Off-Policy Learning

Project

- Two environments: one discrete, one continuous
- Jelly bean world: https://github.com/eaplatanios/jelly-bean-world
- We will provide you with a learned feature space instead of the native image space
- A Mujoco-based task: https://gym.openai.com/envs/Hopper-v2/
- Project is carried out in teams of 2-4 students
- Deliverables: project report (4-pages NeurIPS style file), 2-minute video presentation
- Leaderboard evaluation will be set up by us
-

Project

- Measurements: return, variance of return over n runs, number of steps until a certain performance level is reached
- Challenge: multi-task evaluation (problem changes after a certain number of episodes)
- We will provide some baselines (random agent, TA basic agent)
- Grading criteria based on performance, creativity of project, presentation (written and video)
- Written report MUST include a statement of contributions that all participants agree with

Off-policy Methods

- Learn the value of the *target policy* π from experience due to *behavior policy b*
- For example, π is the greedy policy (and ultimately the optimal policy) while μ is exploratory (e.g., ε-soft)
- In general, we only require *coverage*, i.e., that *b* generates behavior that covers, or includes, π

$$\pi(a|s) > 0 \quad \text{for every } s,a \text{ at which} \quad b(a|s) > 0$$

- Idea: *importance sampling*
  - Weight each return by the *ratio of the probabilities* of the trajectory under the two policies

Importance Sampling in General

- Suppose we want to estimate the expected value of a function f depending on a random variable X drawn according to the *target* probability distribution $P(X)$.
- If we had N samples x_i drawn from $P(X)$, we could estimate the expectation using the empirical mean:

$$E_P[f] \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

- But instead, we have only samples drawn according to a different *proposal* or *sampling* distribution $Q(X)$.
- How can we do the estimation?

Regular Importance Sampling

- We do a simple trick:

$$
\begin{aligned}
E_P[f] &= \sum_x f(x)P(X=x) \\
&= \sum_x f(x)Q(X=x)\frac{P(X=x)}{Q(X=x)} = E_Q\left[f\frac{P}{Q}\right]
\end{aligned}
$$

- Only requirement: if $P(x) > 0$ then $Q(x) > 0$
- So for an estimator, we should average each sample of the function, $f(x_i)$ *weighted* by the ratio of its probability under the target and the sampling distribution:

$$
E_p[f] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)\frac{P(x_i)}{Q(x_i)}
$$

Normalized Importance Sampling

- Regular importance sampling is an unbiased and consistent estimator, but it can have high variance
- Variance depends on closeness of P and Q
- Instead, we can treat P/Q ratios as weights and do a weighted sum (instead of using N in the denominator)
- This is called Normalized or Weighted IS
- The estimator is biased but consistent and tends to have lower variance

Applying IS to Policy Evaluation

- Function for which we want the expectation is the return
- Target distribution P is the distribution of trajectories under *target policy* π
- Proposal distribution Q is distribution of trajectories under *behavior policy b*
- Note that P and Q can be very different depending on the horizon!
- But there is structure in P and Q that we can exploit

Importance Sampling Ratio

- Probability of the rest of the trajectory, after S_t, under π:

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

- In importance sampling, each return is weighted by the relative probability of the trajectory under the two policies

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

- This is called the *importance sampling ratio*
- All importance sampling ratios have expected value 1

$$\mathbb{E}\left[\frac{\pi(A_k|S_k)}{b(A_k|S_k)}\right] \doteq \sum_a b(a|S_k)\frac{\pi(a|S_k)}{b(a|S_k)} = \sum_a \pi(a|S_k) = 1$$

Importance Sampling

- New notation: time steps increase across episode boundaries:
  - . . . s . . . . ▨ . . . . . . ▨ . . . s . . . . ▨ . . .
  - $t =$ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

$\mathcal{T}(s) = \{4, 20\}$ $\qquad$ $T(4) = 9 \quad T(20) = 25$

set of start times $\qquad$ next termination times

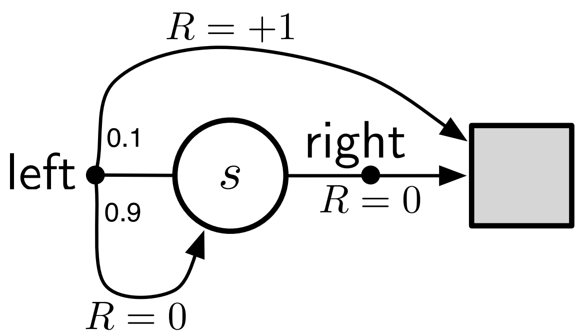
- *Ordinary importance sampling* forms estimate

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

- Whereas *weighted importance sampling* forms estimate

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Example of infinite variance under *ordinary* importance sampling



$\pi(\text{left}|s) = 1$

$b(\text{left}|s) = \frac{1}{2}$

$\gamma = 1$

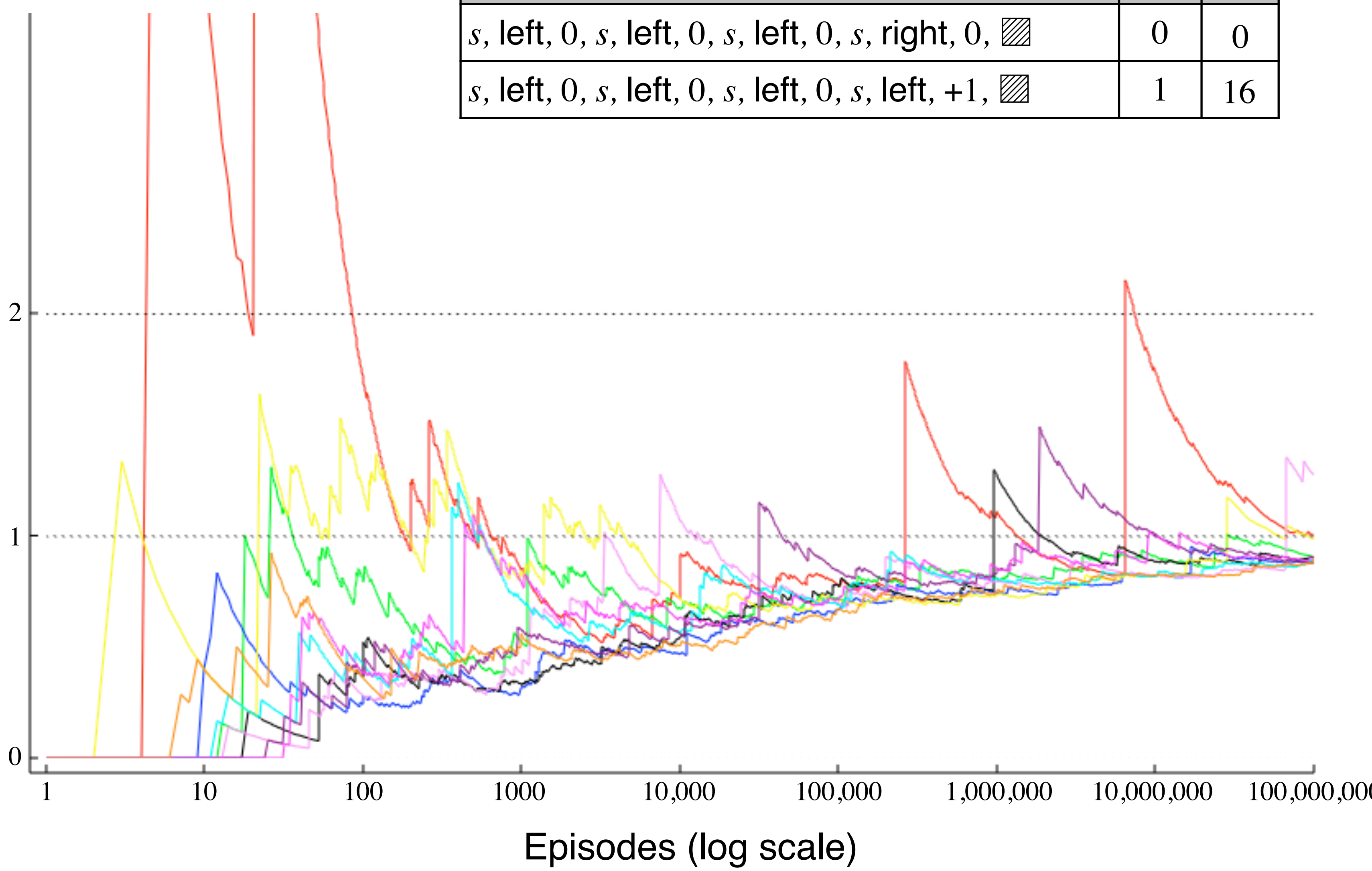
$v_\pi(s) = 1$

$\frac{\pi(\text{right}|s)}{b(\text{right}|s)} =$

$\frac{\pi(\text{left}|s)}{b(\text{left}|s)} =$

Trajectory	G_0	$\rho_{0:T-1}$
s, left, 0, s, left, 0, s, left, 0, s, right, 0, ▨	0	0
s, left, 0, s, left, 0, s, left, 0, s, left, +1, ▨	1	16

Monte-Carlo estimate of $v_\pi(s)$ with ordinary importance sampling (ten runs)



OIS:

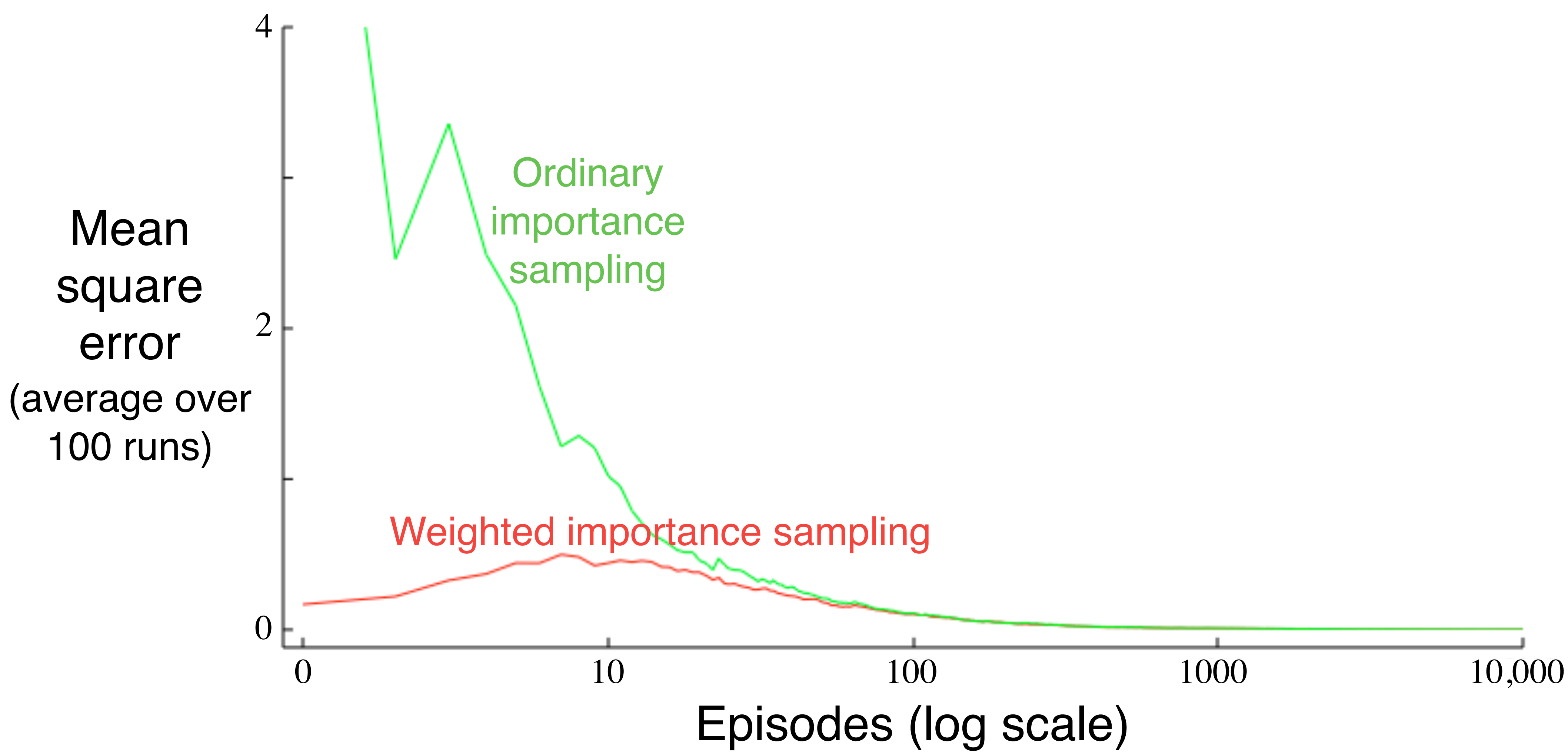
$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

