Zap Q-Learning

Fastest Convergent Q-Learning

Vistas in Control

.IfA AUTOMATIC CONTROL LABORATORY

50th birthday !

September 10-11, 2018

Sean Meyn

COGNITION & CONTROL IN COMPLEX SYSTEMS

Department of Electrical and Computer Engineering — University of Florida

Based on joint research with Adithya M. Devraj and Ana Bušić + ...

Thanks to to the National Science Foundation

Background and Goals

Reinforcement Learning for Control Scientists

Approximate Dynamic Programming

Seminal paper, Watkins & Dayan, Q-learning, 1992:

compute optimal policy for an MDP

Approximate Dynamic Programming

Seminal paper, Watkins & Dayan, Q-learning, 1992:

compute optimal policy for an MDP

Many papers since, and rich theory developed by Tsitsiklis et. al.

Bertsekas & Tsitsiklis, NDP, 1996

Approximate Dynamic Programming

Seminal paper, Watkins & Dayan, Q-learning, 1992:

compute optimal policy for an MDP

Many papers since, and rich theory developed by Tsitsiklis et. al.

Bertsekas & Tsitsiklis, NDP, 1996

Magic ingredient may look familiar ...

Consider the discounted cost optimal control problem:

$$\frac{d}{dt} x_t = f(x_t, u_t)\,, \qquad J^*(x) = \min_{\boldsymbol{u}} \int_0^\infty e^{-\gamma t} c(x_t, u_t)\, dt$$

Approximate Dynamic Programming

Magic ingredient may look familiar ...

Consider the discounted cost optimal control problem:

$$\frac{d}{dt}x_t = f(x_t, u_t)\,, \qquad J^*(x) = \min_{\boldsymbol{u}} \int_0^\infty e^{-\gamma t} c(x_t, u_t)\, dt$$

Magic: Hamiltonian and HJB equation

$$\min_u \{\underbrace{c(x,u) + f(x,u)\cdot \nabla J^*\,(x)}_{\text{Q-function}}\} = \gamma J^*(x)$$

Often easier to estimate Q^* rather than the value function J^*

Q-learning in control language

System: $\dot{x} = f(x,u)$ cost: $c(x,u)$

Magic: Hamiltonian and HJB equation

$$\min_u \{ \underbrace{c(x,u) + f(x,u) \cdot \nabla J^*(x)} \} = \gamma J^*(x)$$

Q-learning in control language

System: $\dot{x} = f(x,u)$ cost: $c(x,u)$

Magic: Hamiltonian and HJB equation

$$\min_u \{\underbrace{c(x,u) + f(x,u) \cdot \nabla J^* (x)}_{\text{Q-function}}\} = \gamma J^*(x)$$

$$Q^*(x,u) = c(x,u) + f(x,u) \cdot \nabla J^*(x)$$

Q-learning in control language

System: $\dot{x} = f(x,u)$ cost: $c(x,u)$

Magic: Hamiltonian and HJB equation

$$\min_u \{\underbrace{c(x,u) + f(x,u) \cdot \nabla J^*(x)}_{\text{Q-function}}\} = \gamma J^*(x)$$

Denote $\underline{Q}^* = \min_u Q^*(x,u) = \gamma J^*(x)$

$\Longrightarrow$ Fixed point equation for Q-function

$$Q^*(x,u) = c(x,u) + f(x,u) \cdot \nabla J^*(x)$$

Q-learning in control language

System: $\dot{x} = f(x,u)$ cost: $c(x,u)$

Magic: Hamiltonian and HJB equation

$$\min_u \{ \underbrace{c(x,u) + f(x,u) \cdot \nabla J^* \,(x)}_{\text{Q-function}} \} = \gamma J^*(x)$$

Denote $\underline{Q}^* = \min_u Q^*(x,u) = \gamma J^*(x)$

$\Longrightarrow$ Fixed point equation for Q-function

$$Q^*(x,u) = c(x,u) + f(x,u) \cdot \nabla J^*(x) = c(x,u) + \gamma^{-1} f(x,u) \cdot \nabla \underline{Q}^* \,(x)$$

Q-learning in control language

System: $\dot{x} = f(x,u)$ cost: $c(x,u)$

Magic: Hamiltonian and HJB equation

$$\min_u \{\underbrace{c(x,u) + f(x,u)\cdot \nabla J^*(x)}_{\text{Q-function}}\} = \gamma J^*(x)$$

Denote $\underline{Q}^* = \min_u Q^*(x,u) = \gamma J^*(x)$

$\Longrightarrow$ Fixed point equation for Q-function

$$Q^*(x,u) = c(x,u) + f(x,u)\cdot\nabla J^*(x) = c(x,u) + \gamma^{-1} f(x,u)\cdot\nabla \underline{Q}^*(x)$$

Parameterization set of approximations, $\{Q^\theta(x,u) : \theta \in \mathbb{R}^d\}$

Bellman error: $\mathcal{E}^\theta(x,u) = \gamma[Q^\theta(x,u) - c(x,u)] - f(x,u)\cdot\nabla \underline{Q}^\theta(x)$

Q-learning in control language

System: $\dot{x} = f(x,u)$ cost: $c(x,u)$

Apply Magic:

$$
\begin{aligned}
\mathcal{E}^\theta(x,u) &= \gamma[Q^\theta(x,u) - c(x,u)] - f(x,u)\cdot\nabla \underline{Q}^\theta\,(x) \\
&= \gamma[Q^\theta(x,u) - c(x,u)] - \frac{d}{dt}\underline{Q}^\theta\,(x_t)\Big|_{x=x(t),\ u=u(t)}
\end{aligned}
$$

Q-learning in control language

System: $\dot{x} = f(x,u)$ cost: $c(x,u)$

Apply Magic: Model Free Error Representation

$$\mathcal{E}^\theta(x,u) = \gamma[Q^\theta(x,u) - c(x,u)] - f(x,u) \cdot \nabla \underline{Q}^\theta(x)$$

$$= \gamma[Q^\theta(x,u) - c(x,u)] - \frac{d}{dt}\underline{Q}^\theta(x_t)\Big|_{x=x(t),\ u=u(t)}$$

Q-learning in control language

System: $\dot{x} = f(x, u)$ cost: $c(x, u)$

Apply Magic: Model Free Error Representation

$$\mathcal{E}^\theta(x,u) = \gamma[Q^\theta(x,u) - c(x,u)] - f(x,u) \cdot \nabla \underline{Q}^\theta\,(x)$$
$$= \gamma[Q^\theta(x,u) - c(x,u)] - \frac{d}{dt}\underline{Q}^\theta\,(x_t)\Big|_{x=x(t),\ u=u(t)}$$

Q-learning and Quasi Stochastic Approximation

- Find zeros of $\overline{h}(\theta) = \nabla \mathsf{E}[\mathcal{E}^\theta(x_\infty, u_\infty)^2]$ using QSA
- (x_∞, u_∞) ergodic steady-state.
- Choose input: stable feedback + mixture of sinusoids,
$$u(t) = -k(x(t)) + \omega(t)$$

[9, 8]

Q-learning in control language

System: $\dot{x} = f(x, u)$ cost: $c(x, u)$

Apply Magic: Model Free Error Representation

$$\mathcal{E}^\theta(x,u) = \gamma[Q^\theta(x,u) - c(x,u)] - f(x,u)\cdot\nabla \underline{Q}^\theta(x)$$
$$= \gamma[Q^\theta(x,u) - c(x,u)] - \frac{d}{dt}\underline{Q}^\theta(x_t)\Big|_{x=x(t),\ u=u(t)}$$

Q-learning and Quasi Stochastic Approximation

- Find zeros of $\overline{h}(\theta) = \nabla \mathsf{E}[\mathcal{E}^\theta(x_\infty, u_\infty)^2]$ using QSA
- (x_∞, u_∞) ergodic steady-state.
- Choose input: stable feedback + mixture of sinusoids,
$$u(t) = -k(x(t)) + \omega(t)$$

[9, 8]

Alert! Approximation based on online input-output measurements!

Q-learning in practice

Questions to be addressed

- How to select function class $\{Q^\theta\}$?

