steps returns, but large variance
  - Small c_s : low variance, but do not use multi-steps returns

Ref: Remi Munos DeepMind, https://project.inria.fr/paiss/files/2018/07/munos-off-policy-dRL.pdf

Retrace

Retrace(λ) for optimal control

Let (μ_k) and (π_k) sequences of behavior and target policies and:

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

Ref: Remi Munos DeepMind, https://project.inria.fr/paiss/files/2018/07/munos-off-policy-dRL.pdf

Retrace

Retrace(λ) for optimal control

Let (μ_k) and (π_k) sequences of behavior and target policies and:

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

<u>Theorem 2</u>

Under previous assumptions

Assume (π_k) are “increasingly greedy” wrt (Q_k)

Then, a.s., $\boxed{Q_k \to Q^*}$

Ref: Remi Munos DeepMind, https://project.inria.fr/paiss/files/2018/07/munos-off-policy-dRL.pdf

Retrace

Remarks

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

Under assumption of finite-state space:

- Convergence to optimal policy
- Cut traces when -and only when- needed
- Adjust the length of the backup to the "off-policy-ness" of the data

Ref: Remi Munos DeepMind, https://project.inria.fr/paiss/files/2018/07/munos-off-policy-dRL.pdf

Retrace

Retrace for deep RL

Several actor-critic architectures at DeepMind:

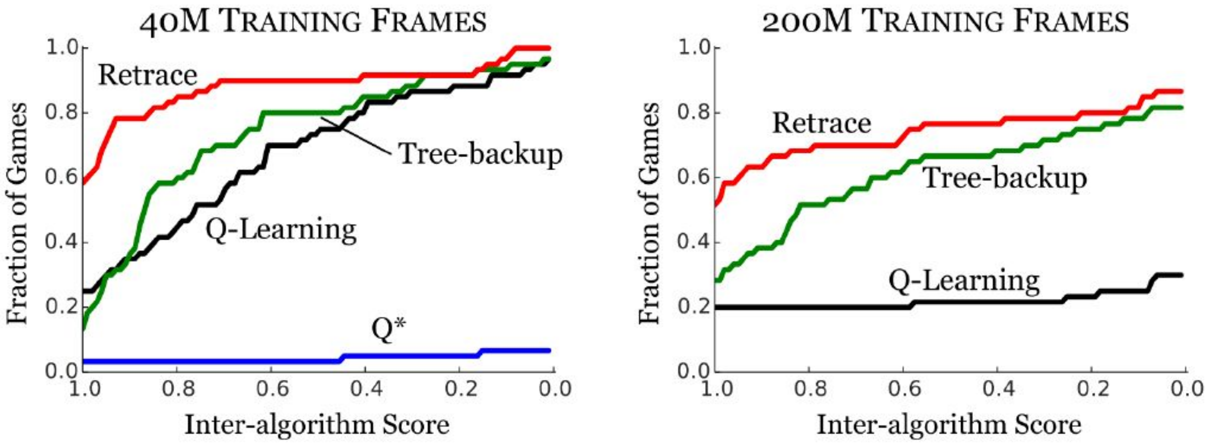
- **ACER** (Actor-Critic for Experience Replay) [Wang et al., 2017]. Policy gradient. Works for continuous actions.
- **Reactor** (Retrace-actor) [Gruesly et al., 2018]. Use beta-LOO to update policy. Use LSTM.
- **MPO** (Maximum a posteriori Policy Optimization) [Abdolmaleki et al., 2018] Soft (KL-regularized) policy improvement.
- **IMPALA** (IMPortance Weighted Actor-Learner Architecture) [Espeholt et al., 2018]. Heavily distributed agent. Uses V-trace.

Ref: Remi Munos DeepMind, https://project.inria.fr/paiss/files/2018/07/munos-off-policy-dRL.pdf

Retrace

Evaluation on Atari 2600

$$f_a(x) = \frac{1}{60} \left| \{ g : z_{a,g} \geq x \} \right|$$

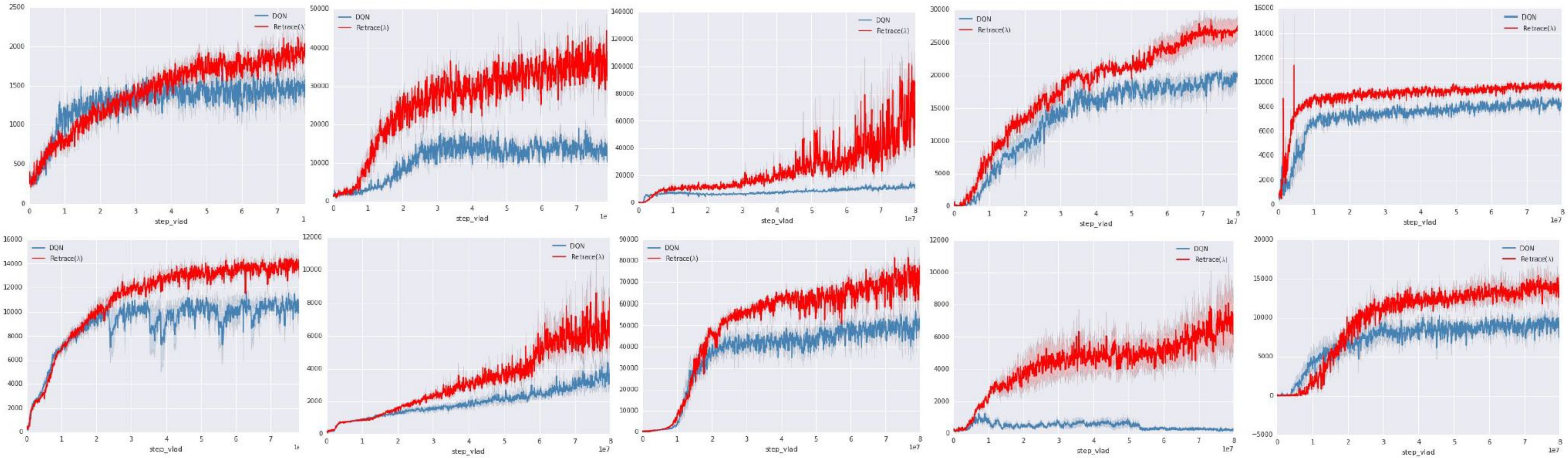


- <u>Performance comparison:</u>
  - Inter-algorithm scores are normalized so that 0 and 1 respectively correspond to the worst and best scores for a particular game (Roughly, a strictly higher curve corresponds to a better algorithm)
  - Retrace(λ) performs best on 30 out of 60 games

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Evaluation on Atari 2600: Retrace vs DQN