WIS:

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1}}$$

Example: Off-policy Estimation of the value of a *single* Blackjack State

- State is player-sum 13, dealer-showing 2, useable ace
- Target policy is stick only on 20 or 21
- Behavior policy is equiprobable
- True value ≈ –0.27726



Discounting-aware Importance Sampling (motivation)

- So far we have weighted returns without taking into account that they are a discounted sum
- This can't be the best one can do!
- For example, suppose $\boldsymbol{\gamma} = 0$
  - Then G_0 will be weighted by

$$\rho_{0:T-1} = \frac{\pi(A_0|S_0)}{b(A_0|S_0)} \frac{\pi(A_1|S_1)}{b(A_1|S_1)} \cdots \frac{\pi(A_{T-1}|S_{T-1})}{b(A_{T-1}|S_{T-1})}$$

  - But it really need only be weighted by

$$\rho_{0:1} = \frac{\pi(A_0|S_0)}{b(A_0|S_0)}$$

  - Which would have <u>much smaller variance</u>

Discounting-aware Importance Sampling

- Define the flat partial return:

$$\bar{G}_{t:h} \doteq R_{t+1} + R_{t+2} + \cdots + R_h, \qquad 0 \leq t < h \leq T$$

- Then

$$G_t = (1-\gamma) \sum_{h=t+1}^{T-1} \gamma^{h-t-1} \bar{G}_{t:h} \quad + \quad \gamma^{T-t-1} \bar{G}_{t:T}$$

- Ordinary discounting-aware IS:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \left((1-\gamma) \sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1} \rho_{t:h-1} \bar{G}_{t:h} \quad + \quad \gamma^{T(t)-t-1} \rho_{t:T(t)-1} \bar{G}_{t:T(t)} \right)}{|\mathcal{T}(s)|}$$

- Weighted discounting-aware IS:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \left((1-\gamma) \sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1} \rho_{t:h-1} \bar{G}_{t:h} \quad + \quad \gamma^{T(t)-t-1} \rho_{t:T(t)-1} \bar{G}_{t:T(t)} \right)}{\sum_{t \in \mathcal{T}(s)} \left((1-\gamma) \sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1} \rho_{t:h-1} \quad + \quad \gamma^{T(t)-t-1} \rho_{t:T(t)-1} \right)}$$

Per-reward Importance Sampling

- Another way of reducing variance, even if $\boldsymbol{\gamma} = 1$
- Uses the fact that the return is a *sum of rewards*

$$\rho_t^T G_t = \rho_t^T R_{t+1} + \gamma \rho_t^T R_{t+2} + \cdots + \gamma^{k-1} \rho_t^T R_{t+k} + \cdots + \gamma^{T-t-1} \rho_t^T R_T$$

- where

$$\rho_t^T R_{t+k} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \cdots \xcancel{\frac{\pi(A_{t+k}|S_{t+k})}{\mu(A_{t+k}|S_{t+k})} \cdots \frac{\pi(A_{T-1}|S_{T-1})}{\mu(A_{T-1}|S_{T-1})}} R_{t+k}$$

Per-reward Importance Sampling

- Another way of reducing variance, even if **γ = 1**
- Uses the fact that the return is a *sum of rewards*

$$\rho_{t:T-1}G_t = \rho_{t:T-1}R_{t+1} + \cdots + \gamma^{k-1}\rho_{t:T-1}R_{t+k} + \cdots + \gamma^{T-t-1}\rho_{t:T-1}R_T$$

$$\rho_{t:T-1}R_{t+k} = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}\frac{\pi(A_{t+1}|S_{t+1})}{b(A_{t+1}|S_{t+1})}\cdots\cancel{\frac{\pi(A_{t+k}|S_{t+k})}{b(A_{t+k}|S_{t+k})}\cdots\frac{\pi(A_{T-1}|S_{T-1})}{b(A_{T-1}|S_{T-1})}}R_{t+k}.$$

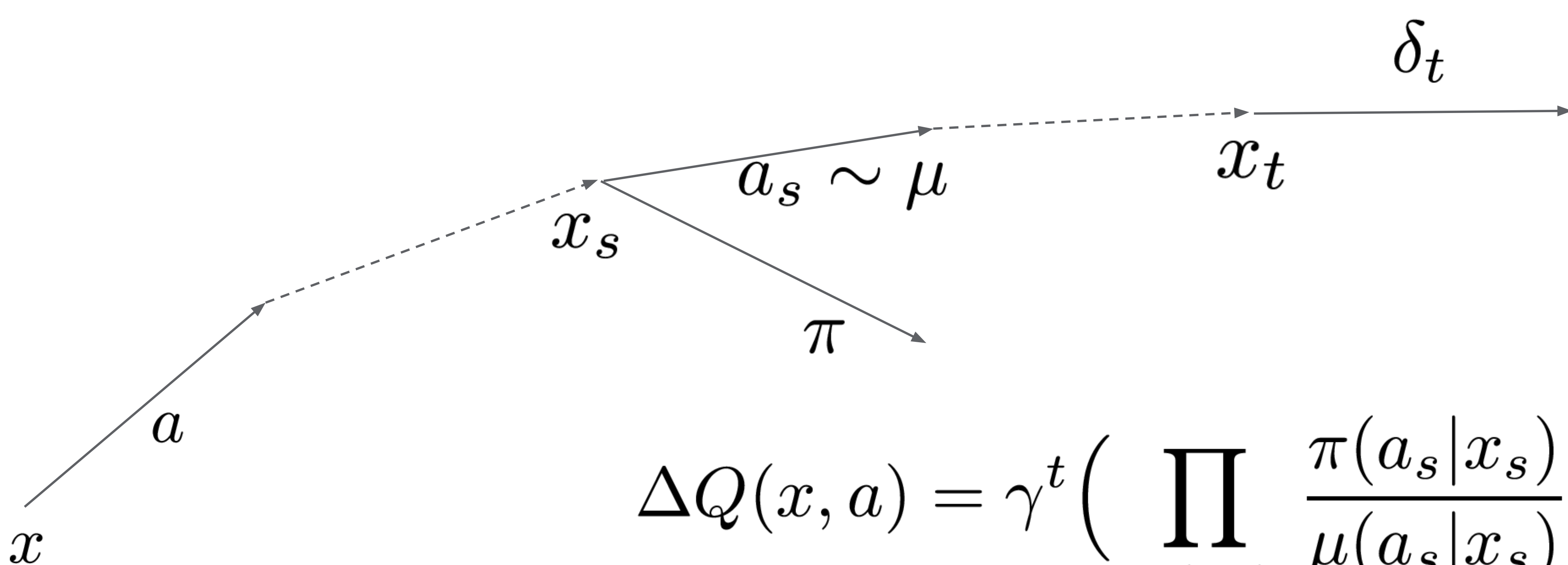
$$\therefore\ \mathbb{E}[\rho_{t:T-1}R_{t+k}] = \mathbb{E}[\rho_{t:t+k-1}R_{t+k}]$$

$$\therefore\ \mathbb{E}[\rho_{t:T-1}G_t] = \mathbb{E}\Big[\underbrace{\rho_{t:t}R_{t+1} + \cdots + \gamma^{k-1}\rho_{t:t+k-1}R_{t+k} + \cdots + \gamma^{T-t-1}\rho_{t:T-1}R_T}_{\tilde{G}_t}\Big]$$

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)}\tilde{G}_t}{|\mathcal{T}(s)|}$$

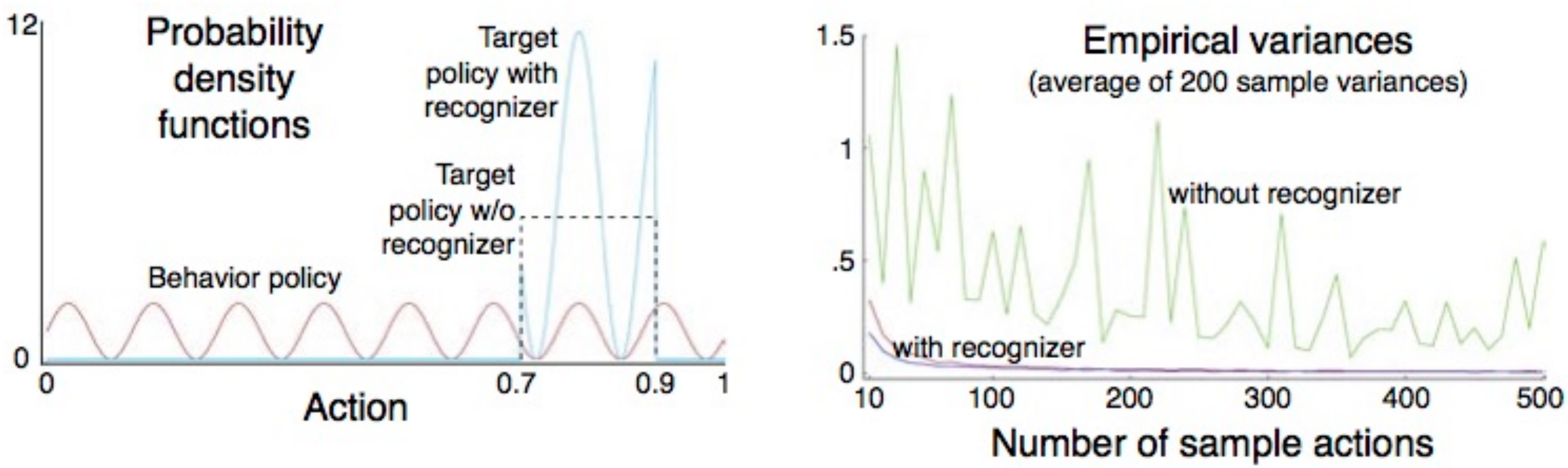
Implementation

- Importance sampling ratios fold into the eligibility trace
- Multiply at each step by an extra factor
- But on long trajectories traces will get cut a lot!



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

Recognizers



- Recognizer makes a *target policy that aligns with the behavior*
- Goal: Make off-policy learning efficient

- Target policy is obtained by composing the behavior policy with the recognizer:

$$\pi(s,a) = \frac{b(s,a)\rho(s,a)}{\sum_{a'} b(s,a')\rho(s,a')}$$

Recognizer Properties

- Suppose we have a behavior policy b and we only consider target policies that choose action from a subset $a_1, ...a_k$
- Then, the policy that minimizes the variance of *one-step importance sampling updates* corresponds to the binary recognizer that is 1 for $a_1, \ldots a_k$ and 0 otherwise:

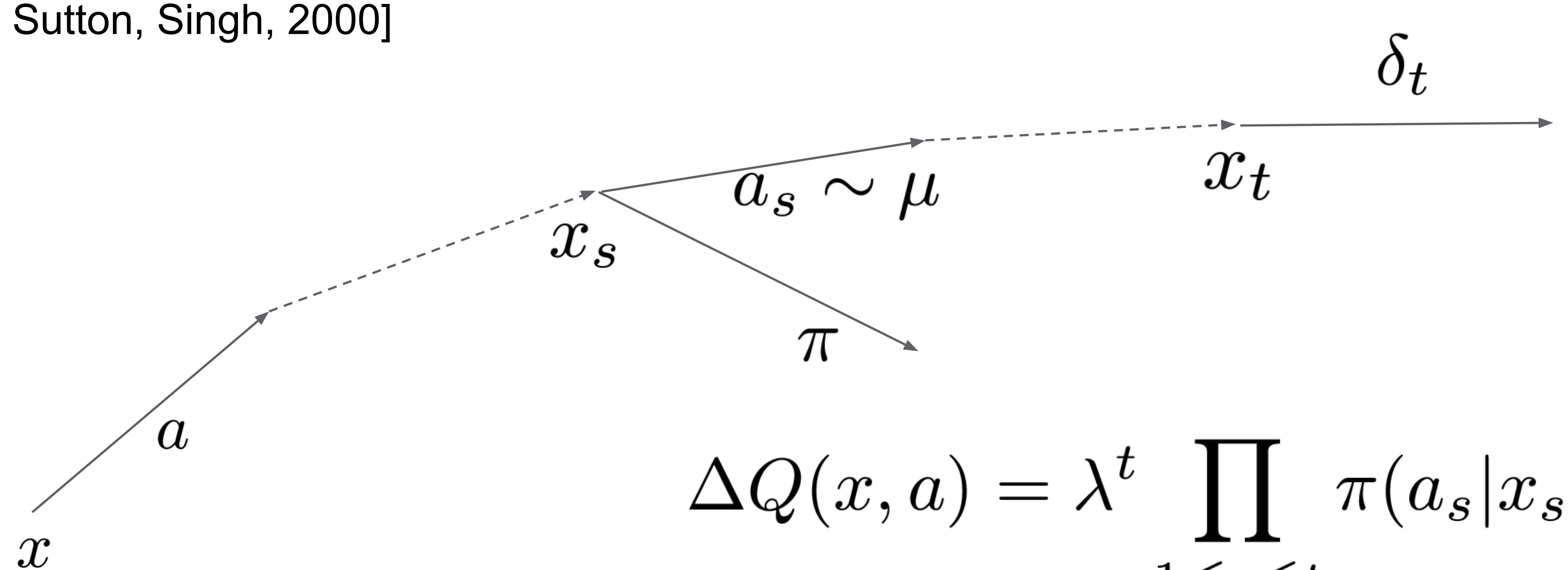
$$\arg\min_{\pi} \mathbf{E}_b \left[\left(\frac{\pi(a_i)}{b(a_i)} \right)^2 \right]$$

- Recognizing more actions leads to lower variance
- Recognizer folds in the eligibility trace in place of the importance sampling ratio
- The behavior policy does NOT need to be known (the normalization can be estimated empirically) - connection to imitation learning

Cf. Precup et al, NIPS 2005

Tree Backup

[Precup, Sutton, Singh, 2000]



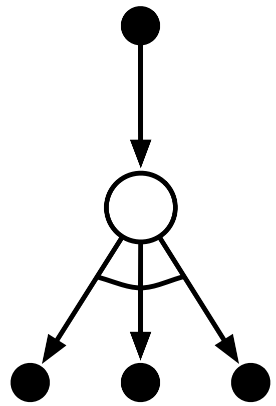
$$\Delta Q(x, a) = \lambda^t \prod_{1 \leq s \leq t} \pi(a_s | x_s) \delta_t$$

Reweight the traces by the product of target probabilities

Q-Learning: Off-Policy TD Control

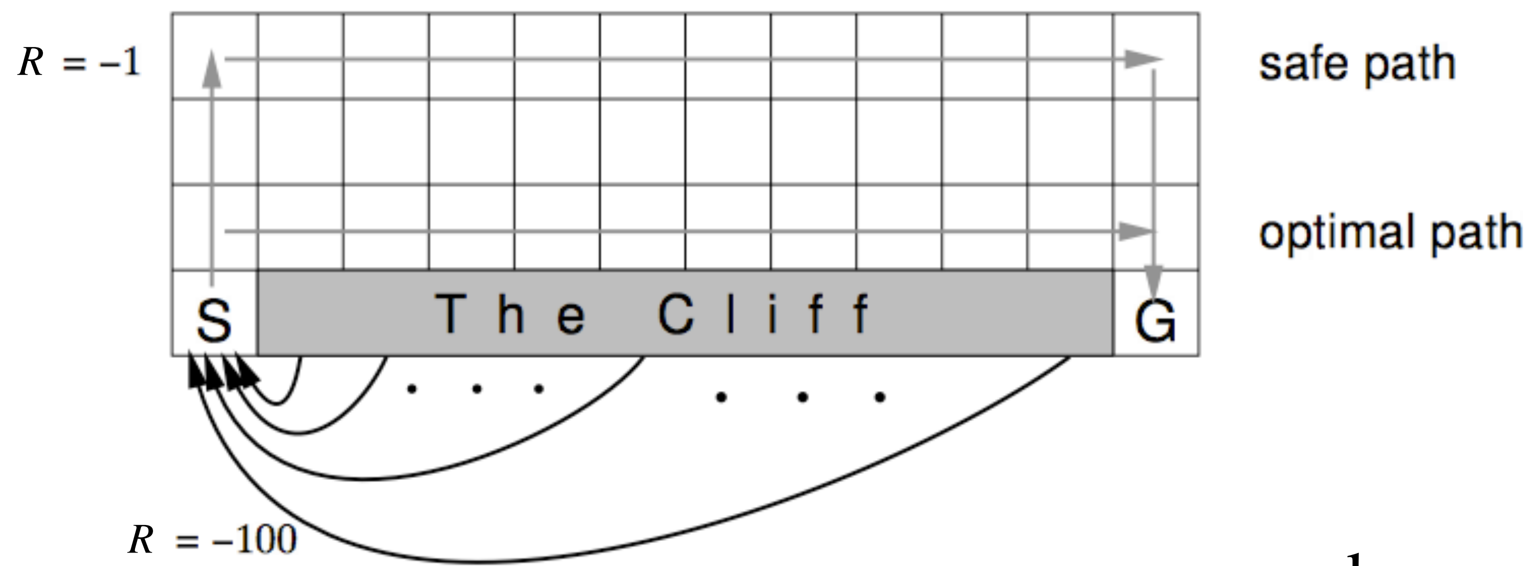
One-step Q-learning:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

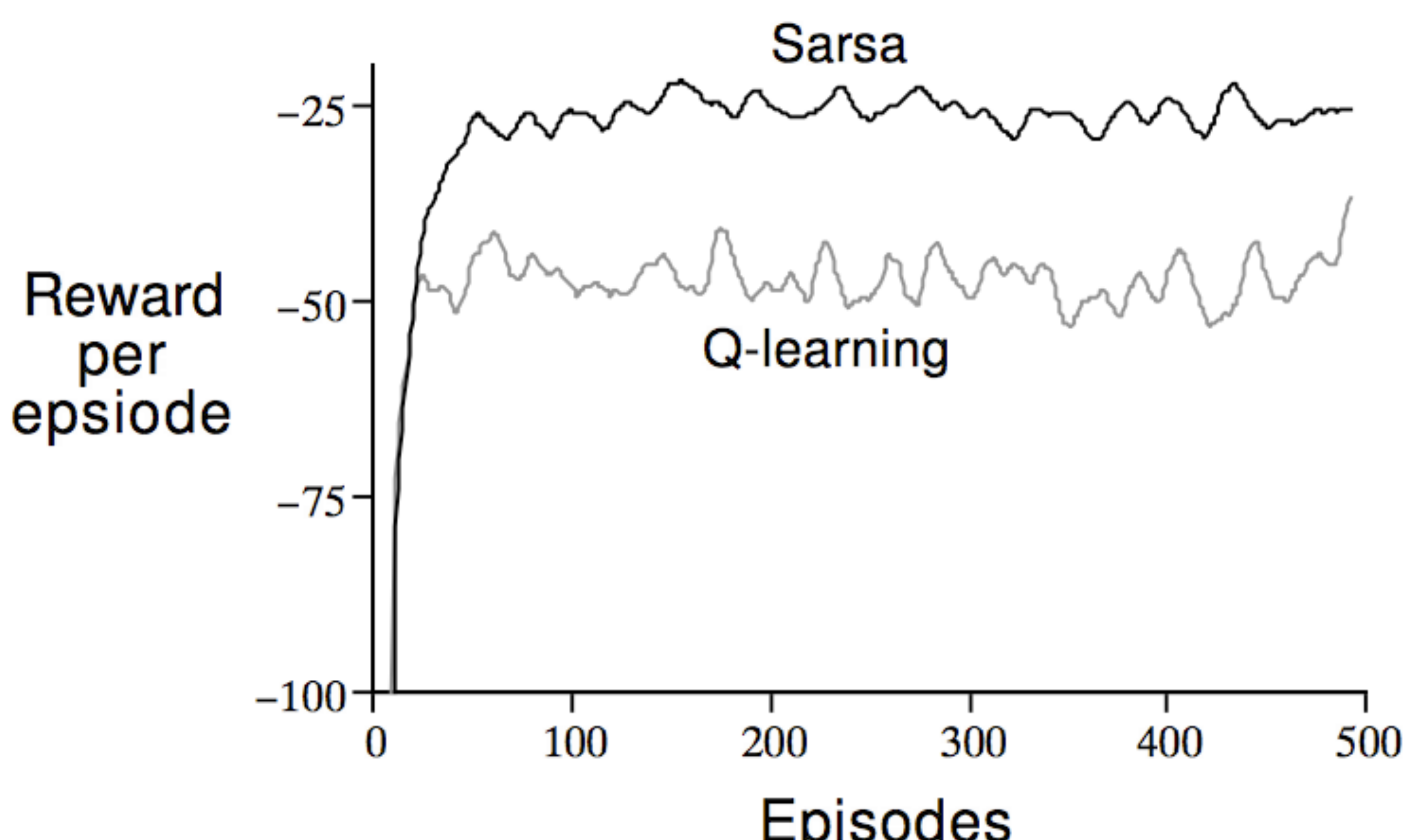


Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S,A)]$
        $S \leftarrow S'$;
    until S is terminal

Cliffwalking



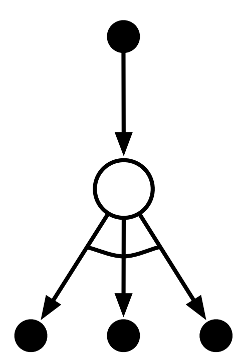
ε−greedy, $\varepsilon = 0.1$



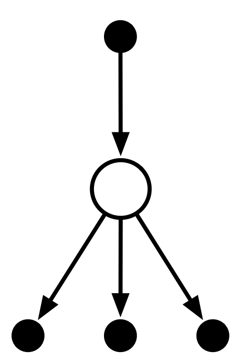
Expected Sarsa

- Instead of the *sample* value-of-next-state, use the expectation!

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big] \\
&\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big]
\end{aligned}
$$



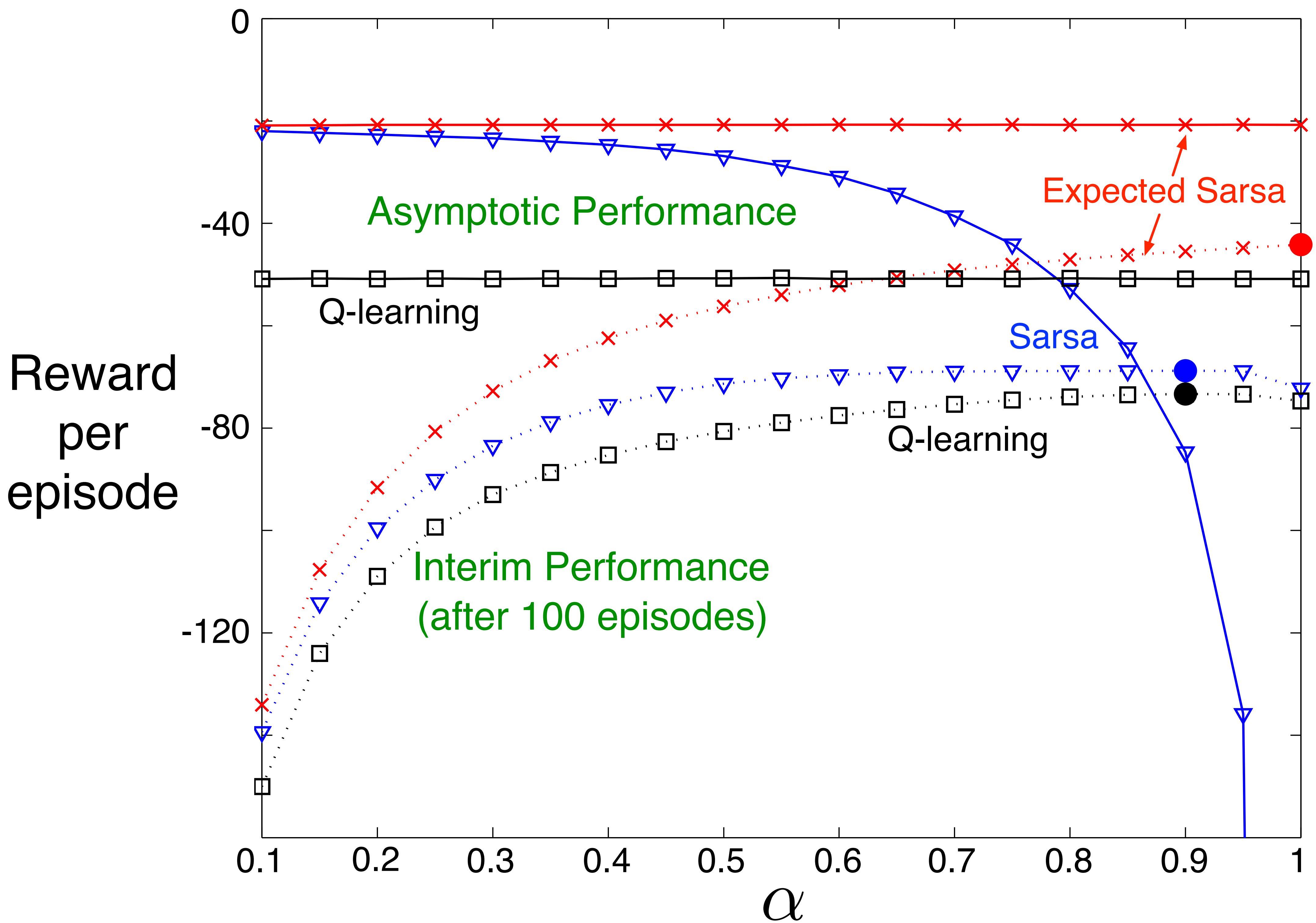
Q-learning



Expected Sarsa

- Expected Sarsa's performs better than Sarsa (but costs more)