Q-learning in practice

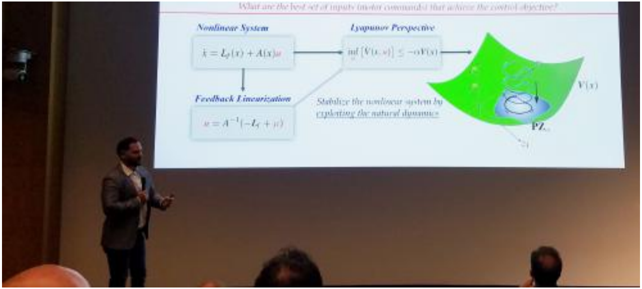
Questions to be addressed

- How to select function class $\{Q^\theta\}$? Go to CDC!

Q-learning in practice

Questions to be addressed

- How to select function class $\{Q^\theta\}$? Go to CDC!
  - Controlled Lyapunov functions from Monday morning



Q-learning in practice

Questions to be addressed

- How to select function class $\{Q^\theta\}$? Go to CDC!
  - Controlled Lyapunov functions from Monday morning
  - Singular perturbation approximate models, and related

Q-learning in practice

Questions to be addressed

- How to select function class $\{Q^\theta\}$? Go to CDC!
  - Controlled Lyapunov functions from Monday morning
  - Singular perturbation approximate models, and related
  - Fluid models for networks, and their workload relaxations [CTCN]

Q-learning in practice

Questions to be addressed

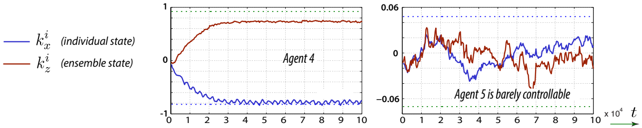
- How to select function class $\{Q^\theta\}$? Go to CDC!
    - Controlled Lyapunov functions from Monday morning
    - Singular perturbation approximate models, and related
    - Fluid models for networks, and their workload relaxations [CTCN]
    - Mean field game solutions for multi-agent systems

Q-learning in practice

Questions to be addressed

- How to select function class $\{Q^\theta\}$? Go to CDC!
  - Controlled Lyapunov functions from Monday morning
  - Singular perturbation approximate models, and related
  - Fluid models for networks, and their workload relaxations [CTCN]
  - Mean field game solutions for multi-agent systems

Feature selection for neuro-dynamic programming, 2011 [8]



Mean-field game used for basis construction for Q-learning
Resulting estimates are consistent with MFG solution

Q-learning in practice

Questions to be addressed

- How to select function class $\{Q^\theta\}$? Go to CDC!
  - Controlled Lyapunov functions from Monday morning
  - Singular perturbation approximate models, and related
  - Fluid models for networks, and their workload relaxations [CTCN]
  - Mean field game solutions for multi-agent systems

Feature selection for neuro-dynamic programming, 2011 [8]

- How to make an algorithm work?

Q-learning in practice

Questions to be addressed

- How to select function class $\{Q^\theta\}$? Go to CDC!
  - Controlled Lyapunov functions from Monday morning
  - Singular perturbation approximate models, and related
  - Fluid models for networks, and their workload relaxations [CTCN]
  - Mean field game solutions for multi-agent systems

Feature selection for neuro-dynamic programming, 2011 [8]

- How to make an algorithm work?

Let's get started ...

$$\boxed{\mathsf{E}[f(\theta, W)]\Big|_{\theta=\theta^*} \quad = \quad 0}$$

Stochastic Approximation

What is Stochastic Approximation?

A simple goal: Find the solution θ^* to

$$\bar{f}(\theta^*) := \mathsf{E}[f(\theta, W)]\Big|_{\theta=\theta^*} = 0$$

What is Stochastic Approximation?

A simple goal: Find the solution θ^* to

$$\bar{f}(\theta^*) := \mathsf{E}[f(\theta, W)]\Big|_{\theta=\theta^*} = 0$$

What makes this hard?

What is Stochastic Approximation?

A simple goal: Find the solution θ^* to

$$\bar{f}(\theta^*) := \mathsf{E}[f(\theta, W)]\Big|_{\theta=\theta^*} = 0$$

What makes this hard?

1. The function f and the distribution of the random vector W may not be known
2. Even if everything is known, computation of the expectation may be expensive. For root finding, we may need to compute the expectation for many values of θ

What is Stochastic Approximation?

A simple goal: Find the solution θ^* to

$$\bar{f}(\theta^*) := \mathsf{E}[f(\theta, W)]\Big|_{\theta=\theta^*} = 0$$

What makes this hard?

1. The function f and the distribution of the random vector W may not be known
2. Even if everything is known, computation of the expectation may be expensive. For root finding, we may need to compute the expectation for many values of θ
3. Motivates stochastic approximation: $\theta(n+1) = \theta(n) + \alpha_n f(\theta(n), W(n))$
   The recursive algorithms we come up with are often slow, and their variance is often **infinite**

Algorithm and Convergence Analysis

Algorithm:

$$\theta(n+1) = \theta(n) + \alpha_n f(\theta(n), W(n))$$

Goal:

$$\bar{f}(\theta^*) := \mathsf{E}[f(\theta, W)]\Big|_{\theta=\theta^*} = 0$$

Interpretation: $\theta^* \equiv$ *stationary point* of the ODE

$$\frac{d}{dt}\theta(t) = \bar{f}(\theta(t))$$

Algorithm and Convergence Analysis

Algorithm:

$$\theta(n+1) = \theta(n) + \alpha_n f(\theta(n), W(n))$$

Goal:

$$\bar{f}(\theta^*) := \mathsf{E}[f(\theta, W)]\Big|_{\theta=\theta^*} = 0$$

Interpretation: $\theta^* \equiv$ *stationary point* of the ODE

$$\frac{d}{dt}\theta(t) = \bar{f}(\theta(t))$$

Analysis: Stability of the ODE $\oplus$ (See Borkar's monograph) $\Longrightarrow$

$$\lim_{n\to\infty} \theta(n) = \theta^*$$

Stochastic Approximation Example

Example: Monte-Carlo

Monte-Carlo Estimation

Estimate the mean $\eta = \mathsf{E}[c(X)]$, where random variable X has density ϱ:

$$\eta = \int c(x)\, \varrho(x)\, dx$$

Stochastic Approximation Example

Example: Monte-Carlo

Monte-Carlo Estimation

Estimate the mean $\eta = \mathsf{E}[c(X)]$

SA interpretation: Find θ^* solving $0 = \mathsf{E}[f(\theta, X)] = \mathsf{E}[c(X) - \theta]$

$$\textit{Algorithm:} \quad \theta(n) = \frac{1}{n}\sum_{i=1}^{n} c(X(i))$$

Stochastic Approximation Example

Example: Monte-Carlo

$\sum \alpha_n = \infty, \sum \alpha_n^2 < \infty$

Monte-Carlo Estimation

Estimate the mean $\eta = \mathsf{E}[c(X)]$

SA interpretation: Find θ^* solving $0 = \mathsf{E}[f(\theta, X)] = \mathsf{E}[c(X) - \theta]$

$$\textit{Algorithm:} \quad \theta(n) = \frac{1}{n}\sum_{i=1}^{n} c(X(i))$$

$$\Longrightarrow \qquad (n+1)\theta(n+1) = \sum_{i=1}^{n+1} c(X(i)) = n\theta(n) + c(X(n+1))$$

$$\Longrightarrow \qquad (n+1)\theta(n+1) = (n+1)\theta(n) + [c(X(n+1)) - \theta(n)]$$

SA Recursion: $\qquad \theta(n+1) = \theta(n) + \alpha_n f(\theta(n), X(n+1))$

Performance Criteria

Two standard approaches to evaluate performance, $\tilde{\theta}(n) := \theta(n) - \theta^*$:

1. Finite-n bound:

$$\mathsf{P}\{\|\tilde{\theta}(n)\| \geq \varepsilon\} \leq \exp(-I(\varepsilon, n)), \qquad I(\varepsilon, n) = O(n\varepsilon^2)$$