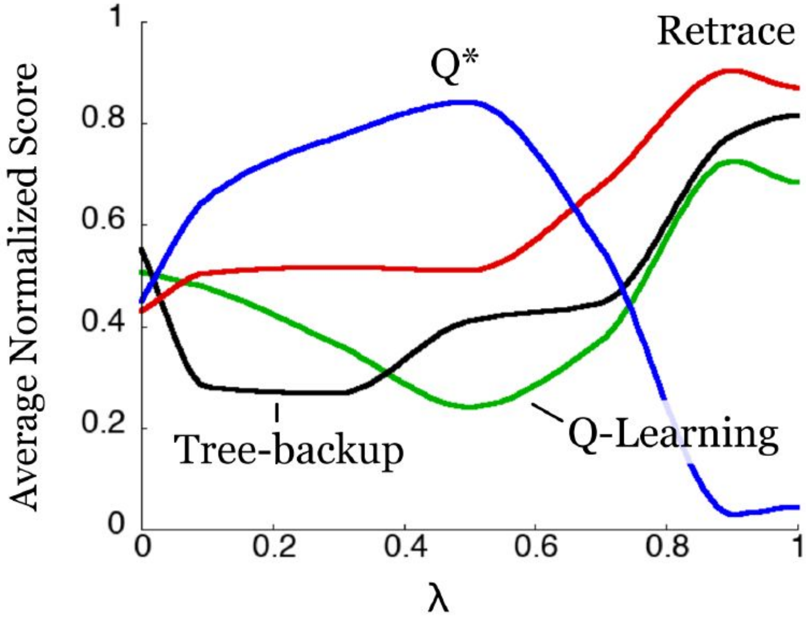
<u>Games:</u> (Blue: DQN Red: Retrace)

Asteroids, Defender, Demon Attack, Hero, Krull, River Raid, Space Invaders, Star Gunner, Wizard of Wor, Zaxxon

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

Evaluation on Atari 2600



- <u>Sensitivity to the value of λ:</u>
    - Retrace(λ) is robust and consistently outperforms Tree-Backup
    - Q* performs best for small values of λ
    - Note that the Q-learning scores are fixed across different λ

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

Retrace

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* (finite state/action space, policy π is increasingly greed)
  - Safe: cut the traces when needed
  - Efficient: but only when needed
  - Works for policy evaluation and for control
  - Particularly suited for deep RL
- Extensions:
  - Works in continuous action spaces
  - Can be used in off-policy policy-gradient [Wang et al., 2016]

Ref: slides from Yasuhiro Fujita, Preferred Networks Inc.

A fast and sample-efficient Actor-Critic agent for Reinforcement Learning (Reactor)

A fast and sample-efficient Actor-Critic agent for Reinforcement Learning (Reactor)

[Contributions]

- Sample-efficiency:
  Higher than Prioritized Dueling DQN (Wang et al., 2017) and Categorical DQN (Bellemare et al., 2017)
- Time-efficiency:
  Better run-time performance than A3C (Mnih et al., 2016).

A fast and sample-efficient Actor-Critic agent for Reinforcement Learning (Reactor)

[Contributions]

- Sample-efficiency:
  Higher than Prioritized Dueling DQN (Wang et al., 2017) and Categorical DQN (Bellemare et al., 2017)
- Time-efficiency:
  Better run-time performance than A3C (Mnih et al., 2016).

[Reactor (Retrace-Actor)]

Combining the sample-efficiency of off-policy experience replay with the time-efficiency of asynchronous algorithms

Reactor

The Reactor is a combination of four novel contributions on top of recent improvements to both deep value-based RL and policy-gradient algorithms.

- β-leave-one-out:
  Improves the trade-off between variance and bias by using action values as a baseline.

- Distributional Retrace:
  Brings multi-step off-policy updates to the distributional reinforcement learning setting

- Prioritized sequences replay:
  Present the lazy initialization for more efficient replay prioritization.

- Agent Architecture:
  Propose an optimized network and parallel training architecture

Reactor

β-leave-one-out

- Need a policy gradient algorithm to train the actor policy π based on current estimate $Q(x, a)$ of $Q^{\pi}(x, a)$:

$$\nabla V^{\pi}(x_0) = \mathbb{E}\left[\sum_t \gamma^t \sum_a Q^{\pi}(x_t, a) \nabla \pi(a|x_t)\right]$$

Reactor

β-leave-one-out

- Need a policy gradient algorithm to train the actor policy π based on current estimate $Q(x,a)$ of $Q^{\pi}(x,a)$:

$$\nabla V^{\pi}(x_0) = \mathbb{E}\left[\sum_t \gamma^t \sum_a Q^{\pi}(x_t, a) \nabla \pi(a|x_t)\right].$$

- Simplify the notations (find a way to estimate gradient G):

$$G = \sum_a Q^{\pi}(a) \nabla \pi(a).$$

Reactor

β-leave-one-out

- Need a policy gradient algorithm to train the actor policy π based on current estimate $Q(x,a)$ of $Q^\pi(x,a)$:

$$\nabla V^\pi(x_0) = \mathbb{E}\left[\sum_t \gamma^t \sum_a Q^\pi(x_t,a)\nabla\pi(a|x_t)\right].$$

- Simplify the notations (find a way to estimate gradient G):

$$G = \sum_a Q^\pi(a)\nabla\pi(a).$$

- Unbiased estimate of G (sampled from behaviour policy μ with IS ratio):

$$\hat{G}_{\mathrm{ISLR}} = \frac{\pi(\hat{a})}{\mu(\hat{a})}(R(\hat{a}) - V)\nabla \log \pi(\hat{a})$$

Reactor

β-leave-one-out

- Need a policy gradient algorithm to train the actor policy π based on current estimate $Q(x,a)$ of $Q^\pi(x,a)$:

$$\nabla V^\pi(x_0) = \mathbb{E}\left[\sum_t \gamma^t \sum_a Q^\pi(x_t,a)\nabla\pi(a|x_t)\right].$$

- Simplify the notations (find a way to estimate gradient G):

$$G = \sum_a Q^\pi(a)\nabla\pi(a).$$

- Unbiased estimate of G (sampled from behaviour policy μ with IS ratio):

$$\hat{G}_{\mathrm{ISLR}} = \frac{\pi(\hat{a})}{\mu(\hat{a})}(R(\hat{a}) - V)\nabla \log \pi(\hat{a})$$

Baseline depends on the state

Reactor

β-leave-one-out

- Need a policy gradient algorithm to train the actor policy π based on current estimate $Q(x,a)$ of $Q^{\pi}(x,a)$:

$$\nabla V^{\pi}(x_0) = \mathbb{E}\left[\sum_t \gamma^t \sum_a Q^{\pi}(x_t,a)\nabla\pi(a|x_t)\right].$$

- Simplify the notations (find a way to estimate gradient G):

$$G = \sum_a Q^{\pi}(a)\nabla\pi(a).$$

- Unbiased estimate of G (sampled from behaviour policy μ with IS ratio):

$$\hat{G}_{\text{ISLR}} = \frac{\pi(\hat{a})}{\mu(\hat{a})}(R(\hat{a}) - V)\nabla \log \pi(\hat{a})$$

Unbiased, but high variance, needs reducing!

