CS&RL In Three Parts

Part 1 Part 2 Part 3

Part 1: Algorithm Design: *Dynamical Systems Approach*

Sean Meyn

COGNITION & CONTROL IN COMPLEX SYSTEMS

Department of Electrical and Computer Engineering University of Florida

Inria International Chair Inria, Paris

Thanks to to our sponsors: NSF and ARO

CS&RL in Three Parts

1. Crash course on stochastic approximation (the ODE method): A dynamical systems approach to algorithm design, addressing transients and variance in learning.
2. MDP theory and a menu of reinforcement learning techniques
3. Actor-Critic methods—an approach to stochastic gradient descent

CS&RL in Three Parts

1. Crash course on stochastic approximation (the ODE method): A dynamical systems approach to algorithm design, addressing transients and variance in learning.
   Based on CS&RL Ch. 8 and [77, 79]. This is all in a probabilistic setting
   —see CS&RL Ch. 4 for a treatment *without probability theory*
2. MDP theory and a menu of reinforcement learning techniques
   Based on CS&RL Ch. 7–10, CTCN Ch. 11, and more recent work: *The projected Bellman equation in RL* [9]
3. Actor-Critic methods—an approach to stochastic gradient descent
   Based on CS&RL Ch. 10, following the MIT school

CS&RL in Three Parts

1. Crash course on stochastic approximation (the ODE method): A dynamical systems approach to algorithm design, addressing transients and variance in learning.
   Based on CS&RL Ch. 8 and [77, 79]. This is all in a probabilistic setting
   —see CS&RL Ch. 4 for a treatment *without probability theory*
2. MDP theory and a menu of reinforcement learning techniques
   Based on CS&RL Ch. 7–10, CTCN Ch. 11, and more recent work: *The projected Bellman equation in RL* [9]
3. Actor-Critic methods—an approach to stochastic gradient descent
   Based on CS&RL Ch. 10, following the MIT school

Why all the theory?

- You will learn many reasons for *failure* in RL algorithms
- A rich tool kit for algorithm design will emerge

CS&RL in Three Parts

1. Crash course on stochastic approximation (the ODE method): A dynamical systems approach to algorithm design, addressing transients and variance in learning.
2. MDP theory and a menu of reinforcement learning techniques
3. Actor-Critic methods—an approach to stochastic gradient descent

Why all the theory?

- You will learn many reasons for *failure* in RL algorithms
- A rich tool kit for algorithm design will emerge

What's left out?

- The wide world of exploration
- Focusing only on *theory* of actor-critic methods

Part 1: Algorithm Design

Outline

Skip to Part 2 Skip to Part 3

1. Goals
2. The ODE Method
3. Unveiling Dynamics
4. Introduction to Zap
5. Conclusions
6. Appendix: Understanding Variance

Resources

References

- ODE Method (using different meaning than in the 1970s)
  - CS&RL, Chapters 4 and 8
  - The ODE Method for Asymptotic Statistics in Stochastic Approximation and Reinforcement Learning [77] Most relevant for this lecture
  - Markovian Foundations for Quasi Stochastic Approximation With Applications to Extremum Seeking Control [100] Today's lecture sans probability
- Gradient Free Optimization
  - CS&RL, Chapters 4 and [99].
  - See bibliography for references on extremum seeking control, such as [109, 115], and Spall's approaches to GFO [85].

Special Thanks

My interests in RL&SA began during my first sabbatical—at IISc with Vivek Borkar

Many other heroes along the way since

Ioannis Kontoyiannis

Ana Busic

Eric Moulines

Adithya Devraj

Many Others!

Alert!

Much of this work will be found in the dissertation of Caio Lauand

Introducing Dr. Lauand in May, 2025

VI. Stunning reliability with Q^{θ} parameterized by various neural networks

Goals

MCMC

Goal: We would like to perform an integral $\eta = \int c(x)\, f(x)\, dx$

where f is a density.

MCMC

Goal: We would like to perform an integral $\eta = \int c(x)\, f(x)\, dx$

where f is a density.

Approach: Construct a Markov chain $\mathbf{\Phi}$ whose steady state distribution has density f (perhaps normalized)

MCMC

Goal: We would like to perform an integral $\eta = \int c(x)\, f(x)\, dx$ where f is a density.

Approach: Construct a Markov chain $\mathbf{\Phi}$ whose steady state distribution has density f

Monte-Carlo estimates:

$$\hat{\eta}_n = \frac{1}{n} \sum_{k=1}^{n} c(\Phi_k)$$

MCMC

Goal: We would like to perform an integral $\eta = \int c(x)\, f(x)\, dx$ where f is a density.

Approach: Construct a Markov chain $\mathbf{\Phi}$ whose steady state distribution has density f

Monte-Carlo estimates:

$$\hat{\eta}_n = \frac{1}{n} \sum_{k=1}^{n} c(\Phi_k)$$

Stochastic approximation:

$$\hat{\eta}_{n+1} = \hat{\eta}_n + \alpha_{n+1}[c(\Phi_{n+1}) - \hat{\eta}_n]$$

MCMC

Goal: We would like to perform an integral $\eta = \int c(x)\, f(x)\, dx$

where f is a density.