2. Asymptotic covariance:

$$\Sigma = \lim_{n\to\infty} n\mathsf{E}\Big[\tilde{\theta}(n)\tilde{\theta}(n)^{\intercal}\Big], \qquad \sqrt{n}\tilde{\theta}(n) \approx N(0, \Sigma)$$

Asymptotic Covariance

$$\Sigma = \lim_{n\to\infty} \Sigma_n = \lim_{n\to\infty} n\mathsf{E}\big[\tilde\theta(n)\tilde\theta(n)^{\intercal}\big]\,, \qquad \sqrt{n}\tilde\theta(n) \approx N(0,\Sigma)$$

SA recursion for covariance:

$$\Sigma_{n+1} \approx \Sigma_n + \tfrac{1}{n}\Big\{(A+\tfrac{1}{2}I)\Sigma_n + \Sigma_n(A+\tfrac{1}{2}I)^{\intercal} + \Sigma_\Delta\Big\}$$

$$A = \tfrac{d}{d\theta}\bar f\,(\theta^*)$$

Conclusions

1. If $\mathrm{Re}\,\lambda(A) \ge -\frac{1}{2}$ for some eigenvalue then Σ is (typically) infinite
2. If $\mathrm{Re}\,\lambda(A) < -\frac{1}{2}$ for all, then $\Sigma = \lim_{n\to\infty}\Sigma_n$ is the unique solution to the Lyapunov equation:

$$\boxed{0 = (A+\tfrac{1}{2}I)\Sigma + \Sigma(A+\tfrac{1}{2}I)^{\intercal} + \Sigma_\Delta}$$

Optimal Asymptotic Covariance

Introduce a $d \times d$ matrix gain sequence $\{G_n\}$:

$$\theta(n+1) = \theta(n) + \frac{1}{n+1} G_n f(\theta(n), X(n))$$

Optimal Asymptotic Covariance

Introduce a $d \times d$ matrix gain sequence $\{G_n\}$:

$$\theta(n+1) = \theta(n) + \frac{1}{n+1} G_n f(\theta(n), X(n))$$

Assume it converges, and linearize:

$$\tilde{\theta}(n+1) \approx \tilde{\theta}(n) + \frac{1}{n+1} G\big(A\tilde{\theta}(n) + \Delta(n+1)\big), \qquad A = \frac{d}{d\theta}\bar{f}\,(\theta^*)\,.$$

Optimal Asymptotic Covariance

Introduce a $d \times d$ matrix gain sequence $\{G_n\}$:

$$\theta(n+1) = \theta(n) + \frac{1}{n+1} G_n f(\theta(n), X(n))$$

Assume it converges, and linearize:

$$\tilde{\theta}(n+1) \approx \tilde{\theta}(n) + \frac{1}{n+1} G\big(A\tilde{\theta}(n) + \Delta(n+1)\big), \qquad A = \frac{d}{d\theta}\bar{f}\,(\theta^*)\,.$$

If $G = G^* := -A^{-1}$ then

- Resembles Monte-Carlo estimate
- Resembles Newton-Rapshon
- It is optimal: $\Sigma^* = G^* \Sigma_\Delta G^{*\,T} \leq \Sigma^G$ *any other* G

Optimal Asymptotic Covariance

Introduce a $d \times d$ matrix gain sequence $\{G_n\}$:

$$\theta(n+1) = \theta(n) + \frac{1}{n+1} G_n f(\theta(n), X(n))$$

Assume it converges, and linearize:

$$\tilde{\theta}(n+1) \approx \tilde{\theta}(n) + \frac{1}{n+1} G\big(A\tilde{\theta}(n) + \Delta(n+1)\big), \qquad A = \frac{d}{d\theta}\bar{f}\,(\theta^*)\,.$$

If $G = G^* := -A^{-1}$ then

- Resembles Monte-Carlo estimate
- Resembles Newton-Rapshon
- It is optimal: $\quad \Sigma^* = G^* \Sigma_\Delta G^{*\,T} \leq \Sigma^G \qquad$ *any other* G

Polyak-Ruppert averaging is also optimal, but first two bullets are missing.

Optimal Variance

Example: return to Monte-Carlo

$$\theta(n+1) = \theta(n) + \frac{g}{n+1}\Big(-\theta(n) + X(n+1)\Big)$$

Optimal Variance

Example: return to Monte-Carlo

$$\theta(n+1) = \theta(n) + \frac{g}{n+1}\Big(-\theta(n) + X(n+1)\Big)$$

$$\Delta(n) = X(n) - \mathsf{E}[X(n)]$$

Optimal Variance

Normalization for analysis:

$$\Delta(n) = X(n) - \mathsf{E}[X(n)]$$

$$\tilde{\theta}(n+1) = \tilde{\theta}(n) + \frac{g}{n+1}\Big(-\tilde{\theta}(n) + \Delta(n+1)\Big)$$

Optimal Variance

Normalization for analysis: $\Delta(n) = X(n) - \mathsf{E}[X(n)]$

$$\tilde{\theta}(n+1) = \tilde{\theta}(n) + \frac{g}{n+1}\Big(-\tilde{\theta}(n) + \Delta(n+1)\Big)$$

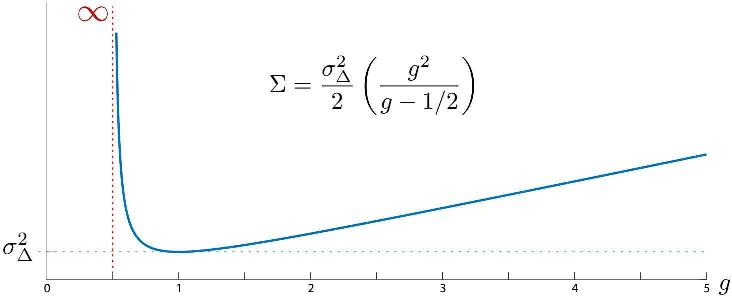
Example: $X(n) = W^2(n)$, $W \sim N(0,1)$

Optimal Variance

Normalization for analysis: $\Delta(n) = X(n) - \mathsf{E}[X(n)]$

$$\tilde\theta(n+1) = \tilde\theta(n) + \frac{g}{n+1}\Big(-\tilde\theta(n) + \Delta(n+1)\Big)$$

Example: $X(n) = W^2(n)$, $W \sim N(0,1)$



Asymptotic variance as a function of g

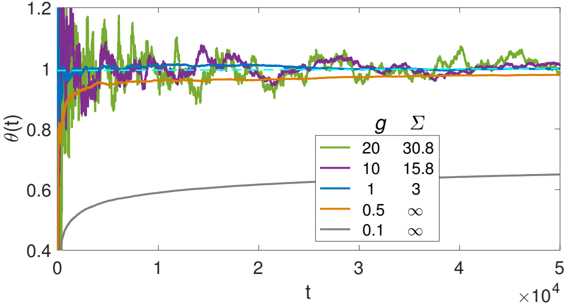
Optimal Variance

Normalization for analysis:

$$\Delta(n) = X(n) - \mathsf{E}[X(n)]$$

$$\tilde{\theta}(n+1) = \tilde{\theta}(n) + \frac{g}{n+1}\Big(-\tilde{\theta}(n) + \Delta(n+1)\Big)$$

Example: $X(n) = W^2(n)$, $W \sim N(0,1)$



SA estimates of $\mathsf{E}[W^2]$, $W \sim N(0,1)$

Optimal Asymptotic Covariance and Zap-SNR

Zap-SNR (designed to emulate deterministic Newton-Raphson)

$$\text{Requires} \quad \widehat{A}_n \approx A(\theta_n) := \frac{d}{d\theta} \bar{f}\,(\theta_n)$$

Optimal Asymptotic Covariance and Zap-SNR

Zap-SNR (designed to emulate deterministic Newton-Raphson)

$$\theta(n+1) = \theta(n) + \alpha_n (-\widehat{A}_n)^{-1} f(\theta(n), X(n))$$

$$\widehat{A}_n = \widehat{A}_{n-1} + \gamma_n (A_n - \widehat{A}_{n-1}), \qquad A_n = \frac{d}{d\theta} f(\theta(n), X(n))$$