Reactor

β-leave-one-out

- leave-one-out (LOO) estimate of G:
  Instead of applying IS, estimate G directly from the return R(a) for the chosen action a and current estimate $Q(x, a)$ of $Q^{\pi}(x, a)$

$$\hat{G}_{\text{LOO}} = R(\hat{a})\nabla\pi(\hat{a}) + \sum_{a \neq \hat{a}} Q(a)\nabla\pi(a).$$

Reactor

β-leave-one-out

- leave-one-out (LOO) estimate of G:
  Instead of applying IS, estimate G directly from the return R(a) for the chosen action a and current estimate $Q(x, a)$ of $Q^{\pi}(x, a)$

$$\hat{G}_{\text{LOO}} = R(\hat{a})\nabla\pi(\hat{a}) + \sum_{a \neq \hat{a}} Q(a)\nabla\pi(a).$$

Low variance

Reactor

β-leave-one-out

- leave-one-out (LOO) estimate of G:
  Instead of applying IS, estimate G directly from the return R(a) for the chosen action a and current estimate $Q(x, a)$ of $Q^\pi(x, a)$

$$\hat{G}_{\text{LOO}} = R(\hat{a})\nabla\pi(\hat{a}) + \sum_{a \neq \hat{a}} Q(a)\nabla\pi(a).$$

but may be biased if the estimated $Q(x, a)$ values differ from $Q^\pi(x, a)$

Reactor

β-leave-one-out

- leave-one-out (LOO) estimate of G:
  Instead of applying IS, estimate G directly from the return R(a) for the chosen action a and current estimate $Q(x,a)$ of $Q^{\pi}(x,a)$

$$\hat{G}_{\text{LOO}} = R(\hat{a})\nabla\pi(\hat{a}) + \sum_{a\neq\hat{a}} Q(a)\nabla\pi(a).$$

- A better bias-variance tradeoff --> β-LOO policy-gradient estimate:

Reactor

β-leave-one-out

- leave-one-out (LOO) estimate of G:
  Instead of applying IS, estimate G directly from the return R(a) for the chosen action a and current estimate $Q(x,a)$ of $Q^{\pi}(x,a)$

$$\hat{G}_{\mathrm{LOO}} = R(\hat{a})\nabla\pi(\hat{a}) + \sum_{a\neq\hat{a}} Q(a)\nabla\pi(a).$$

- A better bias-variance tradeoff --> β-LOO policy-gradient estimate:

$$\hat{G}_{\beta\text{-LOO}} = \beta(R(\hat{a}) - Q(\hat{a}))\nabla\pi(\hat{a}) + \sum_{a} Q(a)\nabla\pi(a),$$

Reactor

β-leave-one-out

- leave-one-out (LOO) estimate of G:
  Instead of applying IS, estimate G directly from the return R(a) for the chosen action a and current estimate $Q(x, a)$ of $Q^{\pi}(x, a)$

$$\hat{G}_{\text{LOO}} = R(\hat{a})\nabla\pi(\hat{a}) + \sum_{a \neq \hat{a}} Q(a)\nabla\pi(a).$$

- A better bias-variance tradeoff --> β-LOO policy-gradient estimate:

$$\hat{G}_{\beta\text{-LOO}} = \beta(R(\hat{a}) - Q(\hat{a}))\nabla\pi(\hat{a}) + \sum_{a} Q(a)\nabla\pi(a),$$

  where β = β(μ, π, a) can be a function of both policies, π and μ, and the selected action a

Reactor

β-leave-one-out

- Property of β-LOO for given β:

  General case: $\hat{G}_{\beta\text{-LOO}} = \beta(R(\hat{a}) - Q(\hat{a}))\nabla\pi(\hat{a}) + \sum_a Q(a)\nabla\pi(a),$

Reactor

β-leave-one-out

- Property of β-LOO for given β:

When β = 1: $$\hat{G}_{\text{LOO}} = R(\hat{a})\nabla\pi(\hat{a}) + \textstyle\sum_{a\neq\hat{a}} Q(a)\nabla\pi(a).$$

Reactor

β-leave-one-out

- Property of β-LOO for given β:

  When β = 1/μ: $$\hat{G}_{\frac{1}{\mu}\text{-LOO}} = \frac{\pi(\hat{a})}{\mu(\hat{a})}(R(\hat{a}) - Q(\hat{a}))\nabla \log \pi(\hat{a}) + \textstyle\sum_a Q(a)\nabla\pi(a).$$

Reactor

β-leave-one-out

- Property of β-LOO for given β:

  When β = 1/μ: $\hat{G}_{\frac{1}{\mu}\text{-LOO}} = \frac{\pi(\hat{a})}{\mu(\hat{a})}(R(\hat{a}) - Q(\hat{a}))\nabla \log \pi(\hat{a}) + \sum_a Q(a)\nabla\pi(a).$

- Choice of β:
  - <u>Low bias</u>: as β(a) is close to 1/μ(a) or $Q(x, a)$ close to $Q^\pi(x, a)$.
  - <u>Unbiased</u>: as β(a) is equal to 1/μ(a)
  - <u>Low Variance</u>: as β(a) is small

- Bias-Variance tradeoff:
  - Choose $\beta(\hat{a}) = \min\left(c, \frac{1}{\mu(\hat{a})}\right)$ for some constant c ≥ 1

Reactor

Distributional Retrace

Extend C51 to multi-step Bellman backup.

- The n-step distributional Bellman target:

$$\sum_i q_i(x_{t+n}, a)\delta_{z_i^n}, \text{ with } z_i^n = \sum_{s=t}^{t+n-1} \gamma^{s-t} r_s + \gamma^n z_i$$

- The expectation is:

$$\sum_{s=t}^{t+n-1} \gamma^{s-t} r_s + \gamma^n Q(x_{t+n}, a)$$

Reactor

Distributional Retrace

- Original Retrace:

$$\Delta Q(x_t, a_t) \stackrel{\text{def}}{=} \sum_{s \geq t} \gamma^{s-t} (c_{t+1} \ldots c_s) \delta_s^\pi Q$$

Reactor

Distributional Retrace

- Original Retrace:

$$\Delta Q(x_t, a_t) \stackrel{\text{def}}{=} \sum_{s \geq t} \gamma^{s-t} (c_{t+1} \dots c_s) \delta_s^\pi Q$$

- Distributional Retrace:

$$\Delta Q(x_t, a_t) = \sum_{n \geq 1} \sum_{a \in \mathcal{A}} \alpha_{n,a} \Big[\underbrace{\sum_{s=t}^{t+n-1} \gamma^{s-t} r_s + \gamma^n Q(x_{t+n}, a)}_{n\text{-step Bellman backup}} \Big] - Q(x_t, a_t)$$

where $\alpha_{n,a} = \big(c_{t+1} \dots c_{t+n-1}\big)\big(\pi(a|x_{t+n}) - \mathbb{I}\{a = a_{t+n}\} c_{t+n}\big)$

Reactor

Distributional Retrace

- A mixture of n-step distribution (Retrace target distribution):

$$\sum_{i=1} q_i^*(x_t, a_t)\delta_{z_i}, \text{ with } q_i^*(x_t, a_t) = \sum_{n\geq 1}\sum_{a} \alpha_{n,a} \sum_{j} q_j(x_{t+n}, a_{t+n}) h_{z_i}(z_j^n)$$

Reactor

Distributional Retrace

- A mixture of n-step distribution (Retrace target distribution):

$$\sum_{i=1} q_i^*(x_t, a_t)\delta_{z_i}, \text{ with } q_i^*(x_t, a_t) = \sum_{n\geq 1}\sum_{a} \alpha_{n,a} \sum_{j} q_j(x_{t+n}, a_{t+n}) h_{z_i}(z_j^n)$$