Approach: Construct a Markov chain $\mathbf{\Phi}$ whose steady state distribution has density f

Monte-Carlo estimates:

$$\hat{\eta}_n = \frac{1}{n} \sum_{k=1}^{n} c(\Phi_k)$$

Stochastic approximation:

$$\hat{\eta}_{n+1} = \hat{\eta}_n + \alpha_{n+1}[c(\Phi_{n+1}) - \hat{\eta}_n]$$

SA = MC for step-size $\alpha_n = 1/n$

Gradient Descent

Minimize objective $\Gamma\colon \mathbb{R}^d$

Rastrigin Objective

Gradient Descent

Minimize objective $\Gamma\colon \mathbb{R}^d$

$\nabla\Gamma\left(\theta^*\right) = 0$ first-order condition for optimality of $\theta^* \in \mathbb{R}^d$.

where ∇ denotes gradient: $[\nabla\Gamma]_i = \frac{\partial}{\partial\theta_i}\Gamma$

Rastrigin Objective

Gradient Descent

Minimize objective $\Gamma\colon \mathbb{R}^d$

Stochastic Gradient Descent:

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\big[-\nabla\Gamma(\theta_n) + W_{n+1}\big]$$

$\nabla\Gamma(\theta^*) = 0$ first-order condition for optimality of $\theta^* \in \mathbb{R}^d$.
where ∇ denotes gradient: $[\nabla\Gamma]_i = \frac{\partial}{\partial\theta_i}\Gamma$

Rastrigin Objective

Gradient Descent

Minimize objective $\Gamma\colon \mathbb{R}^d$

Stochastic Gradient Descent:

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\big[-\nabla\Gamma(\theta_n) + W_{n+1}\big]$$

Zero mean *exploration* included to avoid local minima

$\nabla\Gamma(\theta^*) = 0$ first-order condition for optimality of $\theta^* \in \mathbb{R}^d$.
where ∇ denotes gradient: $[\nabla\Gamma]_i = \frac{\partial}{\partial\theta_i}\Gamma$

Rastrigin Objective

Gradient Descent

Minimize objective $\Gamma\colon \mathbb{R}^d$

Zeroth order methods:
Spall's SPSA algorithm;

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\big[-\widetilde{\nabla}\Gamma(\theta_n) + W_{n+1}\big]$$

Gradient approximation: given a zero mean probing signal $\{\xi_{n+1}\}$,

$$\widetilde{\nabla}\Gamma(\theta_n) = \frac{1}{\varepsilon}\xi_{n+1}\Gamma(\theta_n + \varepsilon\xi_{n+1})$$

$\nabla\Gamma(\theta^*) = 0$ first-order condition for optimality of $\theta^* \in \mathbb{R}^d$.
where ∇ denotes gradient: $[\nabla\Gamma]_i = \frac{\partial}{\partial\theta_i}\Gamma$

Rastrigin Objective

Gradient Descent

Minimize objective $\Gamma\colon \mathbb{R}^d$

Zeroth order methods:
Spall's SPSA algorithm;

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\big[-\widetilde{\nabla}\Gamma(\theta_n) + W_{n+1}\big]$$

Gradient approximation: given a zero mean probing signal $\{\xi_{n+1}\}$,

$$\widetilde{\nabla}\Gamma(\theta_n) = \frac{1}{\varepsilon}\xi_{n+1}\Gamma(\theta_n + \varepsilon\xi_{n+1})$$

Justified through a bit of Taylor series

$\nabla\Gamma(\theta^*) = 0$ first-order condition for optimality of $\theta^* \in \mathbb{R}^d$.
where ∇ denotes gradient: $[\nabla\Gamma]_i = \frac{\partial}{\partial\theta_i}\Gamma$

Rastrigin Objective

Why Gradient Free?

Examples

- Optimizing the wind farms in Colorado

Maximize over θ: $-\Gamma(\theta) =$ power output as function of decision variables θ

Why Gradient Free?

Examples

- Optimizing the wind farms in Colorado
- Actor-Only methods in Reinforcement Learning

$$\widetilde{\nabla}\Gamma(\theta_n) = \nabla\Gamma(\theta_n) + \mathcal{W}_n$$

Many tools for gradient free optimization

Decision rule: $U_n = \phi^\theta(X_n)$

Why Gradient Free?