Optimal Asymptotic Covariance and Zap-SNR

Zap-SNR (designed to emulate deterministic Newton-Raphson)

$$\theta(n+1) = \theta(n) + \alpha_n (-\widehat{A}_n)^{-1} f(\theta(n), X(n))$$

$$\widehat{A}_n = \widehat{A}_{n-1} + \gamma_n (A_n - \widehat{A}_{n-1}), \qquad A_n = \frac{d}{d\theta} f(\theta(n), X(n))$$

$$\widehat{A}_n \approx A(\theta_n) \;\; \text{requires high-gain, } \frac{\gamma_n}{\alpha_n} \to \infty, \qquad n \to \infty$$

Optimal Asymptotic Covariance and Zap-SNR

Zap-SNR (designed to emulate deterministic Newton-Raphson)

$$\theta(n+1) = \theta(n) + \alpha_n(-\widehat{A}_n)^{-1} f(\theta(n), X(n))$$

$$\widehat{A}_n = \widehat{A}_{n-1} + \gamma_n(A_n - \widehat{A}_{n-1}), \qquad A_n = \frac{d}{d\theta} f(\theta(n), X(n))$$

$$\widehat{A}_n \approx A(\theta_n) \ \text{ requires high-gain, } \frac{\gamma_n}{\alpha_n} \to \infty, \qquad n \to \infty$$

Always: $\alpha_n = 1/n$. Numerics that follow: $\gamma_n = (1/n)^\rho$, $\rho \in (0.5, 1)$

Optimal Asymptotic Covariance and Zap-SNR

Zap-SNR (designed to emulate deterministic Newton-Raphson)

$$\theta(n+1) = \theta(n) + \alpha_n(-\widehat{A}_n)^{-1} f(\theta(n), X(n))$$

$$\widehat{A}_n = \widehat{A}_{n-1} + \gamma_n(A_n - \widehat{A}_{n-1}), \qquad A_n = \frac{d}{d\theta} f(\theta(n), X(n))$$

$$\widehat{A}_n \approx A(\theta_n) \ \text{ requires high-gain, } \frac{\gamma_n}{\alpha_n} \to \infty, \qquad n \to \infty$$

Always: $\alpha_n = 1/n$. Numerics that follow: $\gamma_n = (1/n)^\rho$, $\rho \in (0.5, 1)$

ODE for Zap-SNR

$$\frac{d}{dt} x_t = -\big[A(x_t)\big]^{-1} \bar{f}(x_t), \qquad A(x) = \frac{d}{dx} \bar{f}(x)$$

Optimal Asymptotic Covariance and Zap-SNR

Zap-SNR (designed to emulate deterministic Newton-Raphson)

$$\theta(n+1) = \theta(n) + \alpha_n(-\widehat{A}_n)^{-1} f(\theta(n), X(n))$$

$$\widehat{A}_n = \widehat{A}_{n-1} + \gamma_n(A_n - \widehat{A}_{n-1}), \qquad A_n = \frac{d}{d\theta} f(\theta(n), X(n))$$

$$\widehat{A}_n \approx A(\theta_n) \text{ requires high-gain, } \frac{\gamma_n}{\alpha_n} \to \infty, \qquad n \to \infty$$

Always: $\alpha_n = 1/n$. Numerics that follow: $\gamma_n = (1/n)^\rho$, $\rho \in (0.5, 1)$

ODE for Zap-SNR

$$\frac{d}{dt} x_t = -\big[A(x_t)\big]^{-1} \bar{f}(x_t), \qquad A(x) = \frac{d}{dx} \bar{f}(x)$$

- Not necessarily stable (just like in deterministic Newton-Raphson)
- *General* conditions for convergence open

Detailed Comments:	1. In general, it is not clear how techniques from stochastic approximation can add to the literature of reinforcement learning. SA is concerned with stability and asymptotics, while RL is concerned with the efficiency of learning (sample complexity and regret). In general,I cannot convince myself why ppl care about the stability/asymptotic of SA in the context of RL. I think more justification is needed to bring together the theory of SA and RL.

Reinforcement Learning and Stochastic Approximation

condition Q1 in Theorem 1: (X,U) is an irreducible Markov chain. This assumption excludes the possibility of policy exploration and policy adaptation, which is key to RL. Under this theoretical limitation, the proposed method and analysis does not apply to general RL, making the stability/asymptotic results less interesting.

3. One contribution claimed in the paper is variance reduction. It seems that no theoretical justification is provided about how much is the variance reduced? Is it related to any condition

Reinforcement Learning and Stochastic Approximation

SA and RL Design

Functional equations in Stochastic Control

Always of the form

$$0 = \mathsf{E}[F(h^*, \Phi(n+1)) \mid \Phi_0 \ldots \Phi(n)]\,, \qquad h^* = ?$$

SA and RL Design

Functional equations in Stochastic Control

Always of the form

$$0 = \mathsf{E}[F(h^*, \Phi(n+1)) \mid \Phi_0 \ldots \Phi(n)] \,, \qquad h^* = ?$$

$$\Phi(n) = (\text{state, action})$$

SA and RL Design

Functional equations in Stochastic Control

Always of the form

$0 = \mathsf{E}[F(h^*, \Phi(n+1)) \mid \Phi_0 \ldots \Phi(n)]\,, \qquad h^* = ?$

Galerkin relaxation:

$0 = \mathsf{E}[F(h^{\theta^*}, \Phi(n+1))\zeta_n]\,, \qquad \theta^* = ?$

SA and RL Design

Functional equations in Stochastic Control

Always of the form

$$0 = \mathsf{E}[F(h^*, \Phi(n+1)) \mid \Phi_0 \dots \Phi(n)]\,, \qquad h^* = ?$$

Galerkin relaxation:

$$0 = \mathsf{E}[F(h^{\theta^*}, \Phi(n+1))\zeta_n]\,, \qquad \theta^* = ?$$

Necessary Ingredients:

- Parameterized family $\{h^\theta : \theta \in \mathbb{R}^d\}$
- Adapted, d-dimensional stochastic process $\{\zeta_n\}$

Examples are TD- and Q-Learning

SA and RL Design

Functional equations in Stochastic Control

Always of the form

$$0 = \mathsf{E}[F(h^*, \Phi(n+1)) \mid \Phi_0 \dots \Phi(n)] , \qquad h^* = ?$$

Galerkin relaxation:

$$0 = \mathsf{E}[F(h^{\theta^*}, \Phi(n+1))\zeta_n] , \qquad \theta^* = ?$$

Necessary Ingredients:

- Parameterized family $\{h^\theta : \theta \in \mathbb{R}^d\}$
- Adapted, d-dimensional stochastic process $\{\zeta_n\}$

Examples are TD- and Q-Learning

These algorithms are thus special cases of stochastic approximation

(as we all know)

Stochastic Optimal Control

MDP Model

$\boldsymbol{X}$ is a stationary controlled Markov chain, with input $\boldsymbol{U}$

- For all states x and sets A,

  $\mathsf{P}\{X(n+1) \in A \mid X(n) = x,\ U(n) = u, \text{and prior history}\} = P_u(x, A)$
- $c\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$ is a cost function
- $\beta < 1$ a discount factor

Stochastic Optimal Control

MDP Model

$\boldsymbol{X}$ is a stationary controlled Markov chain, with input $\boldsymbol{U}$

- For all states x and sets A,

  $\mathsf{P}\{X(n+1) \in A \mid X(n) = x,\ U(n) = u, \text{and prior history}\} = P_u(x, A)$
- $c\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$ is a cost function
- $\beta < 1$ a discount factor

Value function:

$$h^*(x) = \min_{\boldsymbol{U}} \sum_{n=0}^{\infty} \beta^n \mathsf{E}[c(X(n), U(n)) \mid X(0) = x]$$

Stochastic Optimal Control

MDP Model

$\boldsymbol{X}$ is a stationary controlled Markov chain, with input $\boldsymbol{U}$

- For all states x and sets A,

  $\mathsf{P}\{X(n+1) \in A \mid X(n) = x,\ U(n) = u, \text{and prior history}\} = P_u(x, A)$
- $c\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$ is a cost function
- $\beta < 1$ a discount factor

Value function:

$$h^*(x) = \min_{\boldsymbol{U}} \sum_{n=0}^{\infty} \beta^n \mathsf{E}[c(X(n), U(n)) \mid X(0) = x]$$