- Update the current probabilities by performing a gradient step on the KL-Loss:

$$\nabla \mathrm{KL}(q^*(x_t, a_t), q(x_t, a_t)) = -\sum_{i=1} q_i^*(x_t, a_t) \nabla \log q_i(x_t, a_t)$$

Reactor

Distributional Retrace

- A mixture of n-step distribution (Retrace target distribution):

$$\sum_{i=1} q_i^*(x_t, a_t)\delta_{z_i}, \text{ with } q_i^*(x_t, a_t) = \sum_{n\geq 1}\sum_{a} \alpha_{n,a} \sum_{j} q_j(x_{t+n}, a_{t+n}) h_{z_i}(z_j^n)$$

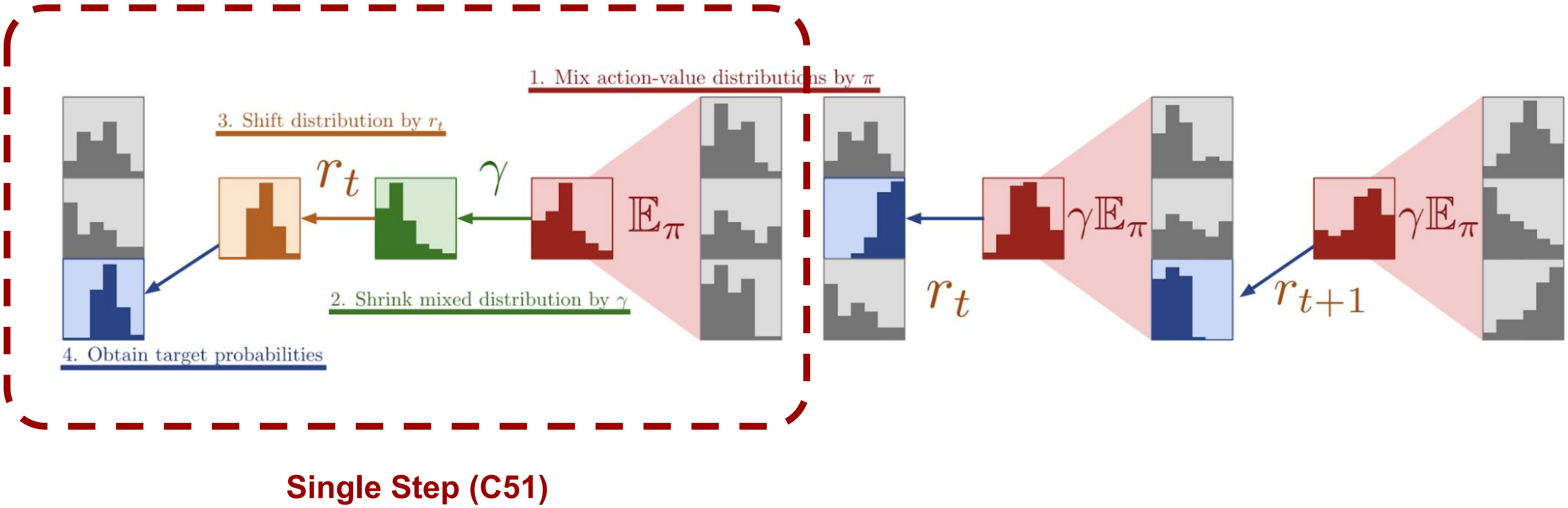
- Update the current probabilities by performing a gradient step on the KL-Loss:

$$\nabla \mathrm{KL}(q^*(x_t, a_t), q(x_t, a_t)) = -\sum_{i=1} q_i^*(x_t, a_t) \nabla \log q_i(x_t, a_t)$$