Examples

- Optimizing the wind farms in Colorado
- Actor-Only methods in Reinforcement Learning
- Actor-Critic methods

$$\widetilde{\nabla}\Gamma(\theta_n) = \nabla\Gamma(\theta_n) + \mathcal{W}_n$$

mean zero *weak sense*

Thanks to TD(1) $\Longrightarrow$ *score function*

Decision rule: $U_n = \phi^\theta(X_n)$

Root Finding — *Not Just About Gradients*

In optimal control and reinforcement learning we often obtain the policy (i.e. decision rule) ϕ^θ from an approximation of a value function:

$$\phi^\theta(x) = \arg\min_u Q^\theta(x, u)$$

Root Finding — *Not Just About Gradients*

In optimal control and reinforcement learning we often obtain the policy (i.e. decision rule) ϕ^θ from an approximation of a value function:

$$\phi^\theta(x) = \arg\min_u Q^\theta(x, u)$$

The ideal Q^* solves a *dynamic programming equation* [stay tuned]

Root Finding — *Not Just About Gradients*

In optimal control and reinforcement learning we often obtain the policy (i.e. decision rule) ϕ^θ from an approximation of a value function:

$$\phi^\theta(x) = \arg\min_u Q^\theta(x, u)$$

The ideal Q^* solves a *dynamic programming equation*

Algorithm design: construct a function $\bar{f} \colon \mathbb{R}^d \to \mathbb{R}^d$.

Root Finding — *Not Just About Gradients*

In optimal control and reinforcement learning we often obtain the policy (i.e. decision rule) ϕ^θ from an approximation of a value function:

$$\phi^\theta(x) = \arg\min_u Q^\theta(x, u)$$

The ideal Q^* solves a *dynamic programming equation*

Algorithm design: construct a function $\bar{f}\colon \mathbb{R}^d \to \mathbb{R}^d$.
With good design,

$$\bar{f}(\theta^*) = 0 \quad \Longrightarrow \quad Q^{\theta^*} \text{ is an approximate solution.}$$

Estimates
θ_n
$\theta_{n+1} = \theta_n + a_{n+1} f(\theta_n, \xi_{n+1})$
Excitation
-100
0
100
Zap 2
Zap 5
Zap 10

ODE Method

ODE Method

Root finding problem: find solution to $\bar{f}(\theta^*) = 0$

ODE Method

Root finding problem: find solution to $\bar{f}(\theta^*) = 0$

ODE algorithm: $$\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$$ Mean Flow

If stable: $\vartheta_t \to \theta^*$ and $\bar{f}(\vartheta_t) \to \bar{f}(\theta^*) = 0$.

ODE Method

Root finding problem: find solution to $\bar{f}(\theta^*) = 0$

ODE algorithm: $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$ Mean Flow

If stable: $\vartheta_t \to \theta^*$ and $\bar{f}(\vartheta_t) \to \bar{f}(\theta^*) = 0$.

Euler approximation: $\theta_{n+1} = \theta_n + \alpha_{n+1}\bar{f}(\theta_n)$

ODE Method

Root finding problem: find solution to $\bar{f}(\theta^*) = 0$

ODE algorithm: $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$ Mean Flow

If stable: $\vartheta_t \to \theta^*$ and $\bar{f}(\vartheta_t) \to \bar{f}(\theta^*) = 0$.

Euler approximation: $\theta_{n+1} = \theta_n + \alpha_{n+1}\bar{f}(\theta_n)$

Stochastic Approximation

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$$

ODE Method

Root finding problem: find solution to $\bar{f}(\theta^*) = 0$

ODE algorithm: $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$ Mean Flow

If stable: $\vartheta_t \to \theta^*$ and $\bar{f}(\vartheta_t) \to \bar{f}(\theta^*) = 0$.

Euler approximation: $\theta_{n+1} = \theta_n + \alpha_{n+1}\bar{f}(\theta_n)$

Stochastic Approximation

$$\begin{aligned}\theta_{n+1} &= \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1}) \\ &= \theta_n + \alpha_{n+1}\big\{\bar{f}(\theta_n) + \text{“NOISE”}\big\}\end{aligned}$$

Under very general conditions:

the ODE, the Euler approximation, and SA are all convergent to θ^*

ODE Method

Root finding problem: find solution to $\bar{f}(\theta^*) = 0$

ODE algorithm: $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$ Mean Flow

If stable: $\vartheta_t \to \theta^*$ and $\bar{f}(\vartheta_t) \to \bar{f}(\theta^*) = 0$.

Euler approximation: $\theta_{n+1} = \theta_n + \alpha_{n+1}\bar{f}(\theta_n)$

Stochastic Approximation

$$\begin{aligned}\theta_{n+1} &= \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1}) \\ &= \theta_n + \alpha_{n+1}\big\{\bar{f}(\theta_n) + \text{“NOISE”}\big\}\end{aligned}$$

Under very general conditions:

the ODE, the Euler approximation, and SA are all convergent to θ^*

Euler approximation is robust to measurement error

1. Design Mean-Flow $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

We will usually ask for more than GAS

Global exponential asymptotic stability: for positive b, δ,

$$\|\vartheta_t - \theta^*\| \leq b e^{-\delta t} \|\vartheta_0 - \theta^*\|$$

And typically require $A^* = \partial_\theta \bar{f}(\theta^*)$ Hurwitz eigenvalues in left half plane of $\mathbb{C}$