Performance on the Cliff-walking Task



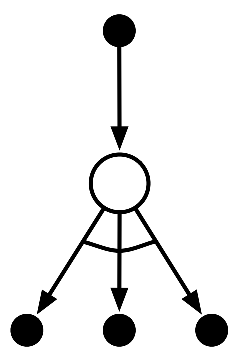
Off-policy Expected Sarsa

- Expected Sarsa generalizes to arbitrary behavior policies μ
  - in which case it includes Q-learning as the special case in which π is the greedy policy

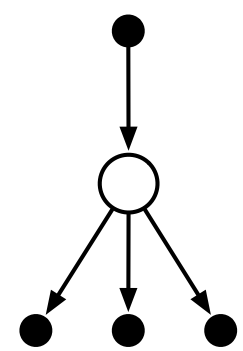
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \Big]$$

$$\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

Nothing changes here



Q-learning

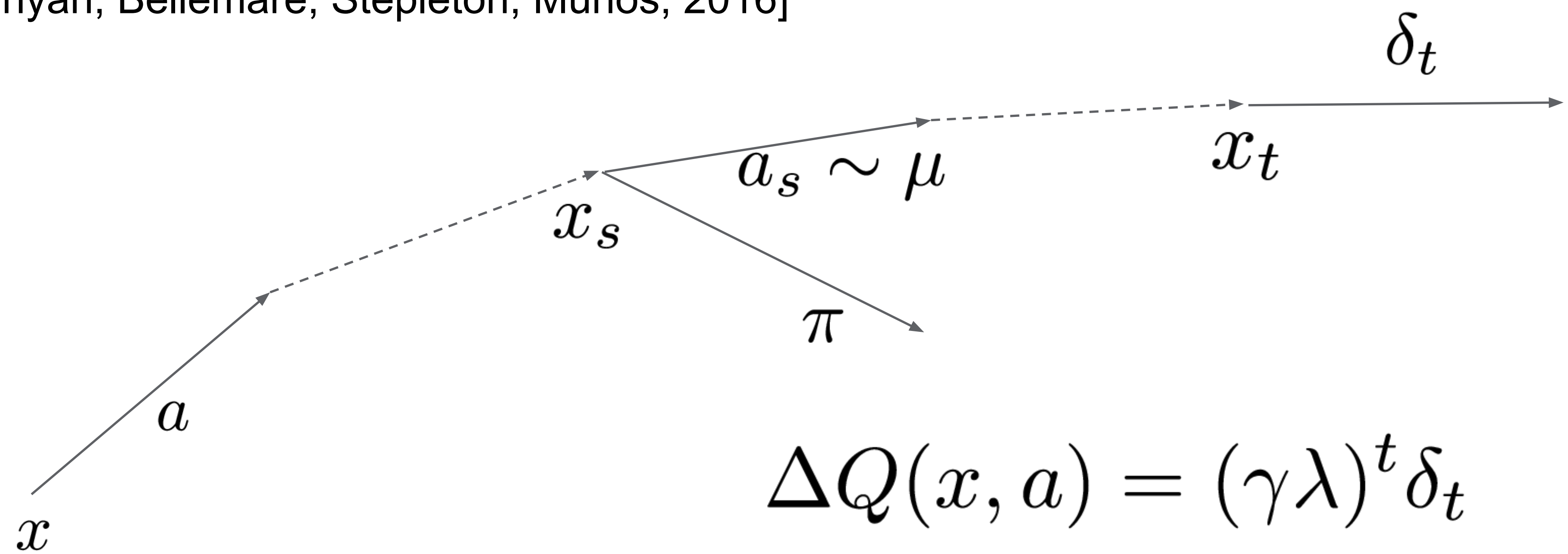


Expected Sarsa

Q-learning with Eligibility Traces

$Q^{\pi}(\lambda)$ 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!

Blueprint Off-policy Algorithm

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

Algorithm:	Trace coefficient:	Problem:
IS	$c_s = \frac{\pi(a_s \mid x_s)}{\mu(a_s \mid x_s)}$	high variance
$Q^\pi(\lambda)$	$c_s = \lambda$	not safe (off-policy)
$TB(\lambda)$	$c_s = \lambda \pi(a_s \mid x_s)$	not efficient (on-policy)

Retrace (Munos et al, 2016)

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

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): but only when needed. Note that $c_s \geq \lambda\pi(a_s|x_s)$

Retrace for Control

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

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

Theorem 2
Under previous assumptions (+ a technical assumption)
Assume (π_k) are "increasingly greedy" wrt (Q_k)
Then, a.s.,

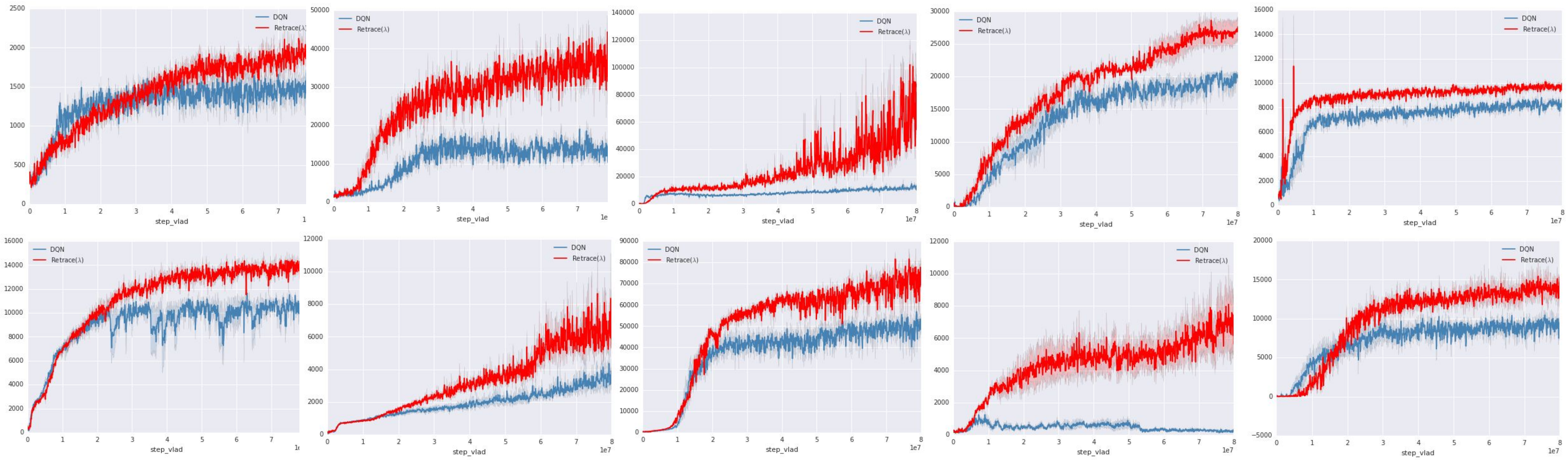
$$\boxed{Q_k \to Q^*}$$

- 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)

- "Increasingly greedy" allows for smoother traces thus faster convergence

- The behavior policies (μ_k) do **not** need to become greedy wrt (Q_k)
  $\rightarrow$ **no GLIE assumption** (Greedy in the limit with infinite exploration)
  (first return-based algo converging to Q^* without GLIE)

Retrace in Atari



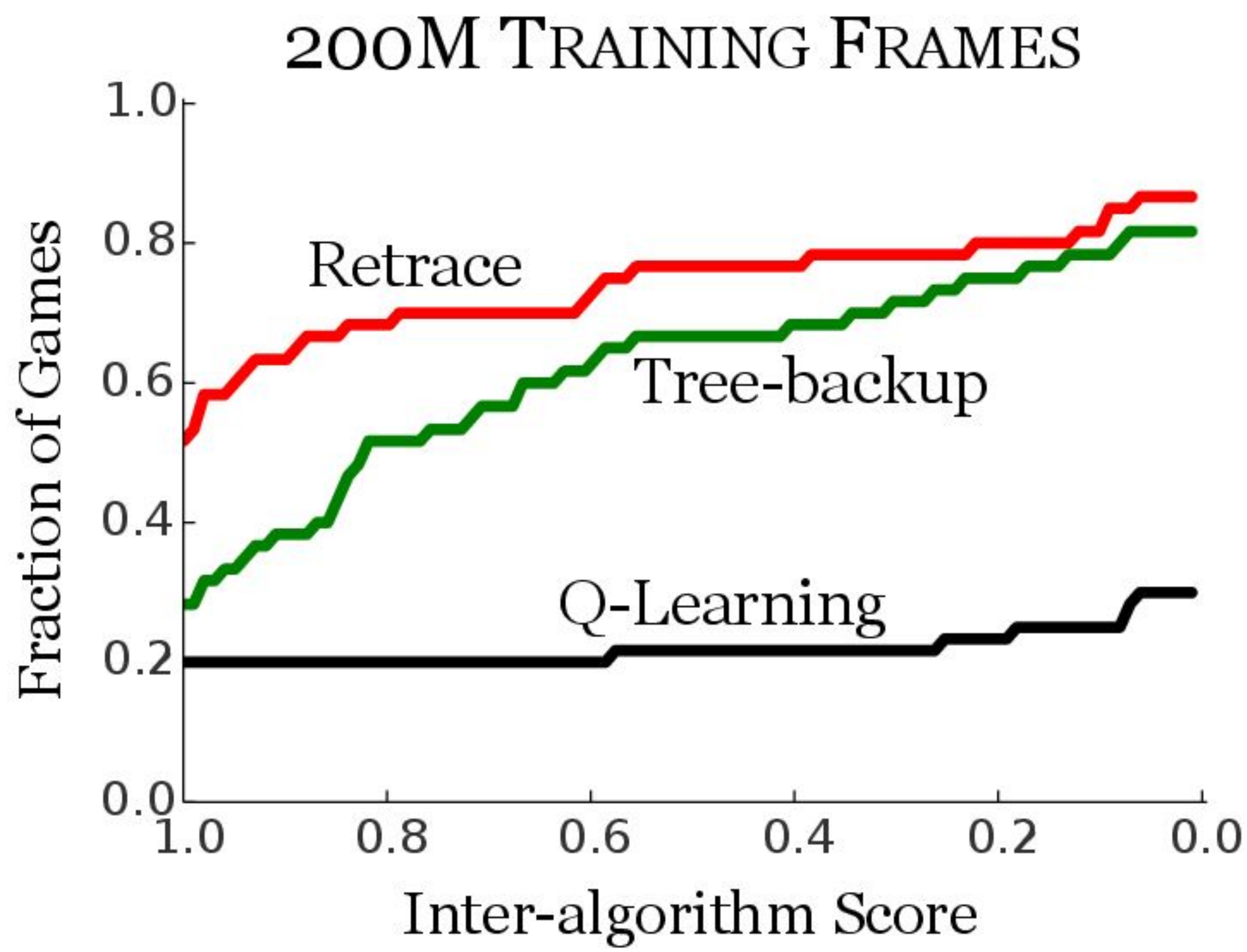
Games:

Asteroids, Defender, Demon Attack, Hero, Krull,

River Raid, Space Invaders, Star Gunner, Wizard of Wor, Zaxxon

Retrace vs Tree Backup

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



V-Trace (Espeholt et al, 2018)

- Off-policy, massively parallel actor-critic

$$v_s \stackrel{\text{def}}{=} V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \Big(\prod_{i=s}^{t-1} c_i \Big) \delta_t V$$

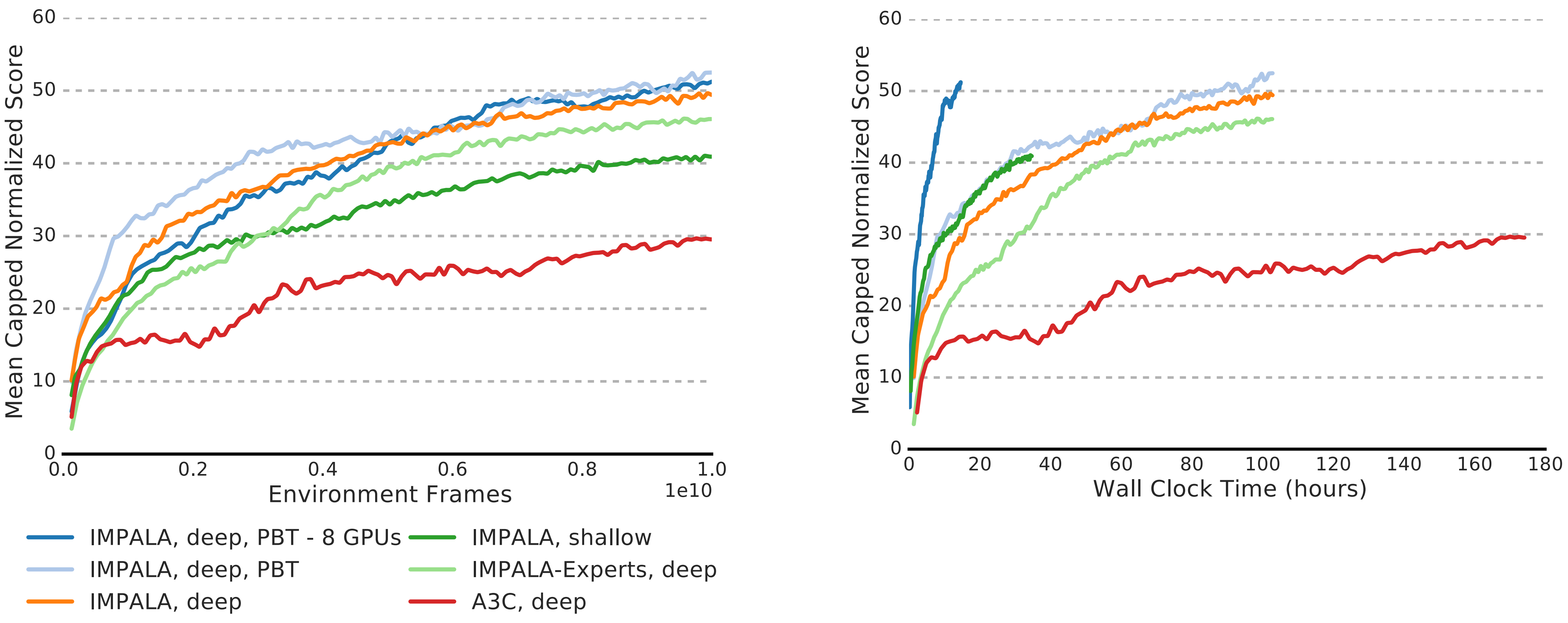
$$\delta_t V \stackrel{\text{def}}{=} \rho_t \big(r_t + \gamma V(x_{t+1}) - V(x_t) \big)$$

$$\rho_t \stackrel{\text{def}}{=} \min \left(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)} \right)$$

- In the on-policy case, this is an n-step backup
- In the tabular off-policy case, converges to the value of:

$$\pi_{\bar{\rho}}(a|x) \stackrel{\text{def}}{=} \frac{\min \big(\bar{\rho}\mu(a|x), \pi(a|x) \big)}{\sum_{b \in A} \min \big(\bar{\rho}\mu(b|x), \pi(b|x) \big)},$$

V-trace results: DMLab



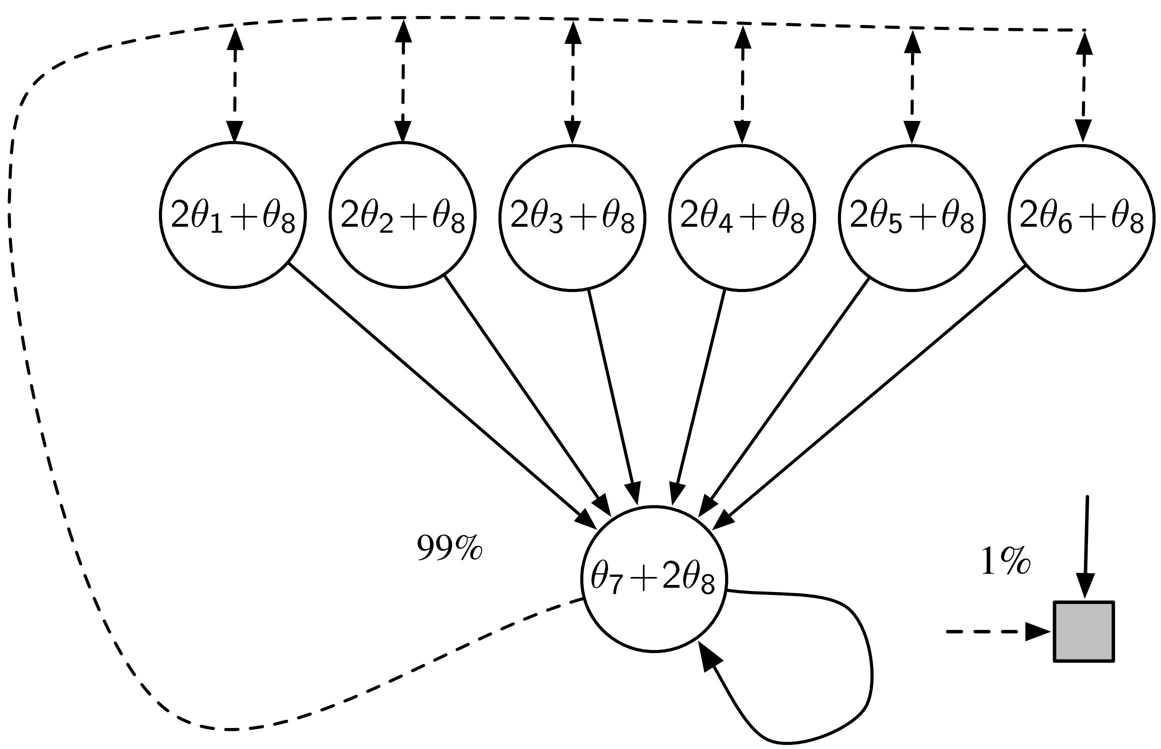
Off-policy is much harder with Function Approximation

- Even linear FA
- Even for prediction (two fixed policies π and μ)
- Even for Dynamic Programming
- The deadly triad: FA, TD, off-policy
  - Any two are OK, but not all three
  - With all three, we may get instability (elements of θ may increase to $\pm\infty$)

Two Off-Policy Learning Problems

- The easy problem is that of off-policy targets (future)
  - Use importance sampling in the target
- The hard problem is that of the distribution of states to update (present): we are no longer updating according to the on-policy distribution

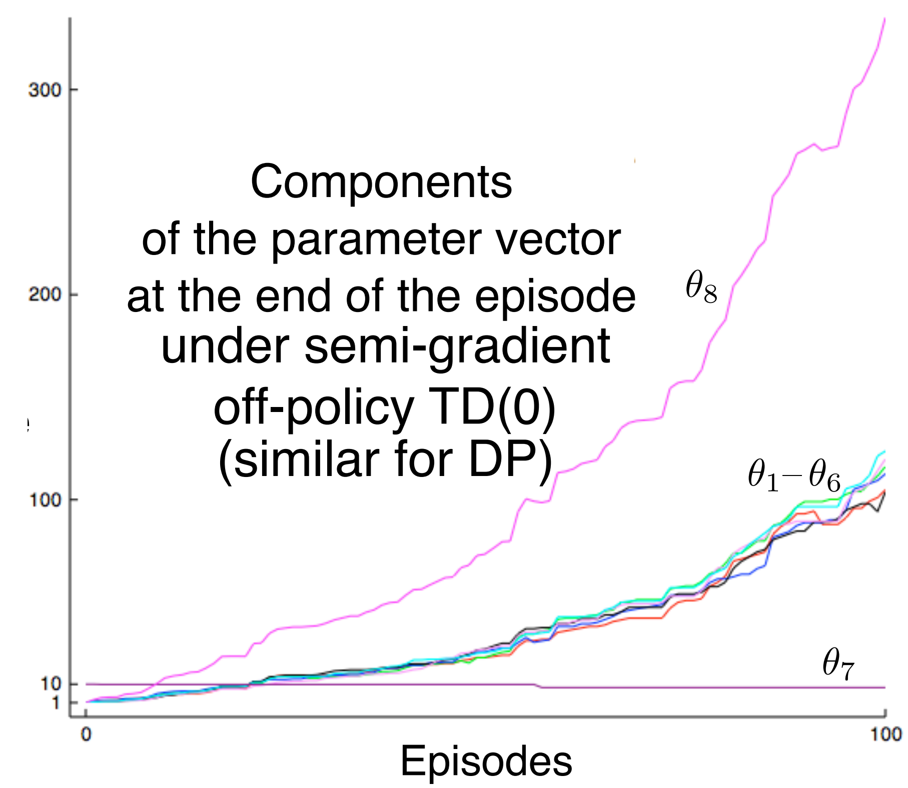
Baird's counterexample



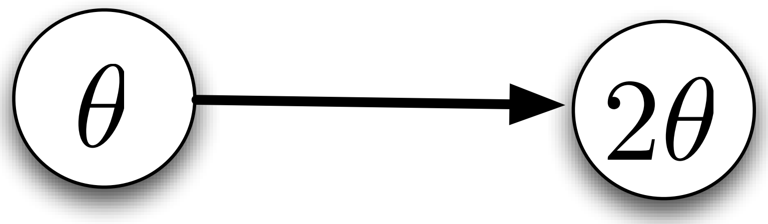
$\pi(\text{solid}|\cdot) = 1$

$\mu(\text{dashed}|\cdot) = 6/7$

$\mu(\text{solid}|\cdot) = 1/7$



TD(0) can diverge: A simple example



$$\begin{aligned}
\delta &= r + \gamma\theta^\top\phi' - \theta^\top\phi \\
&= 0 + 2\theta - \theta \\
&= \theta
\end{aligned}$$

TD update: $\Delta\theta = \alpha\delta\phi$

$= \alpha\theta$ Diverges!

TD fixpoint: $\theta^* = 0$

What causes the instability?