- Distributional Retrace is a **linear combination** of n-step Bellman backups

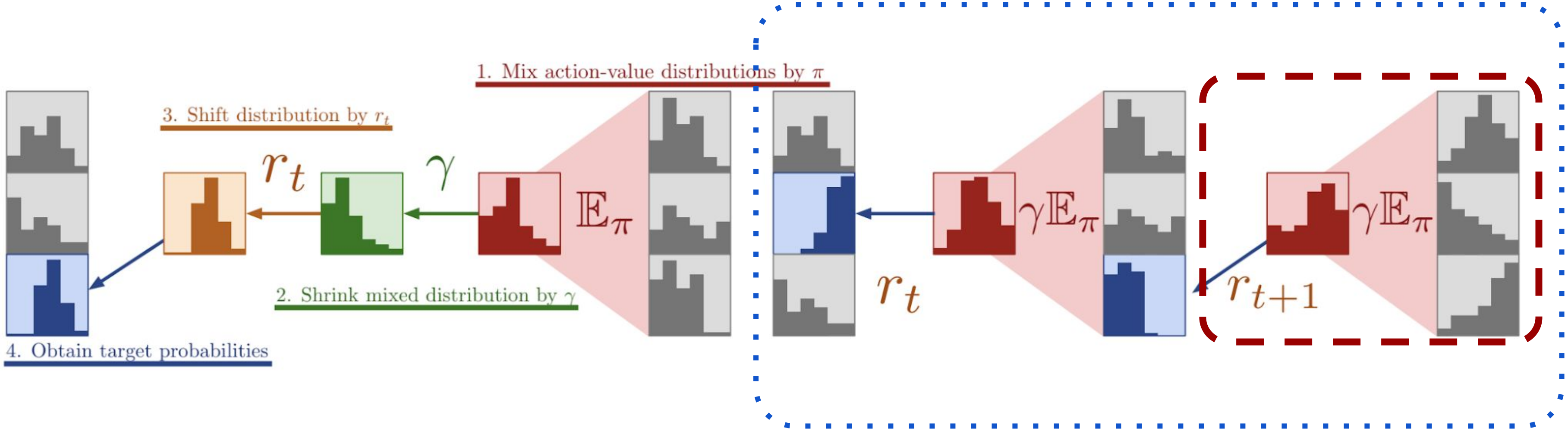
Reactor

Distributional Retrace



Reactor

Distributional Retrace



Multi Steps
Distributional Retrace

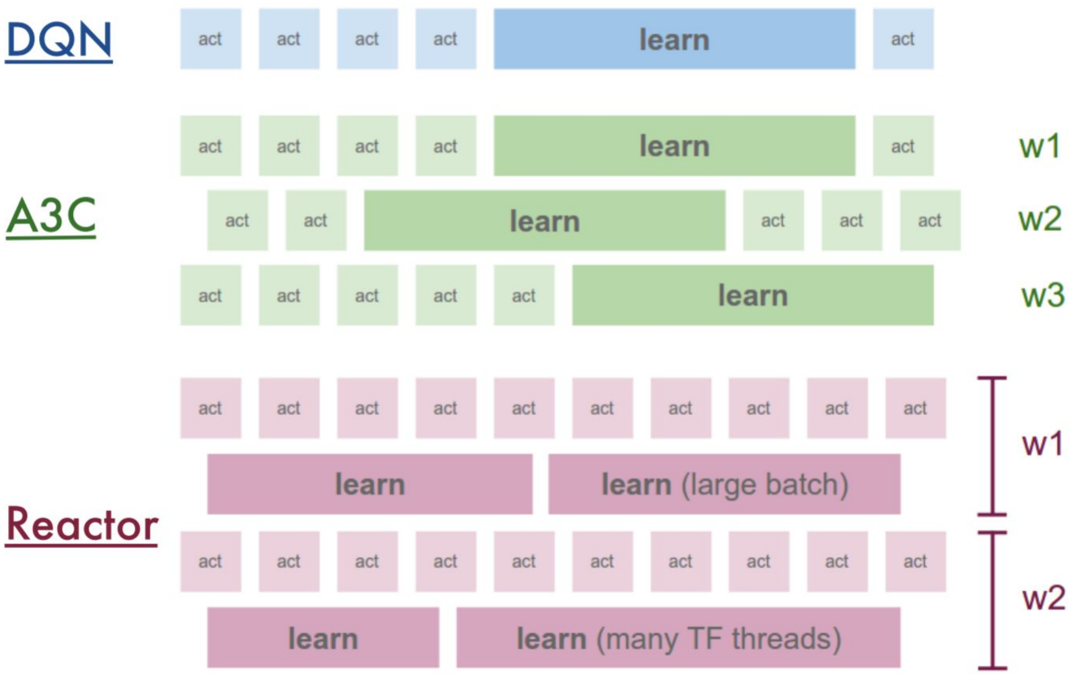
Reactor

Prioritized sequences replay

- Prioritized experience replay adds new transitions to the replay buffer with a constant priority
- Propose a way to add experience to the buffer with no priority, inserting a priority only after the transition has been sampled and used for training.
- Also, instead of sampling transitions, we assign priorities to all (overlapping) sequences of length n.
- When sampling, sequences with an assigned priority are sampled proportionally to that priority.

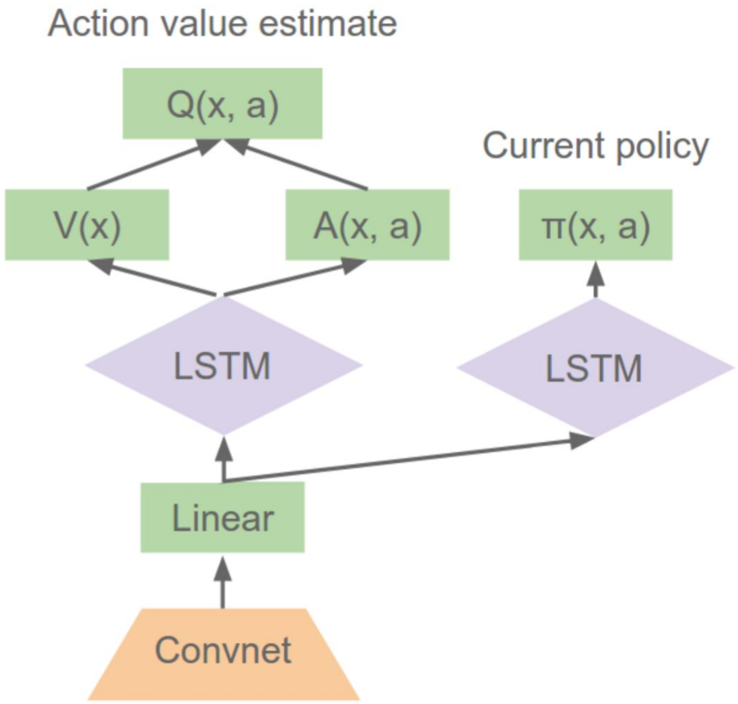
Reactor

Architecture



Agent architecture

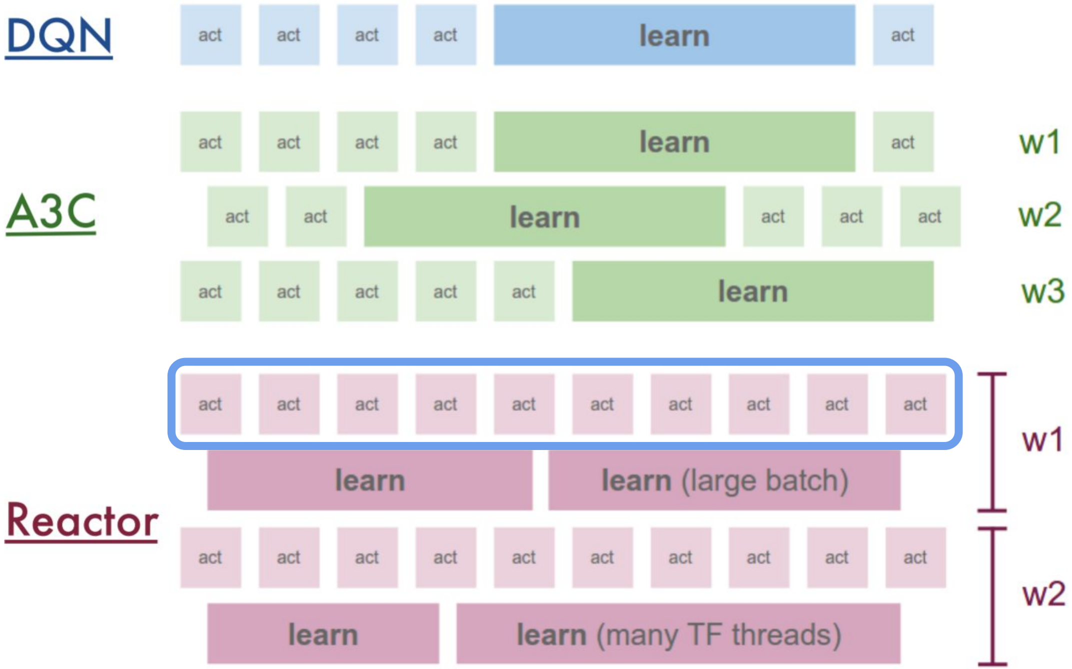
Decouple agent training:
Action-learning pair