1. Design Mean-Flow $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

We will usually ask for more than GAS
Global exponential asymptotic stability: for positive b, δ,

$$\|\vartheta_t - \theta^*\| \leq b e^{-\delta t} \|\vartheta_0 - \theta^*\|$$

Robustness of an Euler approximation and success of SA also requires that $\bar{f}$ is globally Lipschitz continuous

$$\|\bar{f}(\theta') - \bar{f}(\theta)\| \leq \ell \|\theta' - \theta\| \quad \textit{all } \theta', \theta \in \mathbb{R}^d$$

(as well as similar conditions for f)

1. Design Mean-Flow $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

We will usually ask for more than GAS
Global exponential asymptotic stability: for positive b, δ,

$$\|\vartheta_t - \theta^*\| \le b e^{-\delta t} \|\vartheta_0 - \theta^*\|$$

Robustness of an Euler approximation and success of SA also requires that $\bar{f}$ is globally Lipschitz continuous

Example: $\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$
Spall's SPSA algorithm unlikely to be Lipschitz:

$$f(\theta, \xi) = -\frac{1}{\varepsilon} \xi \Gamma(\theta + \varepsilon \xi)$$

1. Design Mean-Flow $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

We will usually ask for more than GAS
Global exponential asymptotic stability: for positive b, δ,

$$\|\vartheta_t - \theta^*\| \leq b e^{-\delta t} \|\vartheta_0 - \theta^*\|$$

Robustness of an Euler approximation and success of SA also requires that $\bar{f}$ is globally Lipschitz continuous

Example: $\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$
Lipschitz continuity through re-normalization:

$$f(\theta, \xi) = -\frac{1}{\epsilon(\theta)} \xi \Gamma(\theta + \epsilon(\theta)\xi)$$

where the exploration gain grows linearly with $\|\theta\|$ when $\nabla\Gamma$ is Lipshitz.

1. Design Mean-Flow $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

We will usually ask for more than GAS
Global exponential asymptotic stability: for positive b, δ,

$$\|\vartheta_t - \theta^*\| \le b e^{-\delta t} \|\vartheta_0 - \theta^*\|$$

Robustness of an Euler approximation and success of SA also requires that $\bar{f}$ is globally Lipschitz continuous

Example: $\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$
Lipschitz continuity through re-normalization:

$$f(\theta, \xi) = -\frac{1}{\epsilon(\theta)} \xi \Gamma(\theta + \epsilon(\theta)\xi)$$

where the exploration gain grows linearly with $\|\theta\|$ when $\nabla\Gamma$ is Lipshitz.
One choice: $\epsilon(\theta) = \varepsilon\sqrt{1 + \|\theta - \theta^{\text{ctr}}\|^2/\sigma^2}$ [99]

1. Design Mean-Flow $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

We will usually ask for more than GAS
Global exponential asymptotic stability: for positive b, δ,

$$\|\vartheta_t - \theta^*\| \le b e^{-\delta t} \|\vartheta_0 - \theta^*\|$$

Robustness of an Euler approximation and success of SA also requires that $\bar{f}$ is globally Lipschitz continuous

Example: $\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$
Lipschitz continuity through re-normalization:

$$f(\theta, \xi) = -\frac{1}{\epsilon(\theta)} \xi \Gamma(\theta + \epsilon(\theta)\xi)$$

where the exploration gain grows linearly with $\|\theta\|$ when $\nabla\Gamma$ is Lipshitz.
One choice: $\epsilon(\theta) = \varepsilon\sqrt{1 + \|\theta - \theta^{\text{ctr}}\|^2/\sigma^2}$ [99]

Similar tricks used in [83] for Oja's algorithm.

2. Design Step-Size

Standard assumption for non-negative step-size sequence $\{\alpha_n\}$:

$$\sum_{n=1}^{\infty} \alpha_n = \infty \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

Typical choice: $\alpha_n = g/(n_e + n)^\rho \quad \frac{1}{2} < \rho < 1$

2. Design Step-Size

Standard assumption for non-negative step-size sequence $\{\alpha_n\}$:

$$\sum_{n=1}^{\infty} \alpha_n = \infty \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

The second assumption appears necessary for the traditional model

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1}) = \theta_n + \alpha_{n+1} \{ \bar{f}(\theta_n) + \Delta_{n+1} \}$$

in which $\{\Delta_{n+1}\}$ is a martingale-difference sequence.