- It has nothing to do with learning or sampling
  - Even dynamic programming suffers from divergence with FA
- It has nothing to do with exploration, greedification, or control
  - Even prediction alone can diverge
- It has nothing to do with local minima or complex non-linear approximators
  - Even simple linear approximators can produce instability

The deadly triad

- The risk of divergence arises whenever we combine three things:
  - Function approximation
    - significantly generalizing from large numbers of examples
  - Bootstrapping
    - learning value estimates from other value estimates, as in dynamic programming and temporal-difference learning
  - Off-policy learning
    - learning about a policy from data not due to that policy, as in Q-learning, where we learn about the greedy policy from data with a necessarily more exploratory policy

How to survive the deadly triad

- Least-squares methods like off-policy LSTD(λ) (Yu 2010, Mahmood et al. 2015, Bradtke & Barto 1996, Boyan 2000) computational costs scale with the *square* of the number of parameters
- True-gradient RL methods (Gradient-TD and proximal-gradient-TD) (Maei et al, 2011, Mahadevan et al, 2015)
- Emphatic-TD methods (Sutton, White & Mahmood 2015, Yu 2015). These semi-gradient methods attain stability through an extension of the early on-policy theorems

Linear Least-Squares

- At minimum of $LS(\mathbf{w})$, the expected update must be zero

$$
\begin{aligned}
\mathbb{E}_{\mathcal{D}}\left[\Delta \mathbf{w}\right] &= 0 \\
\alpha \sum_{t=1}^{T} \mathbf{x}(s_t)(v_t^\pi - \mathbf{x}(s_t)^\top \mathbf{w}) &= 0 \\
\sum_{t=1}^{T} \mathbf{x}(s_t) v_t^\pi &= \sum_{t=1}^{T} \mathbf{x}(s_t)\mathbf{x}(s_t)^\top \mathbf{w} \\
\mathbf{w} &= \left(\sum_{t=1}^{T} \mathbf{x}(s_t)\mathbf{x}(s_t)^\top \right)^{-1} \sum_{t=1}^{T} \mathbf{x}(s_t) v_t^\pi
\end{aligned}
$$

- For N features, direct solution time is $O(N^3)$
- Incremental solution time is $O(N^2)$ using Shermann-Morrison

LSTD

- We do not know true values v_t^π
- In practice, our "training data" must use noisy or biased samples of v_t^π

  LSMC Least Squares Monte-Carlo uses return
  $v_t^\pi \approx G_t$

  LSTD Least Squares Temporal-Difference uses TD target
  $v_t^\pi \approx R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w})$

  LSTD(λ) Least Squares TD(λ) uses λ-return
  $v_t^\pi \approx G_t^\lambda$

- In each case solve directly for fixed point of MC / TD / TD(λ)

Convergence Properties

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✓	✗
	LSTD	✓	✓	-
Off-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✗	✗
	LSTD	✓	✓	-

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
LSPI	✓	(✓)	-

(✓) = chatters around near-optimal value function

Proximal Gradient (Touati et al, 2018)

Given: target policy π, behavior policy μ

Initialize θ_0 and ω_0

for n = 0 $\ldots$ **do**

set $e_0 = 0$

for k = 0 $\ldots$ end of episode **do**

Observe s_k, a_k, r_k, s_{k+1} according to μ

Update traces

$e_k = \lambda\gamma\kappa(s_k, a_k)e_{k-1} + \phi(s_k, a_k)$

Update parameters

$\delta_k = r_k + \gamma\theta_k^\top \mathbb{E}_\pi \phi(s_{k+1}, .) - \theta_k^\top \phi(s_k, a_k)$

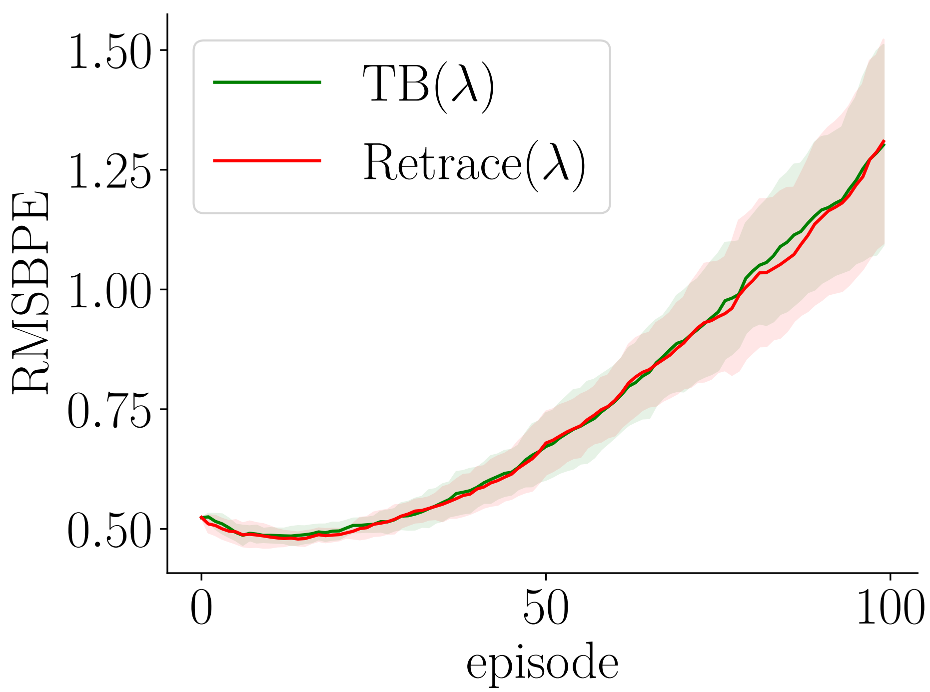
$\omega_{k+1} = \omega_k + \eta_k \left(\delta_k e_k - \omega_k^\top \phi(s_k, a_k)\phi(s_k, a_k)\right)$

$\theta_{k+1} = \theta_k - \alpha_k \omega_k^\top e_k \left(\gamma \mathbb{E}_\pi \phi(s_{k+1}, .) - \phi(s_k, a_k)\right)$

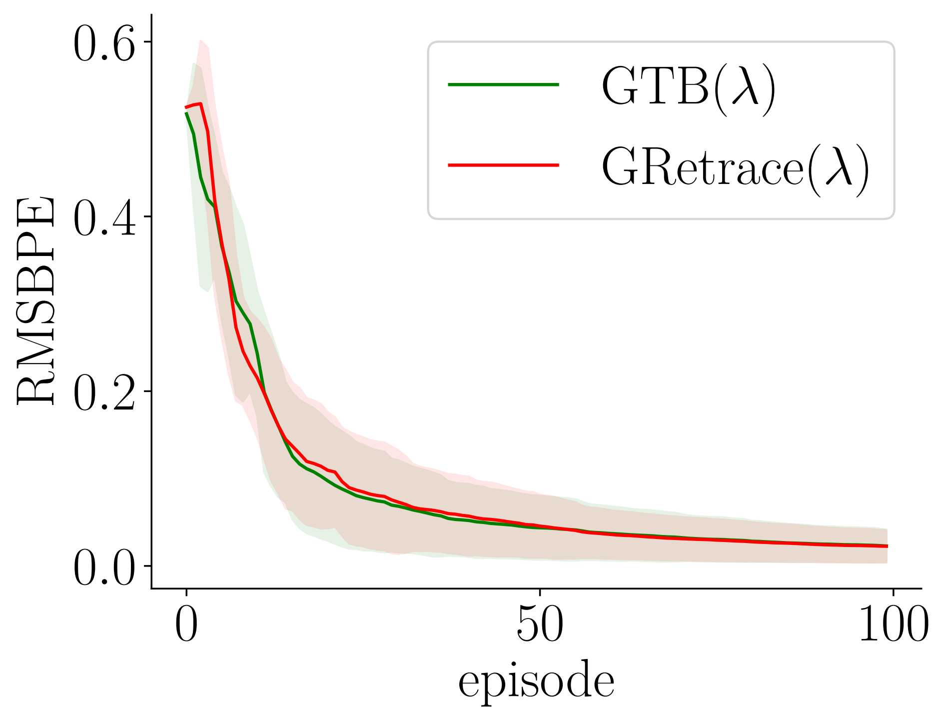
end for

end for

Results



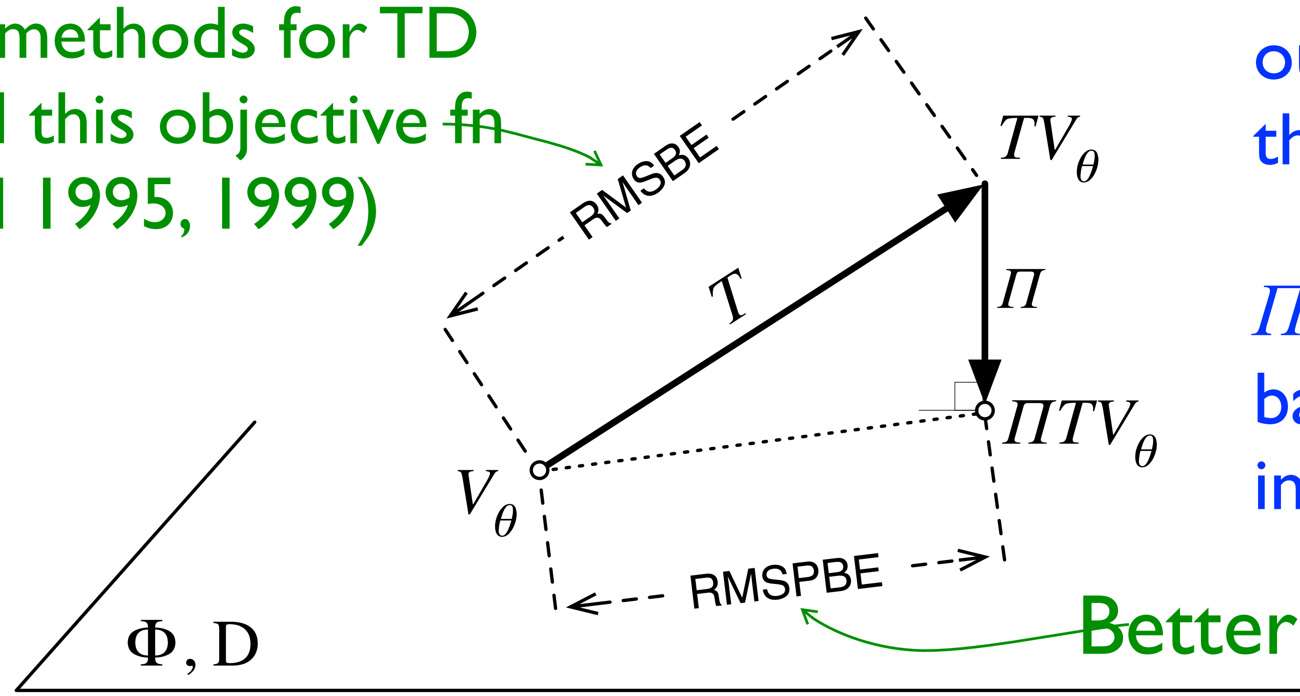
TB(λ)
Retrace(λ)
RMSBPE
1.50
1.25
1.00
0.75
0.50
0
50
100
episode



GTB(λ)
GRetrace(λ)
RMSBPE
0.6
0.4
0.2
0.0
0
50
100
episode

Value function geometry

Previous work on gradient methods for TD minimized this objective fn (Baird 1995, 1999)



The space spanned by the feature vectors, weighted by the state visitation distribution $D = \mathrm{diag}(d)$

T takes you outside the space

Π projects you back into it

Better objective fn?

$V_\theta = \Pi T V_\theta$

Is the TD fix-point

Mean Square *Projected* Bellman Error (MSPBE)

Gradient-Based TD

- Bootstraps (genuine TD)
- Works with linear function approximation (stable, reliably convergent)
- Is simple, like linear TD — O(n)
- Learns fast, like linear TD
- Can learn off-policy
- Learns from online causal trajectories (no repeat sampling from the same state)

TD is not the gradient of anything

TD(0) algorithm:

$$\Delta\theta = \alpha\delta\phi$$

$$\delta = r + \gamma\theta^\top\phi' - \theta^\top\phi$$

Assume there is a J such that:

$$\frac{\partial J}{\partial\theta_i} = \delta\phi_i$$

Then look at the second derivative:

$$\left.\begin{aligned}\frac{\partial^2 J}{\partial\theta_j\partial\theta_i} &= \frac{\partial(\delta\phi_i)}{\partial\theta_j} = (\gamma\phi'_j - \phi_j)\phi_i \\ \frac{\partial^2 J}{\partial\theta_i\partial\theta_j} &= \frac{\partial(\delta\phi_j)}{\partial\theta_i} = (\gamma\phi'_i - \phi_i)\phi_j\end{aligned}\right\} \quad \frac{\partial^2 J}{\partial\theta_j\partial\theta_i} \neq \frac{\partial^2 J}{\partial\theta_i\partial\theta_j}$$

Contradiction!