Network architecture

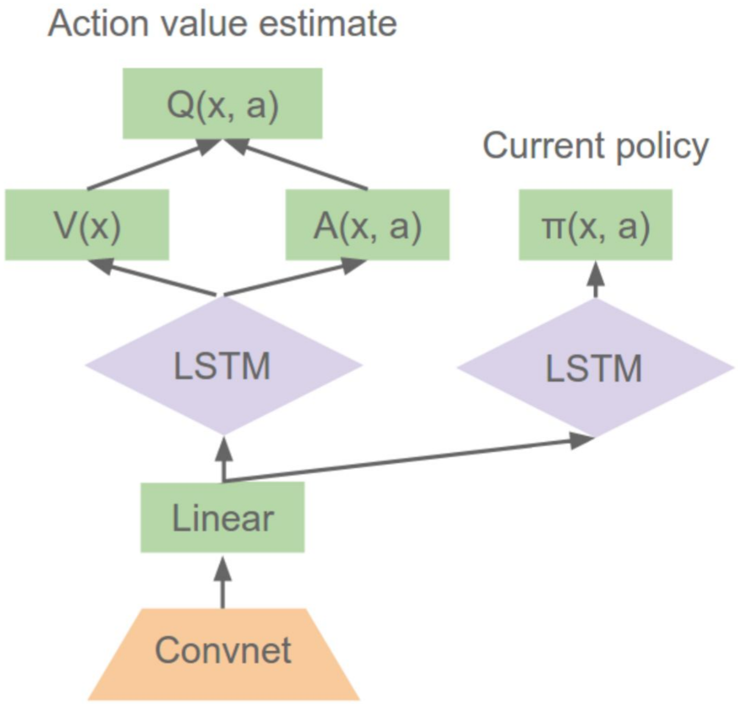
Reactor

Architecture



Agent architecture

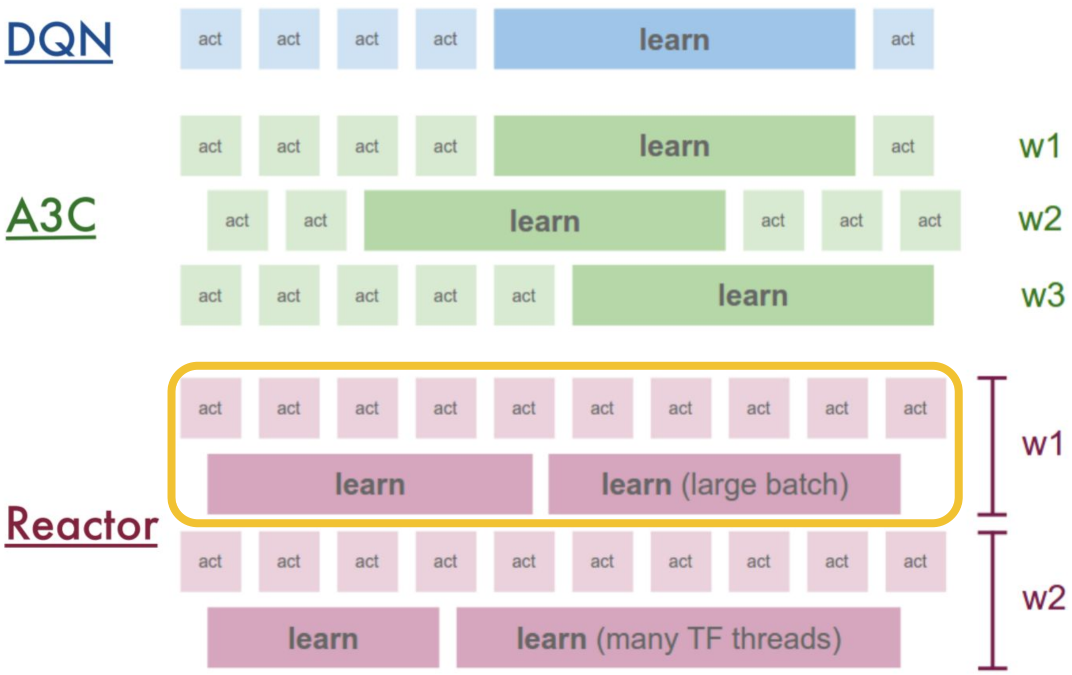
Thread for action or learning



Network architecture

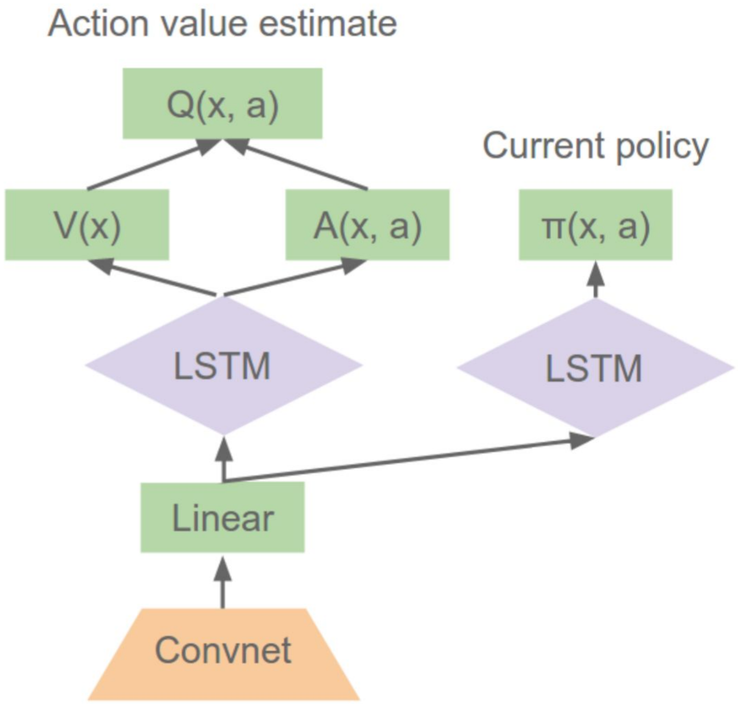
Reactor

Architecture



Worker for action-learning pair

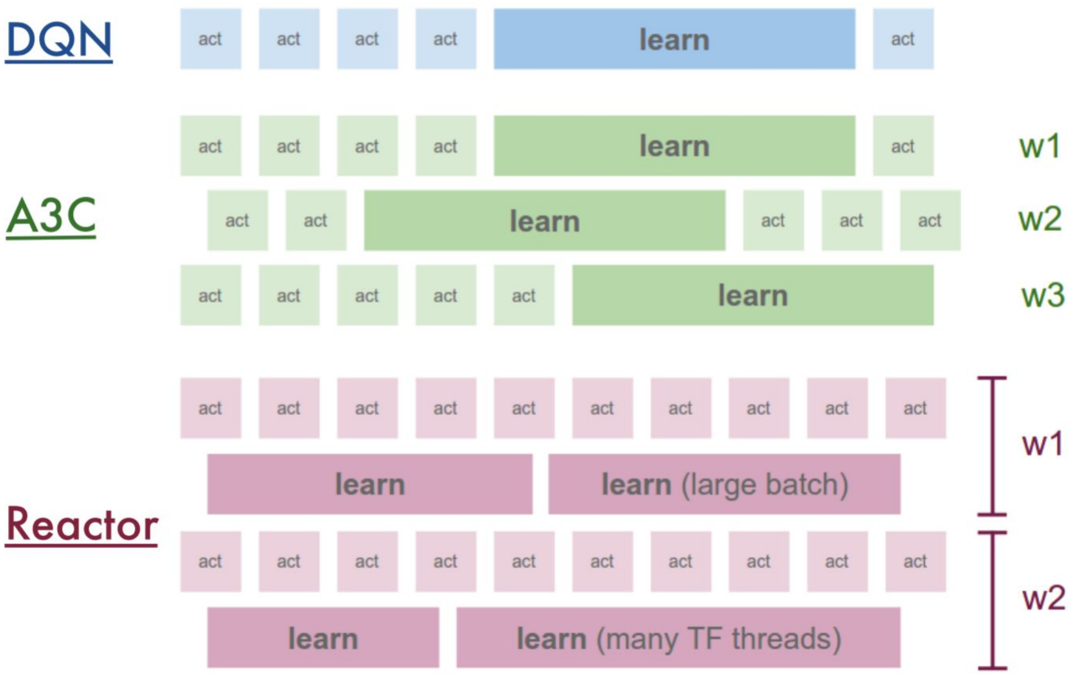
Agent architecture



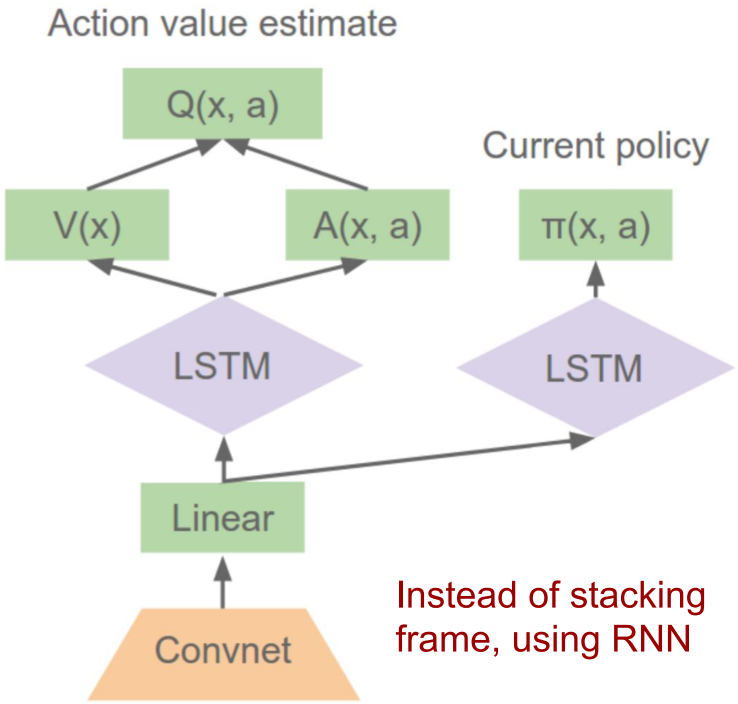
Network architecture

Reactor

Architecture



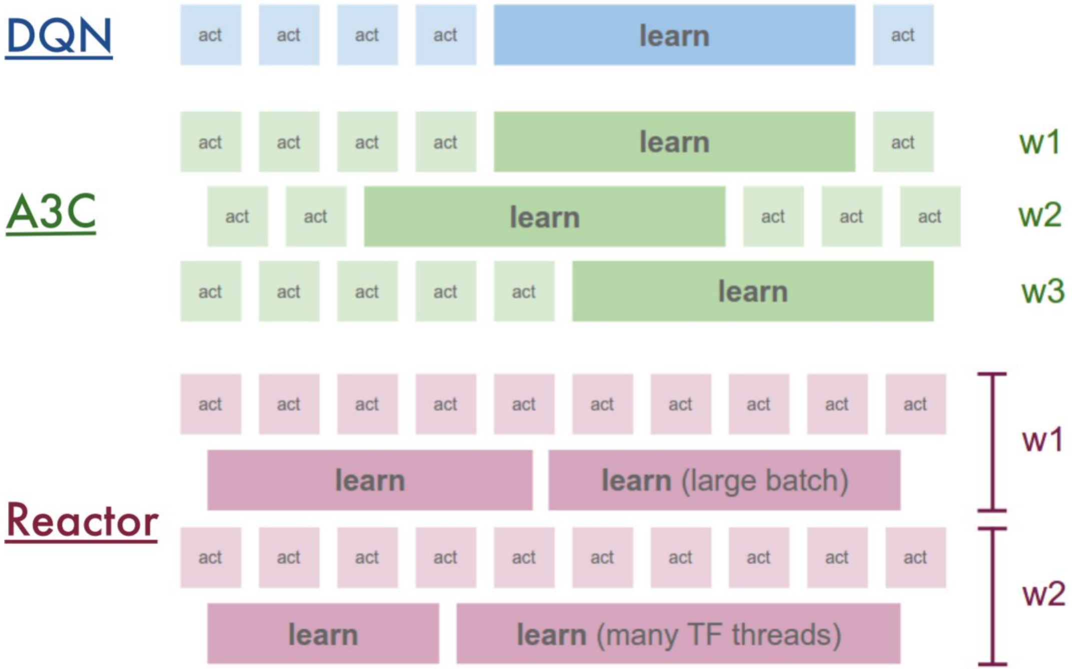
Agent architecture



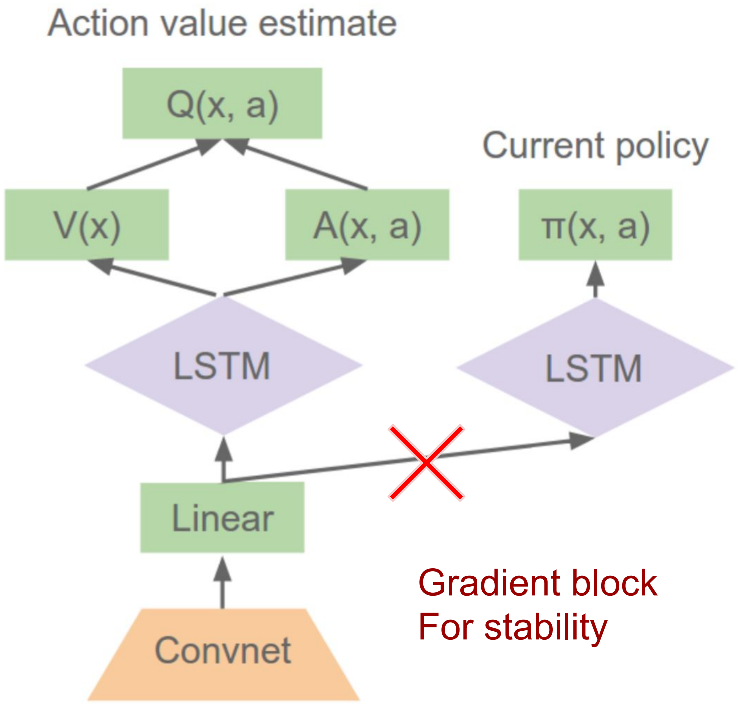
Network architecture

Reactor

Architecture



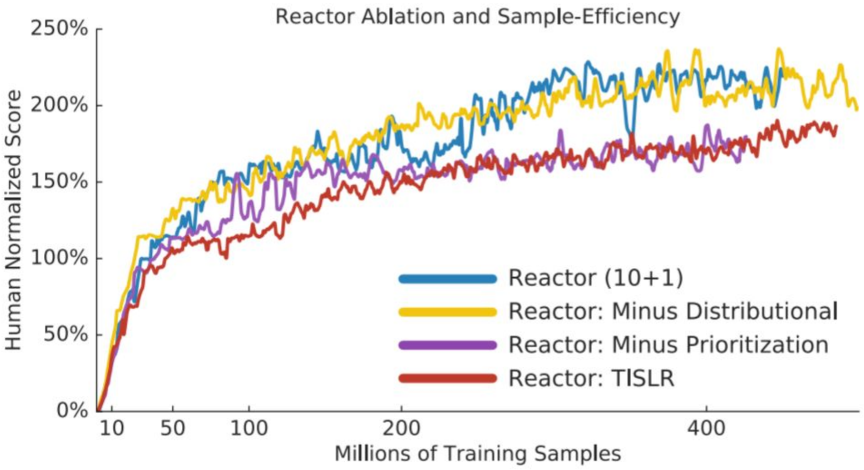
Agent architecture

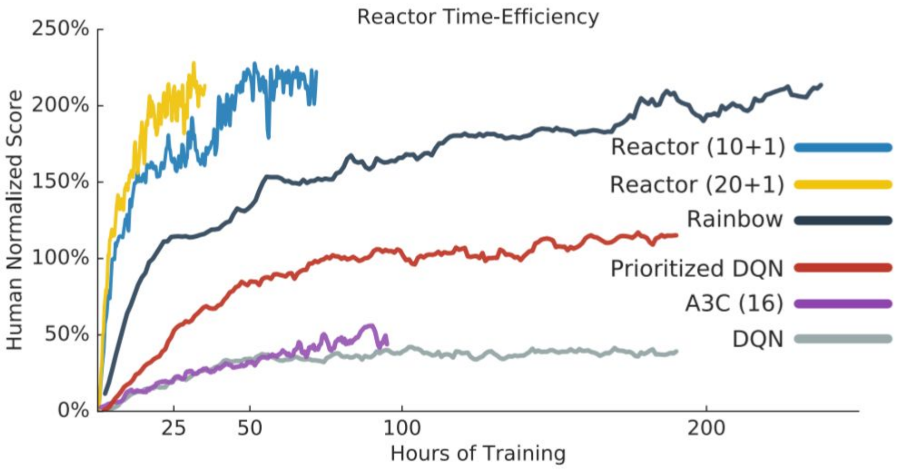


Network architecture

Reactor

Experiments

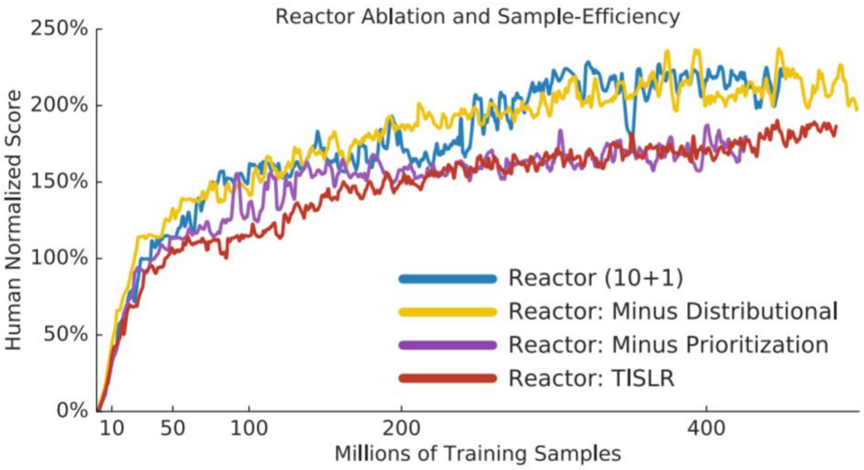


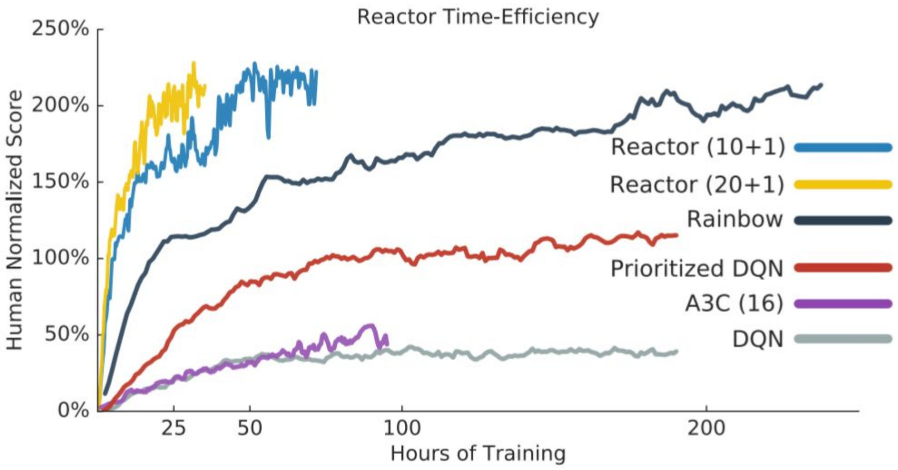


TISLR -> add β-LOO -> add Prioritization -> add distributional