2. Design Step-Size

Standard assumption for non-negative step-size sequence $\{\alpha_n\}$:

$$\sum_{n=1}^{\infty} \alpha_n = \infty \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

New in 2024: $\alpha_n = g/(n_e + n)^\rho \quad 0 < \rho < 1$

2. Design Step-Size

Standard assumption for non-negative step-size sequence $\{\alpha_n\}$:

$$\sum_{n=1}^{\infty} \alpha_n = \infty \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

New in 2024: $\alpha_n = g/(n_e + n)^\rho \quad 0 < \rho < 1$

$\sum_{n=1}^{\infty} \alpha_n^2 = \infty$ if $\rho < 1/2$, but this does not impact convergence [79]

2. Design Step-Size

Standard assumption for non-negative step-size sequence $\{\alpha_n\}$:

$$\sum_{n=1}^{\infty} \alpha_n = \infty \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

New in 2024: $\alpha_n = g/(n_e + n)^\rho \quad 0 < \rho < 1$

A warning from [79]: subject to assumptions

$$\lim_{n\to\infty} \frac{1}{\alpha_n} \mathsf{Cov}\left(\theta_n\right) = \Sigma_\theta$$
$$\lim_{n\to\infty} \frac{1}{\alpha_n} [\theta_n - \theta^*] = \nu_\theta$$

2. Design Step-Size

Standard assumption for non-negative step-size sequence $\{\alpha_n\}$:

$$\sum_{n=1}^{\infty} \alpha_n = \infty \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

New in 2024: $\alpha_n = g/(n_e + n)^\rho \quad 0 < \rho < 1$

A warning from [79]: subject to assumptions

$$\lim_{n\to\infty} \frac{1}{\alpha_n} \mathsf{Cov}\left(\theta_n\right) = \Sigma_\theta$$
$$\lim_{n\to\infty} \frac{1}{\alpha_n} [\theta_n - \theta^*] = \nu_\theta$$

Good news: $\nu_\theta = 0$ for the "additive noise" model
In this case $\rho = 0$ is sometimes acceptable [80, 78] (but I do not see motivation)

2. Design Step-Size

Standard assumption for non-negative step-size sequence $\{\alpha_n\}$:

$$\sum_{n=1}^{\infty} \alpha_n = \infty \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

New in 2024: $\alpha_n = g/(n_e + n)^\rho \quad 0 < \rho < 1$

A warning from [79]: subject to assumptions

$$\lim_{n\to\infty} \frac{1}{\alpha_n} \mathsf{Cov}\left(\theta_n\right) = \Sigma_\theta$$
$$\lim_{n\to\infty} \frac{1}{\alpha_n} [\theta_n - \theta^*] = \nu_\theta$$

If $\nu_\theta \neq 0$ than bias can dominate variance in MSE when $\rho < 1/2$, even with the application of the filtering tricks to be described next

3. Filter Estimates

Polyak-Ruppert averaging: $$\theta_N^{\mathsf{PR}} = \frac{1}{N - N_0} \sum_{n=N_0+1}^{N} \theta_n$$

Ruppert & Polyak [88, 89, 87]

3. Filter Estimates

Polyak-Ruppert averaging: $$\theta_N^{\mathsf{PR}} = \frac{1}{N - N_0} \sum_{n=N_0+1}^{N} \theta_n$$

Consider $\alpha_n = g/(n_e + n)^\rho$.
Subject to assumptions (including $A^* = \partial_\theta \bar{f}(\theta^*)$ Hurwitz)

$$\lim_{n\to\infty} n \,\mathsf{Cov}\,(\theta_n^{\mathsf{PR}}) = \Sigma_\theta^*$$

$$\lim_{n\to\infty} \frac{1}{\alpha_n}[\theta_n^{\mathsf{PR}} - \theta^*] = \nu_\theta \frac{1}{1-\rho}$$

Ruppert & Polyak [88, 89, 87]

3. Filter Estimates

Polyak-Ruppert averaging: $\theta_N^{\mathsf{PR}} = \dfrac{1}{N - N_0} \displaystyle\sum_{n=N_0+1}^{N} \theta_n$

Consider $\alpha_n = g/(n_e + n)^\rho$.
Subject to assumptions (including $A^* = \partial_\theta \bar{f}\,(\theta^*)$ Hurwitz)

$$\lim_{n\to\infty} n\,\mathsf{Cov}\,(\theta_n^{\mathsf{PR}}) = \Sigma_\theta^*$$

$$\lim_{n\to\infty} \frac{1}{\alpha_n}[\theta_n^{\mathsf{PR}} - \theta^*] = \nu_\theta \frac{1}{1-\rho}$$

Implications of averaging:

- Can only amplify bias, and we must assume $\rho < 1$

Ruppert & Polyak [88, 89, 87]

3. Filter Estimates

Polyak-Ruppert averaging: $\theta_N^{\mathsf{PR}} = \dfrac{1}{N - N_0} \displaystyle\sum_{n=N_0+1}^{N} \theta_n$

Consider $\alpha_n = g/(n_e + n)^\rho$.
Subject to assumptions (including $A^* = \partial_\theta \bar{f}(\theta^*)$ Hurwitz)

$$\lim_{n\to\infty} n \,\mathsf{Cov}\,(\theta_n^{\mathsf{PR}}) = \Sigma_\theta^*$$

$$\lim_{n\to\infty} \frac{1}{\alpha_n}[\theta_n^{\mathsf{PR}} - \theta^*] = \nu_\theta \frac{1}{1-\rho}$$