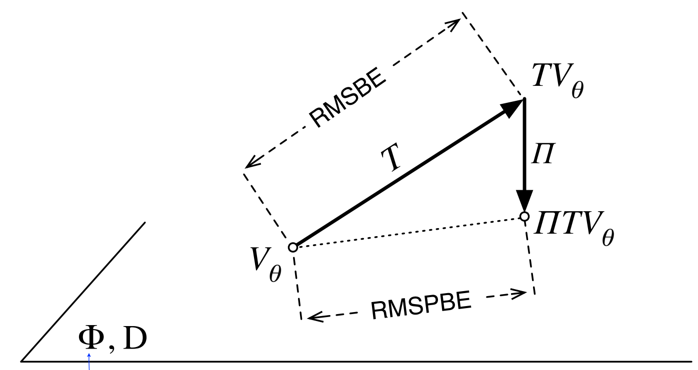
Real 2nd derivatives must be symmetric

Etienne Barnard 199

The Gradient-TD Family of Algorithms

- True gradient-descent algorithms in the Projected Bellman Error
- GTD(λ) and GQ(λ), for learning V and Q
- Solve two open problems:
  - convergent linear-complexity off-policy TD learning
  - convergent non-linear TD
- Extended to control variate, proximal forms by Mahadevan et al.

First relate the geometry to the iid statistics



matrix of the feature vectors for all states

$$\Pi = \Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D$$

$$\Phi^T D(TV_\theta - V_\theta) = \mathbb{E}[\delta\phi]$$

$$\Phi^T D\Phi = \mathbb{E}[\phi\phi^T]$$

$$\begin{aligned}
&\mathrm{MSPBE}(\theta) \\
&= \| V_\theta - \Pi T V_\theta \|_D^2 \\
&= \| \Pi(V_\theta - TV_\theta) \|_D^2 \\
&= (\Pi(V_\theta - TV_\theta))^\top D(\Pi(V_\theta - TV_\theta)) \\
&= (V_\theta - TV_\theta)^\top \Pi^\top D \Pi (V_\theta - TV_\theta) \\
&= (V_\theta - TV_\theta)^\top D^\top \Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D(V_\theta - TV_\theta) \\
&= (\Phi^\top D(TV_\theta - V_\theta))^\top (\Phi^\top D\Phi)^{-1}\Phi^\top D(TV_\theta - V_\theta) \\
&= \mathbb{E}[\delta\phi]^\top\, \mathbb{E}\left[\phi\phi^\top\right]^{-1} \mathbb{E}[\delta\phi]\,.
\end{aligned}$$

Derivation of the TDC algorithm

$$
\begin{tikzcd}
s \arrow[r, "r"] \arrow[d] & s' \arrow[d] \\
\phi & \phi'
\end{tikzcd}
$$

$$
\begin{aligned}
\Delta\theta = -\frac{1}{2}\alpha\nabla_\theta J(\theta) &= -\frac{1}{2}\alpha\nabla_\theta \parallel V_\theta - \Pi T V_\theta \parallel_D^2 \\
&= -\frac{1}{2}\alpha\nabla_\theta \left(\mathbb{E}\left[\delta\phi\right]\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right]\right) \\
&= -\alpha\left(\nabla_\theta\mathbb{E}\left[\delta\phi\right]\right)\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&= -\alpha\mathbb{E}\left[\nabla_\theta[\phi\left(r + \gamma\phi'^\top\theta - \phi^\top\theta\right)]\right]\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&= -\alpha\mathbb{E}\left[\phi\left(\gamma\phi' - \phi\right)^\top\right]^\top\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&= -\alpha\left(\gamma\mathbb{E}\left[\phi'\phi^\top\right] - \mathbb{E}\left[\phi\phi^\top\right]\right)\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&= \alpha\mathbb{E}\left[\delta\phi\right] - \alpha\gamma\mathbb{E}\left[\phi'\phi^\top\right]\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}\left[\delta\phi\right] \\
&\approx \alpha\mathbb{E}\left[\delta\phi\right] - \alpha\gamma\mathbb{E}\left[\phi'\phi^\top\right]w \\
\text{(sampling)} &\approx \alpha\delta\phi - \alpha\gamma\phi'\phi^\top w
\end{aligned}
$$

This is the trick!

$w \in \Re^n$ is a second set of weights

TD with gradient correction (TDC) algorithm

- on each transition

aka GTD(0)

- update two parameters

$$\begin{tikzcd} s \arrow[r, "r"] \arrow[d] & s' \arrow[d] \\ \phi & \phi' \end{tikzcd}$$

- where, as usual

TD(0)

with gradient correction

$$\theta \leftarrow \theta + \boxed{\alpha\delta\phi} - \boxed{\alpha\gamma\phi'\,(\phi^\top w)}$$

$$w \leftarrow w + \beta(\delta - \phi^\top w)\phi$$

estimate of the TD error (δ) for the current state ϕ

$$\delta = r + \gamma\theta^\top\phi' - \theta^\top\phi$$

Convergence theorems

- All algorithms converge w.p.1 to the TD fix-point:

$$\mathbb{E}[\delta\phi] \longrightarrow 0$$

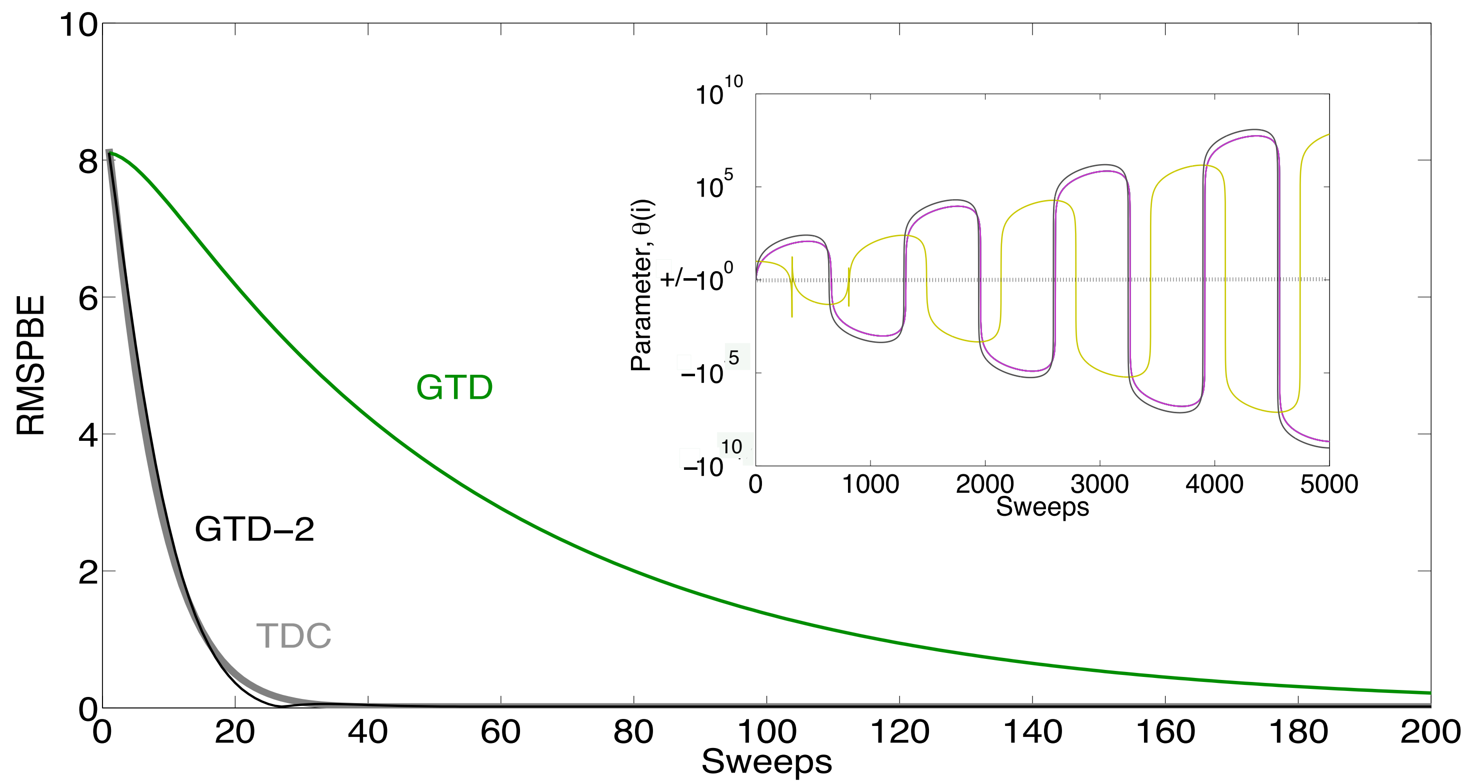
- GTD, GTD-2 converges at one time scale

$$\alpha = \beta \longrightarrow 0$$

- TD-C converges in a two-time-scale sense

$$\alpha, \beta \longrightarrow 0 \qquad \frac{\alpha}{\beta} \longrightarrow 0$$

Off-policy result: Baird's counter-example



Gradient algorithms converge. TD diverges.

A little more theory

$$
\begin{aligned}
\Delta\theta \propto \delta\phi &= \left(r + \gamma\theta^\top\phi' - \theta^\top\phi\right)\phi \\
&= \theta^\top\left(\gamma\phi' - \phi\right)\phi + r\phi \\
&= \phi\left(\gamma\phi' - \phi\right)^\top\theta + r\phi \\
\mathbb{E}\left[\Delta\theta\right] &\propto -\mathbb{E}\left[\phi\left(\phi - \gamma\phi'\right)^\top\right]\theta + \mathbb{E}\left[r\phi\right]
\end{aligned}
$$

$$
\mathbb{E}\left[\Delta\theta\right] \propto \quad - A\theta \quad + \quad b
$$

convergent if A is pos. def.

therefore, at the TD fixpoint:

$$
\begin{aligned}
A\theta^* &= b \\
\theta^* &= A^{-1}b
\end{aligned}
$$

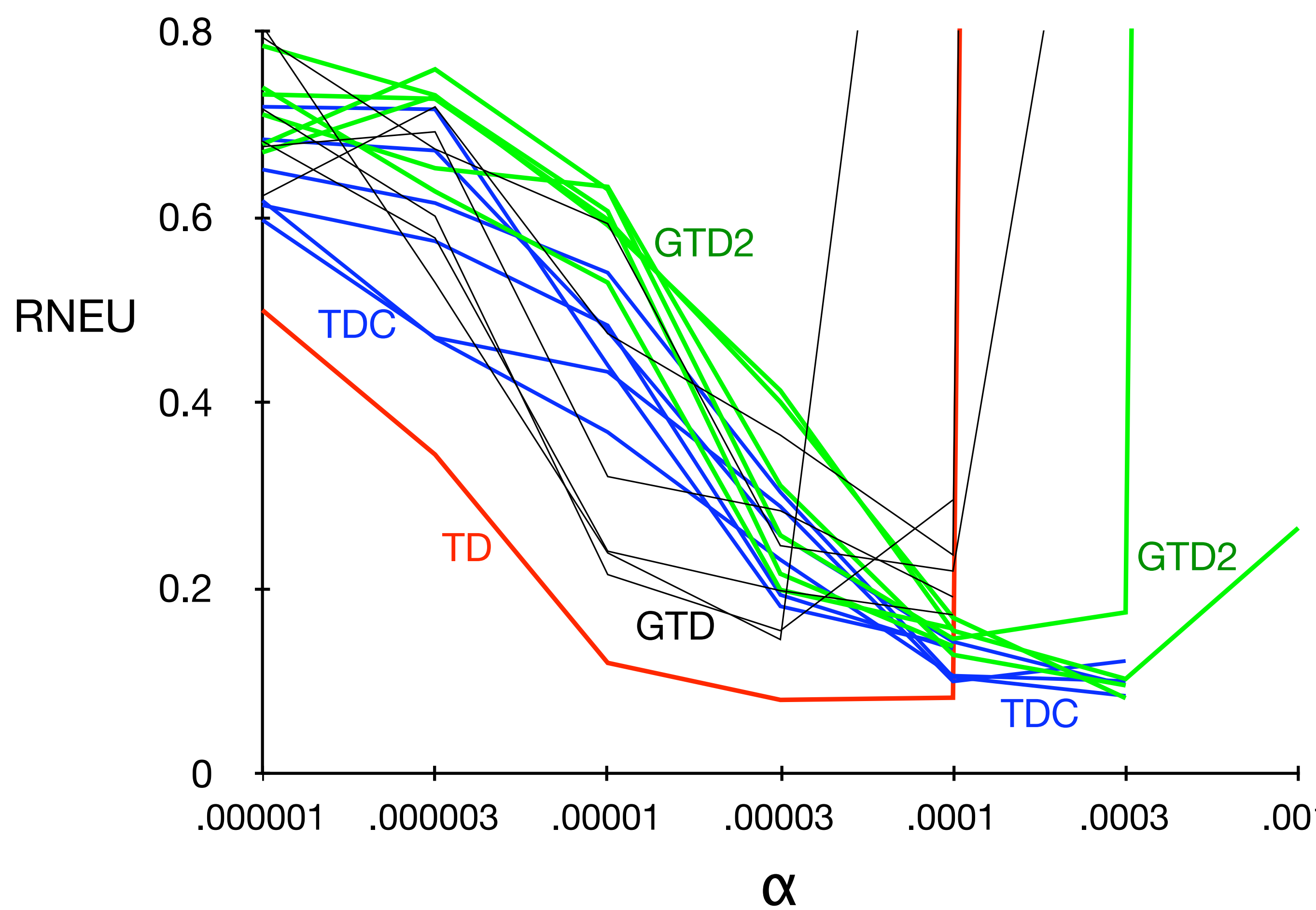
LSTD computes this directly

$$
-\frac{1}{2}\nabla_\theta \mathrm{MSPBE} = -\underbrace{A^\top C^{-1}(A}_{\text{always pos. def.}}\theta - b)
$$