Bellman equation:

$$h^*(x) = \min_u \{c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u]\}$$

Q-function

Bellman equation:

$$h^*(x) = \min_u \{c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u]\}$$

Q-function

Bellman equation:

$$h^*(x) = \min_u \{c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u]\}$$

Q-function:

$$Q^*(x,u) := c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u]$$

Q-function

Bellman equation:

$$h^*(x) = \min_u \{ c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u] \}$$

Q-function:

$$Q^*(x,u) := c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u]$$

$$h^*(x) = \min_u Q^*(x,u)$$

Q-function

Bellman equation:

$$h^*(x) = \min_u \{c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u]\}$$

Q-function:

$$Q^*(x,u) := c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u]$$

$$h^*(x) = \min_u Q^*(x,u)$$

Another Bellman equation:

$$Q^*(x,u) = c(x,u) + \beta \mathsf{E}[\underline{Q}^*(X(n+1)) \mid X(n) = x,\ U(n) = u]$$
$$\underline{Q}^*(x) = \min_u Q^*(x,u)$$

Q-function

Bellman equation:

$$h^*(x) = \min_u \{c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u]\}$$

Q-function:

$$Q^*(x,u) := c(x,u) + \beta \mathsf{E}[h^*(X(n+1)) \mid X(n) = x,\ U(n) = u]$$

$$h^*(x) = \min_u Q^*(x,u)$$

Another Bellman equation:

$$Q^*(x,u) = c(x,u) + \beta \mathsf{E}[\underline{Q}^*(X(n+1)) \mid X(n) = x,\ U(n) = u]$$
$$\underline{Q}^*(x) = \min_u Q^*(x,u)$$

Q-function: trick to swap expectation and minimum

Q-Learning and Galerkin Relaxation

Dynamic programming

Find function Q^* that solves

$$\mathsf{E}\Big[c(X(n),U(n)) + \beta \underline{Q}^*(X(n+1)) - Q^*(X(n),U(n)) \mid \mathcal{F}_n\Big] = 0$$

Q-Learning and Galerkin Relaxation

Dynamic programming

Find function Q^* that solves

$$\mathsf{E}\big[c(X(n),U(n)) + \beta \underline{Q}^*(X(n+1)) - Q^*(X(n),U(n)) \mid \mathcal{F}_n\big] = 0$$

That is,

$$0 = \mathsf{E}[F(Q^*, \Phi(n+1)) \mid \Phi_0 \ldots \Phi(n)]\,,$$

$$\text{with } \Phi(n+1) = (X(n+1), X(n), U(n)).$$

Q-Learning and Galerkin Relaxation

Dynamic programming

Find function Q^* that solves

$$\mathsf{E}\Big[c(X(n),U(n)) + \beta \underline{Q}^*(X(n+1)) - Q^*(X(n),U(n)) \mid \mathcal{F}_n\Big] = 0$$

Q-Learning

Find θ^* that solves

$$\mathsf{E}\Big[\Big(c(X(n),U(n)) + \beta \underline{Q}^{\theta^*}((X(n+1)) - Q^{\theta^*}((X(n),U(n))\Big)\zeta_n\Big] = 0$$

The family $\{Q^\theta\}$ and *eligibility vectors* $\{\zeta_n\}$ are part of algorithm design.

Watkins' Q-learning

Find θ^* that solves

$$\mathsf{E}\Big[\Big(c(X(n),U(n)) + \beta \underline{Q}^{\theta^*}((X(n+1)) - Q^{\theta^*}((X(n),U(n))\Big)\zeta_n\Big] = 0$$

Watkins' Q-learning

Find θ^* that solves

$$\mathsf{E}\Big[\Big(c(X(n),U(n)) + \beta \underline{Q}^{\theta^*}((X(n+1)) - Q^{\theta^*}((X(n),U(n))\Big)\zeta_n\Big] = 0$$

Watkin's algorithm is Stochastic Approximation

The family $\{Q^\theta\}$ and *eligibility vectors* $\{\zeta_n\}$ in this design:

- Linearly parameterized family of functions: $Q^\theta(x,u) = \theta^{\intercal}\psi(x,u)$
- $\zeta_n \equiv \psi(X_n, U_n)$
- $\psi_i(x,u) = 1\{x = x^i, u = u^i\}$ (complete basis)

Watkins' Q-learning

Find θ^* that solves

$$\mathsf{E}\Big[\Big(c(X(n),U(n)) + \beta \underline{Q}^{\theta^*}((X(n+1)) - Q^{\theta^*}((X(n),U(n))\Big)\zeta_n\Big] = 0$$

Watkin's algorithm is Stochastic Approximation

The family $\{Q^\theta\}$ and *eligibility vectors* $\{\zeta_n\}$ in this design:

- Linearly parameterized family of functions: $Q^\theta(x,u) = \theta^\intercal \psi(x,u)$
- $\zeta_n \equiv \psi(X_n, U_n)$
- $\psi_i(x,u) = 1\{x = x^i, u = u^i\}$ (complete basis)

Asymptotic covariance is infinite for $\beta \geq 1/2$ [NIPS 2017]

Watkins' Q-learning

Big Question: *Can we Zap Q-Learning?*

Find θ^* that solves

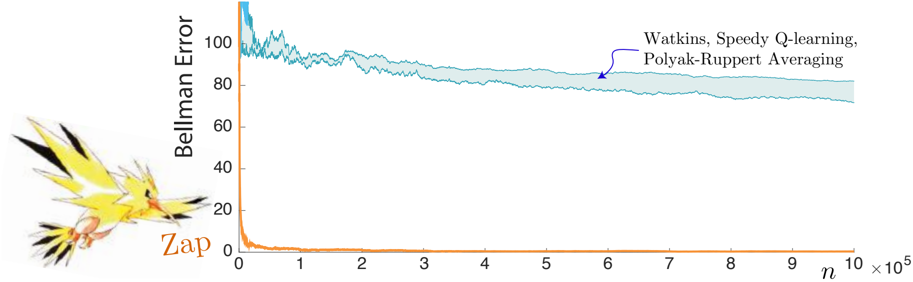
$$\mathsf{E}\Big[\Big(c(X(n),U(n)) + \beta \underline{Q}^{\theta^*}((X(n+1)) - Q^{\theta^*}((X(n),U(n))\Big)\zeta_n\Big] = 0$$

Watkin's algorithm is Stochastic Approximation

The family $\{Q^\theta\}$ and *eligibility vectors* $\{\zeta_n\}$ in this design:

- Linearly parameterized family of functions: $Q^\theta(x,u) = \theta^\intercal \psi(x,u)$
- $\zeta_n \equiv \psi(X_n, U_n)$
- $\psi_i(x,u) = 1\{x = x^i, u = u^i\}$ (complete basis)

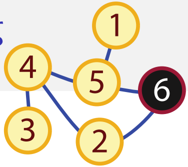
Asymptotic covariance is infinite for $\beta \geq 1/2$ [NIPS 2017]



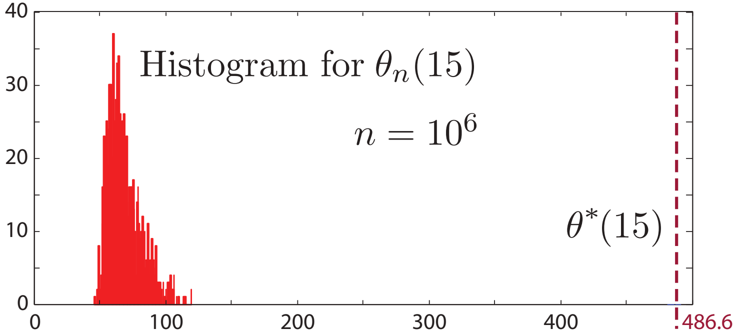
Zap Q-Learning

Asymptotic Covariance of Watkins' Q-Learning

Improvements are needed!



Histogram of parameter estimates after 10^6 iterations.