Implications of averaging:
- Can only amplify bias, and we must assume $\rho < 1$
- Covariance decays as $1/n$ — significant acceleration

Ruppert & Polyak [88, 89, 87]

3. Filter Estimates

Polyak-Ruppert averaging: $\theta_N^{\mathsf{PR}} = \dfrac{1}{N - N_0} \displaystyle\sum_{n=N_0+1}^{N} \theta_n$

Consider $\alpha_n = g/(n_e + n)^\rho$.
Subject to assumptions (including $A^* = \partial_\theta \bar{f}(\theta^*)$ Hurwitz)

$$\lim_{n\to\infty} n \,\mathsf{Cov}\,(\theta_n^{\mathsf{PR}}) = \Sigma_\theta^*$$

$$\lim_{n\to\infty} \frac{1}{\alpha_n}[\theta_n^{\mathsf{PR}} - \theta^*] = \nu_\theta \frac{1}{1-\rho}$$

Implications of averaging:
- Can only amplify bias, and we must assume $\rho < 1$
- Covariance decays as $1/n$ — significant acceleration
- The asymptotic covariance matrix Σ_θ^* is minimal in a matricial sense

Ruppert & Polyak [88, 89, 87]

Asymptotic Statistics

Optimality of PR-Averaging

$$\mathsf{Cov}(\theta_N^{\mathsf{PR}}) = \frac{1}{N - N_0}\Sigma_N^{\mathsf{PR}}$$

$$\Sigma_N^{\mathsf{PR}} \to \Sigma_\theta^* = [A^*]^{-1}\Sigma_\Delta[A^{*T}]^{-1}$$ minimal in strongest sense

With $\Delta_n^* = f(\theta^*, \xi_n)$:

$$\Sigma_\Delta \stackrel{\text{def}}{=} \sum_{n=-\infty}^{\infty} \mathsf{E}[\Delta_0^* \Delta_n^{*T}]$$

Asymptotic Statistics

Optimality of PR-Averaging

$$\mathsf{Cov}(\theta_N^{\mathsf{PR}}) = \frac{1}{N - N_0}\Sigma_N^{\mathsf{PR}}$$

$$\Sigma_N^{\mathsf{PR}} \to \Sigma_\theta^* = [A^*]^{-1}\Sigma_\Delta[A^{*T}]^{-1} \text{ minimal in strongest sense}$$

If either $\nu_\theta = 0$ or $\rho \in (1/2, 1)$, then
The Central Limit Theorem holds *tremendous tool for validation, ...*

Histogram from 10^3 independent runs

More Rules

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$$

There are of course a few more assumptions

More Rules

$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$

There are of course a few more assumptions
The weakest assumptions in [77] require
• $\xi_{n+1} = G_0(\Phi_{n+1})$ in which $\mathbf{\Phi}$ evolves on a general state space X, and satisfies a conditional Markov property:

$$\mathsf{P}\{\Phi_{n+1} \in A \mid \mathcal{F}_n; \theta_n = \theta\,,\ \Phi_n = x\} = P_\theta(x, A)\,, \quad n \geq 0\,,$$

for each (measurable) $A \subset \mathsf{X}$, $x \in \mathsf{X}$, and $\theta \in \mathbb{R}^d$.

More Rules

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$$

There are of course a few more assumptions
The weakest assumptions in [77] require
• $\xi_{n+1} = G_0(\Phi_{n+1})$ in which $\mathbf{\Phi}$ evolves on a general state space X, and satisfies a conditional Markov property:

$$\mathsf{P}\{\Phi_{n+1} \in A \mid \mathcal{F}_n; \theta_n = \theta\,,\ \Phi_n = x\} = P_\theta(x, A)\,, \quad n \geq 0\,,$$

for each (measurable) $A \subset \mathsf{X}$, $x \in \mathsf{X}$, and $\theta \in \mathbb{R}^d$.
• The transition kernels $\{P_\theta\}$ are Lipschitz continuous in θ

More Rules

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$$

There are of course a few more assumptions
The weakest assumptions in [77] require
• $\xi_{n+1} = G_0(\Phi_{n+1})$ in which $\mathbf{\Phi}$ evolves on a general state space X, and satisfies a conditional Markov property:

$$\mathsf{P}\{\Phi_{n+1} \in A \mid \mathcal{F}_n; \theta_n = \theta\,,\ \Phi_n = x\} = P_\theta(x, A)\,, \quad n \geq 0\,,$$

for each (measurable) $A \subset \mathsf{X}$, $x \in \mathsf{X}$, and $\theta \in \mathbb{R}^d$.
• The transition kernels $\{P_\theta\}$ are Lipschitz continuous in θ
• The Markov chain with transition kernel P_θ satisfies an ergodicity assumption. If X is finite it is sufficient that the Markov chain with transition matrix P_θ is aperiodic and irreducible for each θ (some uniformity is required).
In this case there is a unique invariant pmf (prob. mass function) π_θ and

$$\bar{f}(\theta) = \sum_x f(\theta, x)\, \pi_\theta(x)\,, \quad \theta \in \mathbb{R}^d$$

$$\alpha_n = g/(n_e + n)^\rho$$

Unveiling Dynamics

SA Error $\theta_{n+1} = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \text{“NOISE”}\}$ $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