$C = \mathbb{E}\left[\phi\phi^\top\right]$ covariance matrix

Example: Go

- Learn a linear value function (probability of winning) for 9x9 Go from self play
- One million features, each corresponding to a template on a part of the Go board



Summary

ISSUE \ ALGORITHM	TD(λ), Sarsa(λ)	Approx. DP	LSTD(λ), LSPE(λ)	Fitted-Q	Residual gradient	GDP	GTD(λ), GQ(λ)
Linear computation	✓	✓	✖	✖	✓	✓	✓
Nonlinear convergent	✖	✖	✖	✓	✓	✓	✓
Off-policy convergent	✖	✖	✓	✖	✓	✓	✓
Model-free, online	✓	✖	✓	✖	✓	✖	✓
Converges to PBE = 0	✓	✓	✓	✓	✖	✓	✓

Off-Policy with TD and FA is still Challenging

- Gradient TD, proximal gradient TD, and hybrids
- Emphatic TD (Ask Rupam about this!)
- Higher λ (less TD)
- Better state rep'ns (less FA)
- Recognizers (less off-policy)
- LSTD ($O(n^2)$ methods)

Emphatic temporal-difference learning

- Rupam Mahmood, Huizhen (Janey) Yu, Martha White, Rich Sutton

- Reinforcement Learning and Artificial Intelligence Laboratory
- Department of Computing Science
- University of Alberta
- Canada

State weightings are important, powerful, even magical,

- They are the difference between convergence and divergence in on-policy and off-policy TD learning
- They are needed to make the problem well-defined
- We can change the weighting by *emphasizing* some steps more than others in learning

Often some time steps are more important

- Early time steps of an *episode* may be more important
  - Because of *discounting*
  - Because the control objective is to maximize the value of the *starting state*
- In general, function approximation resources are limited
  - Not all states can be accurately valued
  - The accuracy of different state must be traded off!
  - You may want to control the tradeoff

Bootstrapping interacts with state importance

- In the Monte Carlo case ($\lambda=1$) the values of different states (or time steps) are estimated independently,
  and their importances can be assigned independently
- But with bootstrapping ($\lambda<1$) each state's value is estimated based on the estimated values of later states; if the state is important, then it becomes important to accurately value the later states even if they are not important on their own

Two kinds of importance

- Intrinsic and derived, primary and secondary
  - The one you specify, and the one that follows from it because of bootstrapping
- Our terms: *Interest* and *Emphasis*
  - Your intrinsic *interest* in valuing accurately on a time step
  - The total resultant *emphasis* that you place on each time step

- Data

$\phi : \mathcal{S} \to \Re^n$
feature function

$$\cdots \ \boldsymbol{\phi}(S_t) \ A_t \ R_{t+1} \ \boldsymbol{\phi}(S_{t+1}) \ A_{t+1} \ R_{t+2} \ \cdots$$

- State distribution

Problem

behavior policy

$$d_\mu(s) = \lim_{t\to\infty} \Pr\big[S_t = s \;\big|\; A_{0:t-1} \sim \mu\big]$$

- Objective to minimize

parameter vector

true value function

transpose (inner product)

$$\mathrm{MSE}(\boldsymbol{\theta}) = \sum_{s\in\mathcal{S}} d_\mu(s) i(s) \Big(v_\pi(s) - \boldsymbol{\theta}^\top \boldsymbol{\phi}(s)\Big)^2$$

interest function
$i : \mathcal{S} \to \Re^+$

target policy

Solution

- Emphatic TD(0)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha M_t \rho_t \left(R_{t+1} + \gamma \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t\right) \boldsymbol{\phi}_t$$

emphasis
$M_t > 0$

importance sampling ratio

$\rho_t = \dfrac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \quad \mathbb{E}[\rho_t] = 1$

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t)$$

$$\mathbf{A}_t = \sum_{k=0}^{t} M_k \rho_k \boldsymbol{\phi}_k \big(\boldsymbol{\phi}_k - \gamma \boldsymbol{\phi}_{k+1}\big)^\top \quad \mathbf{b}_t = \sum_{k=1}^{t} M_k \rho_k R_k \boldsymbol{\phi}_k$$

Real-time off-policy prediction learning with linear function approximation

$\boldsymbol{\phi} : \mathcal{S} \to \Re^n$
feature function

$$\cdots\ \boldsymbol{\phi}(S_t)\ A_t\ R_{t+1}\ \boldsymbol{\phi}(S_{t+1})\ A_{t+1}\ R_{t+2}\ \cdots$$

- Data

Problem

behavior policy

$$d_\mu(s) = \lim_{t\to\infty} \Pr\big[S_t = s \mid A_{0:t-1} \sim \mu\big]$$

- State distribution

parameter vector, true value function, transpose (inner product)

$$\mathrm{MSE}(\boldsymbol{\theta}) = \sum_{s\in\mathcal{S}} d_\mu(s)\, i(s) \Big(v_\pi(s) - \boldsymbol{\theta}^\top \boldsymbol{\phi}(s)\Big)^2$$

interest function
$i : \mathcal{S} \to \Re^+$
target policy

- Objective to minimize

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha M_t \rho_t \left(R_{t+1} + \gamma \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t\right) \boldsymbol{\phi}_t$$

emphasis
$M_t > 0$

importance sampling ratio

$$\rho_t = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \quad \mathbb{E}[\rho_t] = 1$$

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t)$$

Solution

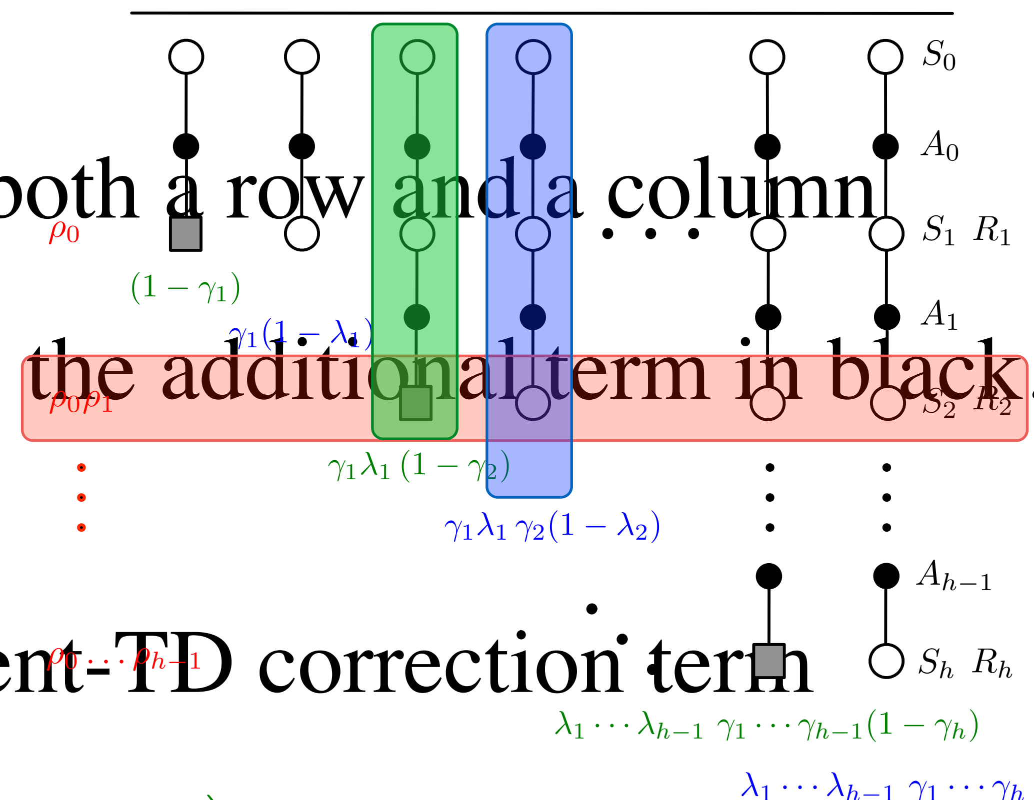
- Emphatic TD(0)

$$\mathbf{A}_t = \sum_{k=0}^{t} M_k \rho_k \boldsymbol{\phi}_k \big(\boldsymbol{\phi}_k - \gamma \boldsymbol{\phi}_{k+1}\big)^\top \qquad \mathbf{b}_t = \sum_{k=1}^{t} M_k \rho_k R_k \boldsymbol{\phi}_k$$

$$\boldsymbol{\theta}_{t+1} = \mathbf{A}_t^{-1} \mathbf{b}_t$$

True online GTD(λ) forward view

- A 'matrix' backup diagram
  - weights are drawn from both a row and a column



- Not shown in the diagram is the additional term in black, which has expected value 0
- Also not shown is the gradient-TD correction term

$$Z_t^h = G_{t,h}^{\lambda\rho} = \sum_{i=t+1}^{j+1}\sum_{j=t}^{h-1}\left[\left(\prod_{m=t}^{i-1}\rho_m\right)\left(\prod_{m=t+1}^{j}\gamma_m\lambda_m(1-\gamma_{j+1})\right)R_i + \left(\prod_{m=t}^{i-1}\rho_m\right)\left(\prod_{m=t+1}^{j}\gamma_m\lambda_m\gamma_{j+1}(1-\lambda_{j+1}^h)\right)\left(R_i + \mathbf{1}_{\{i=j\}}\phi_{i+1}^\top\theta_i\right)\right] + \sum_{j=t}^{h-1}\left(\prod_{m=t}^{j}\rho_m\right)\left(\prod_{m=t}^{j+1}\gamma_m\lambda_m\right)(1-\rho_{j+1})\phi_{j+1}^\top\theta_j$$

$$\rho_t = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$$

Emphasis algorithm

(Sutton, Mahmood & White 2015)

- Derived from analysis of general bootstrapping relationships (Sutton, Mahmood, Precup & van Hasselt 2014)
- Emphasis is a scalar signal

$$M_t \geq 0$$

- Defined from a new scalar *followon trace*

$$M_t = \lambda_t \, i(S_t) + (1 - \lambda_t) F_t$$

$$F_t \geq 0$$

$$F_t = \rho_{t-1} \gamma_t F_{t-1} + i(S_t)$$

Off-policy implications

- The emphasis weighting is *stable under off-policy TD(λ)* (like the on-policy weighting) (Sutton, Mahmood & White 2015)
  - It is the *followon* weighting, from the interest weighted behavior distribution ($d_\mu(s)i(s)$), under the target policy
- Learning is *convergent* (though not necessarily of finite variance) under the emphasis weighting for arbitrary target and behavior policies (with coverage) (Yu 2015)
- There are error bounds analogous to those for on-policy TD(λ) (Munos)
- Emphatic TD is the simplest convergent off-policy TD algorithm (one parameter, one learning rate)