Example from Devraj & M 2017

Zap Q-learning

Zap Q-Learning $\equiv$ Zap-SNR for Q-Learning

$$0 = \bar{f}(\theta) = \mathsf{E}\big[f(\theta, W(n))\big]$$
$$:= \mathsf{E}\Big[\zeta_n\Big(c(X(n), U(n)) + \beta \underline{Q}^\theta(X(n+1)) - Q^\theta(X(n), U(n))\Big)\Big]$$

$A(\theta) = \frac{d}{d\theta}\bar{f}(\theta)$; At points of differentiability:

$$A(\theta) = \mathsf{E}\Big[\zeta_n\big[\beta\psi(X(n+1), \phi^\theta(X(n+1))) - \psi(X(n), U(n))\big]^T\Big]$$
$$\phi^\theta(X(n+1)) := \arg\min_u Q^\theta(X(n+1), u)$$

Zap Q-learning

Zap Q-Learning $\equiv$ Zap-SNR for Q-Learning

Algorithm:

$$\theta(n+1) = \theta(n) + \alpha_n (-\widehat{A}_n)^{-1} f(\theta(n), \Phi(n)) \,, \qquad \widehat{A}_n = \widehat{A}_{n-1} + \gamma_n (A_n - \widehat{A}_{n-1})$$

$$\begin{aligned} A_{n+1} &:= \frac{d}{d\theta} f\left(\theta_n, \Phi(n)\right) \\ &= \zeta_n \big[\beta\psi(X(n+1), \phi^{\theta_n}(X(n+1))) - \psi(X(n), U(n))\big]^{T} \end{aligned}$$

Zap Q-learning

Zap Q-Learning $\equiv$ Zap-SNR for Q-Learning

ODE Analysis: change of variables $q = \mathcal{Q}^*(\varsigma)$

Functional $\mathcal{Q}^*$ maps cost functions to Q-functions:

$$q(x,u) = \varsigma(x,u) + \beta \sum_{x'} P_u(x,x') \min_{u'} q(x',u')$$

Zap Q-learning

Zap Q-Learning ≡ Zap-SNR for Q-Learning

ODE Analysis: change of variables $q = \mathcal{Q}^*(\varsigma)$

Functional $\mathcal{Q}^*$ maps cost functions to Q-functions:

$$q(x,u) = \varsigma(x,u) + \beta \sum_{x'} P_u(x,x') \min_{u'} q(x',u')$$

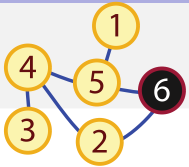
ODE for Zap-Q

$$q_t = \mathcal{Q}^*(\varsigma_t), \qquad \frac{d}{dt}\varsigma_t = -\varsigma_t + c$$

$\Rightarrow$ convergence, optimal covariance, ...

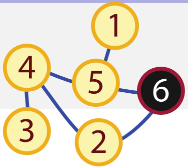
Zap Q-Learning

Example: Stochastic Shortest Path



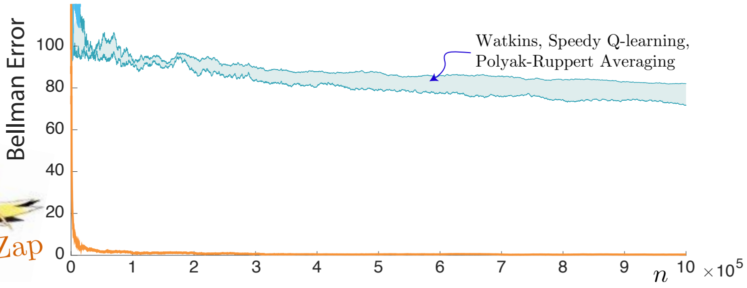
Zap Q-Learning

Example: Stochastic Shortest Path



Convergence with Zap gain $\gamma_n = n^{-0.85}$

Watkins' algorithm has infinite asymptotic covariance with $\alpha_n = 1/n$

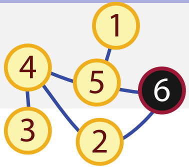


Convergence of Zap-Q Learning

Discount factor: $\beta = 0.99$

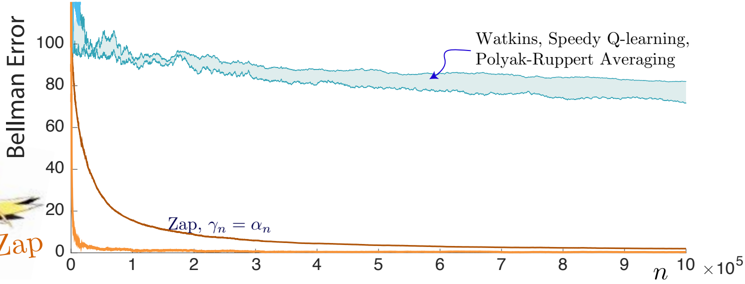
Zap Q-Learning

Example: Stochastic Shortest Path



Convergence with Zap gain $\gamma_n = n^{-0.85}$

Watkins' algorithm has infinite asymptotic covariance with $\alpha_n = 1/n$

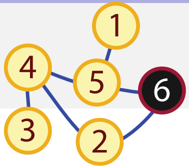


Convergence of Zap-Q Learning

Discount factor: $\beta = 0.99$

Zap Q-Learning

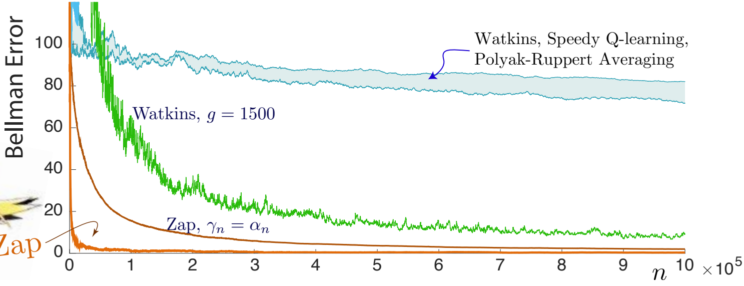
Example: Stochastic Shortest Path



Convergence with Zap gain $\gamma_n = n^{-0.85}$

Watkins' algorithm has infinite asymptotic covariance with $\alpha_n = 1/n$

Optimal scalar gain is approximately $\alpha_n = 1500/n$

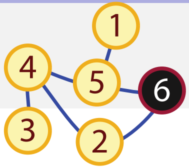


Convergence of Zap-Q Learning

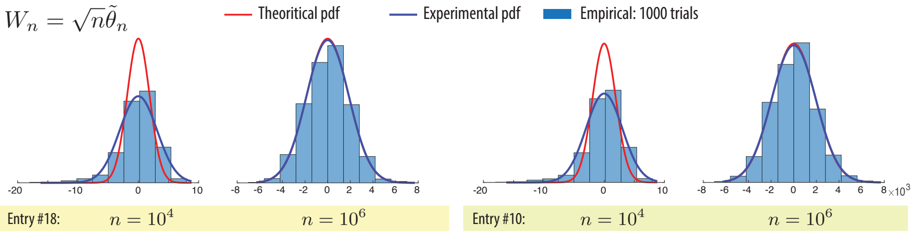
Discount factor: $\beta = 0.99$

Zap Q-Learning

Optimize Walk to Cafe



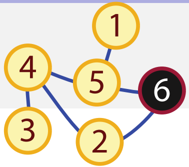
Convergence with Zap gain $\gamma_n = n^{-0.85}$