1 Asymptotic Covariance:

$\alpha_n = \frac{g}{(n_e+n)^\rho}$

$\frac{1}{\sqrt{\alpha_n}}[\theta_n - \vartheta_{\tau_n}] \approx N(0,\Sigma)$ *without averaging*

Using $\alpha_{n+1} = \frac{g}{(n_e+n)^\rho}$:

$$\tau_n \approx \begin{cases} g\log(n) & \rho = 1 \\ \frac{g}{1-\rho} n\alpha_n & \rho < 1 \end{cases}$$

SA Error $\theta_{n+1} = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \text{“NOISE”}\}$ $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

1 $\frac{1}{\sqrt{\alpha_n}}[\theta_n - \vartheta_{\tau_n}] \approx N(0, \Sigma)$ *without averaging* where $\tau_n = \sum_{k=1}^{n} \alpha_k$

Using $\alpha_{n+1} = \frac{g}{(n_e+n)^\rho}$:

$$\tau_n \approx \begin{cases} g\log(n) & \rho = 1 \\ \frac{g}{1-\rho} n\alpha_n & \rho < 1 \end{cases}$$

SA Error $\theta_{n+1} = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \text{“NOISE”}\}$ $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

1 $\frac{1}{\sqrt{\alpha_n}}[\theta_n - \vartheta_{\tau_n}] \approx N(0, \Sigma)$ *without averaging* where $\tau_n = \sum_{k=1}^{n} \alpha_k$

2 $\vartheta_t \to \theta^*$ exponentially fast, but

Using $\alpha_{n+1} = \frac{g}{(n_e+n)^\rho}$:

$$\tau_n \approx \begin{cases} g\log(n) & \rho = 1 \\ \frac{g}{1-\rho} n\alpha_n & \rho < 1 \end{cases}$$

SA Error $\theta_{n+1} = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \text{“NOISE”}\}$ $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

1 $\frac{1}{\sqrt{\alpha_n}}[\theta_n - \vartheta_{\tau_n}] \approx N(0, \Sigma)$ *without averaging* where $\tau_n = \sum_{k=1}^{n} \alpha_k$

2 $\vartheta_t \to \theta^*$ exponentially fast, but τ_n is increasing slowly, *and*

Using $\alpha_{n+1} = \frac{g}{(n_e+n)^\rho}$:

$$\tau_n \approx \begin{cases} g\log(n) & \rho = 1 \\ \frac{g}{1-\rho} n\alpha_n & \rho < 1 \end{cases}$$

SA Error $\theta_{n+1} = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \text{“NOISE”}\}$ $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

1 $\frac{1}{\sqrt{\alpha_n}}[\theta_n - \vartheta_{\tau_n}] \approx N(0, \Sigma)$ *without averaging* where $\tau_n = \sum_{k=1}^{n} \alpha_k$

2 $\vartheta_t \to \theta^*$ exponentially fast, but τ_n is increasing slowly,
and nonlinear dynamics can complicate gain selection

Using $\alpha_{n+1} = \frac{g}{(n_e+n)^\rho}$:

$$\tau_n \approx \begin{cases} g\log(n) & \rho = 1 \\ \frac{g}{1-\rho} n\alpha_n & \rho < 1 \end{cases}$$

SA Error $\theta_{n+1} = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \text{“NOISE”}\}$ $\frac{d}{dt}\vartheta_t = \bar{f}(\vartheta_t)$

1 $\frac{1}{\sqrt{\alpha_n}}[\theta_n - \vartheta_{\tau_n}] \approx N(0, \Sigma)$ *without averaging* where $\tau_n = \sum_{k=1}^{n} \alpha_k$

2 $\vartheta_t \to \theta^*$ exponentially fast, but τ_n is increasing slowly,
and nonlinear dynamics can complicate gain selection

What can happen in applications when $\rho = 1$:

- θ_n far from θ^*, the dynamics are slow, need large g
- $\theta_n \approx \theta^*$, *best gain is far smaller*

Using $\alpha_{n+1} = \frac{g}{(n_e+n)^\rho}$:

$$\tau_n \approx \begin{cases} g\log(n) & \rho = 1 \\ \frac{g}{1-\rho} n\alpha_n & \rho < 1 \end{cases}$$