Reactor

Experiments





Reactor (10+1) means:

- 10 workers for action-learner pair
- 1 worker for shared parameter server (for network)

Reactor

Experiments

Reactor performances on Atari

Algorithm	Normalized scores	Mean rank	Elo
Random	0.00	11.65	-563
Human	1.00	6.82	0
DQN	0.69	9.05	-172
DDQN	1.11	7.63	-58
Duel	1.17	6.35	32
Prior	1.13	6.63	13
Prior. Duel.	1.15	6.25	40
A3C LSTM	1.13	6.30	37
Rainbow	1.53	4.18	186
Reactor ND [5]	1.51	4.98	126
Reactor	1.65	4.58	156
Reactor 500M	$\boxed{\textbf{1.82}}$	**3.65**	**227**

Table 1: Random human starts

Algorithm	Normalized scores	Mean rank	Elo
Random	0.00	10.93	-673
Human	1.00	6.89	0
DQN	0.79	8.65	-167
DDQN	1.18	7.28	-27
Duel	1.51	5.19	143
Prior	1.24	6.11	70
Prior. Duel.	1.72	5.44	126
ACER[6] 500M	1.9	-	-
Rainbow	**2.31**	3.63	270
Reactor ND [5]	1.80	4.53	195
Reactor	1.87	4.46	196
Reactor 500M	2.30	**3.47**	**280**