CLT gives good prediction of finite-n performance

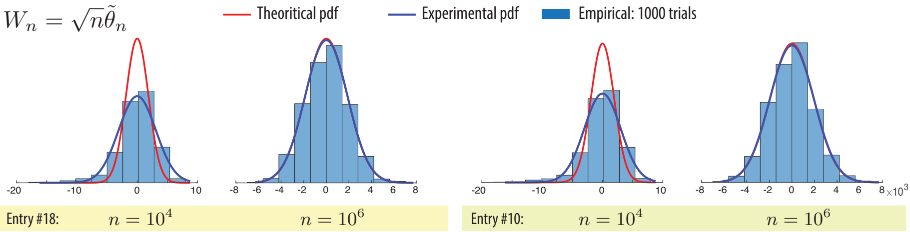
Zap Q-Learning

Optimize Walk to Cafe



Convergence with Zap gain $\gamma_n = n^{-0.85}$

$W_n = \sqrt{n}\tilde{\theta}_n$



CLT gives good prediction of finite-n performance

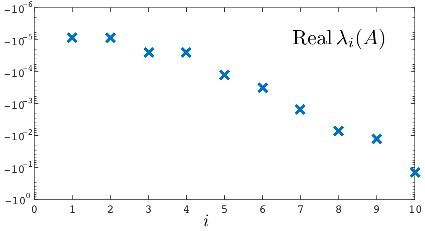
Discount factor: $\beta = 0.99$

Zap Q-Learning

Model of Tsitsiklis and Van Roy: **Optimal Stopping Time in Finance**

State space: $\mathbb{R}^{100}$

Parameterized Q-function: Q^θ with $\theta \in \mathbb{R}^{10}$



$\mathrm{Real}\, \lambda > -\frac{1}{2}$ for every eigenvalue λ

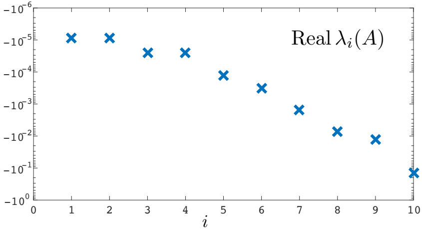
Asymptotic covariance is infinite

Zap Q-Learning

Model of Tsitsiklis and Van Roy: **Optimal Stopping Time in Finance**

State space: $\mathbb{R}^{100}$

Parameterized Q-function: Q^{θ} with $\theta \in \mathbb{R}^{10}$



Real $\lambda > -\frac{1}{2}$ for every eigenvalue λ

Asymptotic covariance is infinite

Authors observed slow convergence
Proposed a matrix gain sequence

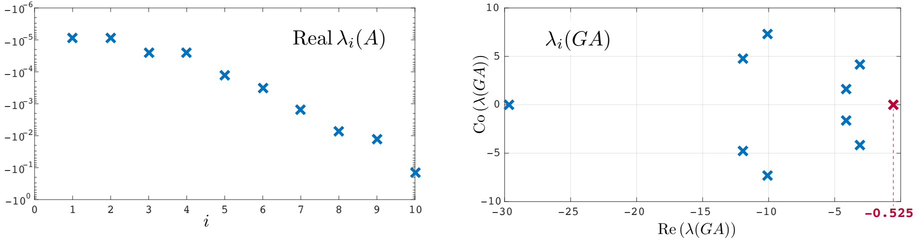
$\{G_n\}$ (see refs for details)

Zap Q-Learning

Model of Tsitsiklis and Van Roy: **Optimal Stopping Time in Finance**

State space: $\mathbb{R}^{100}$

Parameterized Q-function: Q^θ with $\theta \in \mathbb{R}^{10}$



Eigenvalues of A and GA for the finance example

Favorite choice of gain in [25] barely meets the criterion $\mathrm{Re}(\lambda(GA)) < -\frac{1}{2}$

Zap Q-Learning

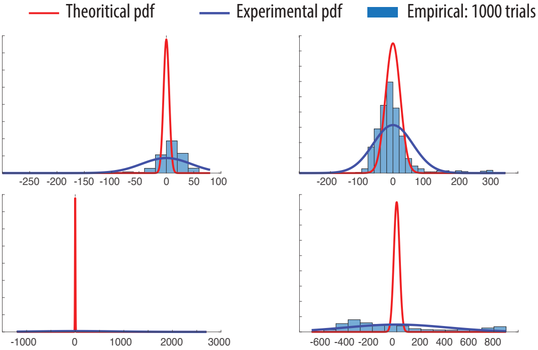
Model of Tsitsiklis and Van Roy: **Optimal Stopping Time in Finance**

State space: $\mathbb{R}^{100}$. Parameterized Q-function: Q^θ with $\theta \in \mathbb{R}^{10}$

$$W_n = \sqrt{n}\tilde{\theta}_n$$

Zap-Q

G-Q



Entry #1: $n = 2 \times 10^6$

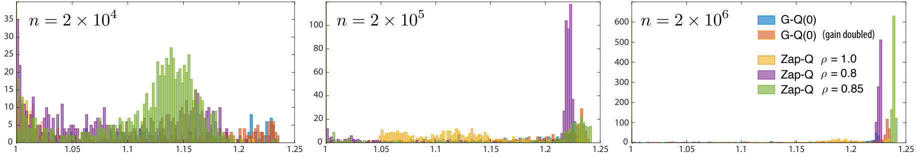
Entry #7: $n = 2 \times 10^6$

Zap Q-Learning

Model of Tsitsiklis and Van Roy: **Optimal Stopping Time in Finance**

State space: $\mathbb{R}^{100}$.
Parameterized Q-function: Q^θ with $\theta \in \mathbb{R}^{10}$

Histograms of the average reward obtained using the different algorithms:



Zap-Q $\gg$ G-Q

Conclusions & Future Work

Conclusions

- Reinforcement Learning is not just cursed by dimension, but also by variance

  We need better design tools to improve performance

Conclusions & Future Work

Conclusions

- Reinforcement Learning is not just cursed by dimension, but also by variance

  We need better design tools to improve performance

- The asymptotic covariance is an awesome design tool. It is also predictive of finite-n performance.

  Example: $g^* = 1500$ was chosen based on **asymptotic** covariance

Conclusions & Future Work

Conclusions

- Reinforcement Learning is not just cursed by dimension, but also by variance

  We need better design tools to improve performance

- The asymptotic covariance is an awesome design tool. It is also predictive of finite-n performance.

  Example: $g^* = 1500$ was chosen based on **asymptotic** covariance

- Future work:
  - Q-learning with function-approximation
    - *Obtain conditions for a stable algorithm in a general setting*
    - *Optimal stopping time problems*
- Adaptive optimization of algorithm parameters

Conclusions & Future Work

Conclusions

- Reinforcement Learning is not just cursed by dimension, but also by variance

  We need better design tools to improve performance

- The asymptotic covariance is an awesome design tool. It is also predictive of finite-n performance.

  Example: $g^* = 1500$ was chosen based on **asymptotic** covariance

- Future work:
  - Q-learning with function-approximation
    - *Obtain conditions for a stable algorithm in a general setting*
    - *Optimal stopping time problems*

- Adaptive optimization of algorithm parameters
- Zapped Momentum Methods [2]

Conclusions & Future Work

Opportunities for the controls community

Conclusions & Future Work

Opportunities for the controls community

We Are Q!

Conclusions & Future Work

Opportunities for the controls community

We Are Q!

- Apply your favorite model-reduction technique, or class of policies, or family of value functions, and create your own RL algorithm

Conclusions & Future Work

Opportunities for the controls community

We Are Q!

- Apply your favorite model-reduction technique,
  or class of policies, or family of value functions,
  and create your own RL algorithm
- Unshackle conventions of the RL research community.
  This lecture is *reinforcement learning* $\equiv$ on online optimization.

Conclusions & Future Work

Opportunities for the controls community

We Are Q!

- Apply your favorite model-reduction technique,
  or class of policies, or family of value functions,
  and create your own RL algorithm
- Unshackle conventions of the RL research community.
  This lecture is *reinforcement learning* $\equiv$ on online optimization.
  If you have a model, you have many options.

Conclusions & Future Work

Opportunities for the controls community

We Are Q!

- Apply your favorite model-reduction technique, or class of policies, or family of value functions, and create your own RL algorithm
- Unshackle conventions of the RL research community. This lecture is *reinforcement learning* $\equiv$ on online optimization. If you have a model, you have many options.
- The most exciting applications may be for your favorite model:

$$\frac{d}{dt}x_t = f(x_t, u_t)\,, \quad J^*(x) = \min_{\boldsymbol{u}} \int_0^\infty c(x_t, u_t)\,dt\,,\ x_0 = x$$

Conclusions & Future Work

Opportunities for the controls community

We Are Q!