Table 2: 30 random no-op starts.

Reactor

Experiments

Rainbow in no-op case is more sample efficiency,
But may be overfitting

Reactor performances on Atari

ALGORITHM	NORMALIZED SCORES	MEAN RANK	ELO
RANDOM	0.00	11.65	-563
HUMAN	1.00	6.82	0
DQN	0.69	9.05	-172
DDQN	1.11	7.63	-58
DUEL	1.17	6.35	32
PRIOR	1.13	6.63	13
PRIOR. DUEL.	1.15	6.25	40
A3C LSTM	1.13	6.30	37
RAINBOW	1.53	4.18	186
REACTOR ND [5]	1.51	4.98	126
REACTOR	1.65	4.58	156
REACTOR 500M	**1.82**	**3.65**	**227**

Table 1: Random human starts

ALGORITHM	NORMALIZED SCORES	MEAN RANK	ELO
RANDOM	0.00	10.93	-673
HUMAN	1.00	6.89	0
DQN	0.79	8.65	-167
DDQN	1.18	7.28	-27
DUEL	1.51	5.19	143
PRIOR	1.24	6.11	70
PRIOR. DUEL.	1.72	5.44	126
ACER[6] 500M	1.9	-	-
RAINBOW	**2.31**	3.63	270
REACTOR ND [5]	1.80	4.53	195
REACTOR	1.87	4.46	196
REACTOR 500M	2.30	**3.47**	**280**

Table 2: 30 random no-op starts.

Thank you !