- Apply your favorite model-reduction technique,
  or class of policies, or family of value functions,
  and create your own RL algorithm
- Unshackle conventions of the RL research community.
  This lecture is *reinforcement learning* $\equiv$ on online optimization.
  If you have a model, you have many options.
- The most exciting applications may be for your favorite model:

$$\frac{d}{dt}x_t = f(x_t, u_t)\,, \quad J^*(x) = \min_{\boldsymbol{u}} \int_0^\infty c(x_t, u_t)\, dt\,,\ \ x_0 = x$$

Forget about Markov chains and randomized policies!

Follow your heart

Conclusions & Future Work

Opportunities for the controls community

We Are Q!

- Apply your favorite model-reduction technique, or class of policies, or family of value functions, and create your own RL algorithm
- Unshackle conventions of the RL research community. This lecture is *reinforcement learning* $\equiv$ on online optimization. If you have a model, you have many options.
- The most exciting applications may be for your favorite model:

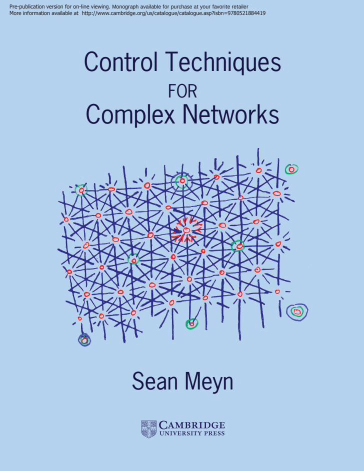
$$\frac{d}{dt}x_t = f(x_t, u_t)\,, \quad J^*(x) = \min_{\boldsymbol{u}} \int_0^\infty c(x_t, u_t)\, dt\,,\ x_0 = x$$

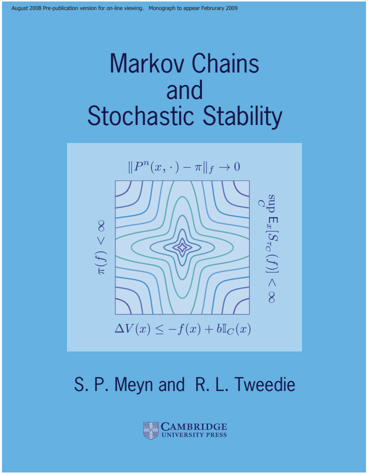
Forget about Markov chains and randomized policies!

Follow your heart

Åström, Cassandras, Jovanovic, Jain, Kristic, friends@NREL ... *we are Q!*

Thank you!


Pre-publication version for on-line viewing. Monograph available for purchase at your favorite retailer
More information available at http://www.cambridge.org/us/catalogue/catalogue.asp?isbn=9780521884419
Control Techniques
FOR
Complex Networks
Sean Meyn
CAMBRIDGE
UNIVERSITY PRESS


August 2008 Pre-publication version for on-line viewing. Monograph to appear Februrary 2009
Markov Chains
and
Stochastic Stability
$\|P^n(x,\cdot)-\pi\|_f \to 0$
$\pi(f) < \infty$
$\sup_C \mathsf{E}_x[S_{\tau_C}(f)] < \infty$
$\Delta V(x) \le -f(x) + b\mathbb{I}_C(x)$
S. P. Meyn and R. L. Tweedie
CAMBRIDGE
UNIVERSITY PRESS

References

This lecture

- A. M. Devraj and S. P. Meyn, *Zap Q-learning.* *Advances in Neural Information Processing Systems (NIPS).* Dec. 2017.
- A. M. Devraj and S. P. Meyn, *Fastest convergence for Q-learning.* Available on *ArXiv*. Jul. 2017.

Berkeley short course, March 2018

- Part I (Basics, with focus on variance of algorithms)

  https://www.youtube.com/watch?v=dhEF5pfYmvc
- Part II (Zap Q-learning)

  https://www.youtube.com/watch?v=Y3w8f1xIb6s

Selected References I

[1] A. M. Devraj and S. P. Meyn. *Fastest convergence for Q-learning.* *ArXiv* , July 2017 (extended version of NIPS 2017).

[2] A. M. Devraj, A. Bušić, and S. Meyn. Zap Q Learning – a user's guide. In *Proceedings of the fifth Indian Control Conference*, 9-11 January, 2019 2019.

[3] A. Benveniste, M. Métivier, and P. Priouret. *Adaptive algorithms and stochastic approximations*, volume 22 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1990. Translated from the French by Stephen S. Wilson.

[4] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Hindustan Book Agency and Cambridge University Press, Delhi, India and Cambridge, UK, 2008.

[5] V. S. Borkar and S. P. Meyn. *The ODE method for convergence of stochastic approximation and reinforcement learning.* *SIAM J. Control Optim.*, 38(2):447–469, 2000.

[6] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. Cambridge University Press, Cambridge, second edition, 2009. Published in the Cambridge Mathematical Library.

[7] S. P. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, 2007. *See last chapter on simulation and average-cost TD learning*

Selected References II

[8] D. Huang, W. Chen, P. Mehta, S. Meyn, and A. Surana. *Feature selection for neuro-dynamic programming.* In F. Lewis, editor, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley, 2011.

[9] P. G. Mehta and S. P. Meyn. *Q-learning and Pontryagin's minimum principle.* In *IEEE Conference on Decision and Control*, pages 3598–3605, Dec. 2009.

[10] D. Ruppert. *A Newton-Raphson version of the multivariate Robbins-Monro procedure. The Annals of Statistics*, 13(1):236–245, 1985.

[11] D. Ruppert. *Efficient estimators from a slowly convergent Robbins-Monro processes.* Technical Report Tech. Rept. No. 781, Cornell University, School of Operations Research and Industrial Engineering, Ithaca, NY, 1988.

[12] B. T. Polyak. *A new method of stochastic approximation type. Avtomatika i telemekhanika (in Russian). translated in Automat. Remote Control, 51 (1991)*, pages 98–107, 1990.

[13] B. T. Polyak and A. B. Juditsky. *Acceleration of stochastic approximation by averaging. SIAM J. Control Optim.*, 30(4):838–855, 1992.

[14] V. R. Konda and J. N. Tsitsiklis. *Convergence rate of linear two-time-scale stochastic approximation. Ann. Appl. Probab.*, 14(2):796–819, 2004.

Selected References III

[15] E. Moulines and F. R. Bach. *Non-asymptotic analysis of stochastic approximation algorithms for machine learning.* In *Advances in Neural Information Processing Systems 24*, pages 451–459. Curran Associates, Inc., 2011.

[16] C. Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

[17] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, Cambridge, UK, 1989.

[18] C. J. C. H. Watkins and P. Dayan. *Q-learning. Machine Learning*, 8(3-4):279–292, 1992.

[19] R. S. Sutton.*Learning to predict by the methods of temporal differences. Mach. Learn.*, 3(1):9–44, 1988.

[20] J. N. Tsitsiklis and B. Van Roy. *An analysis of temporal-difference learning with function approximation. IEEE Trans. Automat. Control*, 42(5):674–690, 1997.

[21] C. Szepesvári. *The asymptotic convergence-rate of Q-learning.* In *Proceedings of the 10th Internat. Conf. on Neural Info. Proc. Systems*, pages 1064–1070. MIT Press, 1997.

[22] M. G. Azar, R. Munos, M. Ghavamzadeh, and H. Kappen. *Speedy Q-learning.* In *Advances in Neural Information Processing Systems*, 2011.

Selected References IV

[23] E. Even-Dar and Y. Mansour. *Learning rates for Q-learning. Journal of Machine Learning Research*, 5(Dec):1–25, 2003.

[24] J. N. Tsitsiklis and B. Van Roy. *Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. IEEE Trans. Automat. Control*, 44(10):1840–1851, 1999.

[25] D. Choi and B. Van Roy. *A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning. Discrete Event Dynamic Systems: Theory and Applications*, 16(2):207–239, 2006.

[26] S. J. Bradtke and A. G. Barto. *Linear least-squares algorithms for temporal difference learning. Mach. Learn.*, 22(1-3):33–57, 1996.

[27] J. A. Boyan. *Technical update: Least-squares temporal difference learning. Mach. Learn.*, 49(2-3):233–246, 2002.

[28] A. Nedic and D. Bertsekas. *Least squares policy evaluation algorithms with linear function approximation. Discrete Event Dyn. Systems: Theory and Appl.*, 13(1-2):79–110, 2003.