Example: SGD Section 8.3 CS&RL

$\alpha_n = g/(n_e + n)^\rho$

Stochastic Gradient Descent:

$$\bar{L}(\theta) = \mathsf{E}[L(\theta, \Phi_n)]$$

$$\bar{f}(\theta) = -\nabla \bar{L}(\theta)$$

$$\theta_{n+1} = \theta_n - \frac{g}{n^\rho} \nabla L(\theta_n, \Phi_{n+1})$$

Example: SGD Section 8.3 CS&RL

$\alpha_n = g/(n_e + n)^\rho$

$$\bar{f}(\theta) = -\nabla \bar{L}(\theta)$$

ODE bound using $\rho = 1$ and setting $n_e = 1$

$$|\vartheta_{\tau_n} - \theta^*| \le |\vartheta_0 - \theta^*| e^{0.34g} n^{-0.34g}$$

$$\theta_{n+1} = \theta_n - \frac{g}{n^\rho} \nabla L(\theta_n, \Phi_{n+1})$$

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

$$|\vartheta_t - \theta^*| \le |\vartheta_0 - \theta^*| e^{-0.34t}$$

Example: SGD Section 8.3 CS&RL

$\alpha_n = g/(n_e + n)^\rho$

$$\bar{f}(\theta) = -\nabla \bar{L}(\theta)$$

ODE bound using $\rho = 1$ and setting $n_e = 1$

$$|\vartheta_{\tau_n} - \theta^*| \le |\vartheta_0 - \theta^*| e^{0.34g} n^{-0.34g}$$

$g \ge 3$ to kill deterministic behavior,
but $g^* = -[A^*]^{-1} = 1/4$ gives optimal variance

$$\theta_{n+1} = \theta_n - \frac{g}{n^\rho} \nabla L(\theta_n, \Phi_{n+1})$$

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

$$|\vartheta_t - \theta^*| \le |\vartheta_0 - \theta^*| e^{-0.34t}$$

Example: SGD Section 8.3 CS&RL

$\alpha_n = g/(n_e + n)^\rho$

$$\bar{f}(\theta) = -\nabla \bar{L}(\theta)$$

ODE bound using $\rho = 1$ and setting $n_e = 1$

$$|\vartheta_{\tau_n} - \theta^*| \le |\vartheta_0 - \theta^*| e^{0.34g} n^{-0.34g}$$

$g \ge 3$ to kill deterministic behavior,
but $g^* = 1/4$ gives optimal variance

$$\theta_{n+1} = \theta_n - \frac{g}{n^\rho} \nabla L(\theta_n, \Phi_{n+1})$$

Dynamics for $g^* = 1/4$

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

$$|\vartheta_t - \theta^*| \le |\vartheta_0 - \theta^*| e^{-0.34t}$$

$\tau_N < 3$ for $N =$ *one million*

Example: SGD Section 8.3 CS&RL

$\alpha_n = g/(n_e + n)^\rho$

$$\bar{f}(\theta) = -\nabla \bar{L}(\theta)$$

ODE bound using $\rho = 1$ and setting $n_e = 1$

$$|\vartheta_{\tau_n} - \theta^*| \le |\vartheta_0 - \theta^*| e^{0.34g} n^{-0.34g}$$

$g \ge 3$ to kill deterministic behavior,
but $g^* = 1/4$ gives optimal variance

$$\theta_{n+1} = \theta_n - \frac{g}{n^\rho} \nabla L(\theta_n, \Phi_{n+1})$$

Dynamics for $g^* = 1/4$

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

$$|\vartheta_t - \theta^*| \le |\vartheta_0 - \theta^*| e^{-0.34t}$$

$\tau_N < 3$ for $N =$ *one million*

CLT approximation: rapid for $\theta_0 = 0$

Example: SGD Section 8.3 CS&RL

$\alpha_n = g/(n_e + n)^\rho$

$$\bar{f}(\theta) = -\nabla \bar{L}(\theta)$$

ODE bound using $\rho = 1$ and setting $n_e = 1$

$$|\vartheta_{\tau_n} - \theta^*| \le |\vartheta_0 - \theta^*| e^{0.34g} n^{-0.34g}$$

$g \ge 3$ to kill deterministic behavior,
but $g^* = 1/4$ gives optimal variance

$$\theta_{n+1} = \theta_n - \frac{g}{n^\rho} \nabla L(\theta_n, \Phi_{n+1})$$

Dynamics for $g^* = 1/4$

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

$$|\vartheta_t - \theta^*| \le |\vartheta_0 - \theta^*| e^{-0.34t}$$

$\tau_N < 3$ for $N =$ *one million*

CLT approximation: rapid for $\theta_0 = 0$ slow for $\theta_0 = 100$

Example: SGD Section 8.3 CS&RL

$\alpha_n = g/(n_e + n)^\rho$

$$\bar{f}(\theta) = -\nabla \bar{L}(\theta)$$

ODE bound using $\rho = 1$ and setting $n_e = 1$

$$|\vartheta_{\tau_n} - \theta^*| \le |\vartheta_0 - \theta^*| e^{0.34g} n^{-0.34g}$$

$$\theta_{n+1} = \theta_n - \frac{g}{n^\rho} \nabla L(\theta_n, \Phi_{n+1})$$

Polyak-Ruppert to the rescue

Histograms from Polyak-Ruppert averaging: big and small g

$\tilde{\theta}_n = \theta_n - \theta^*$; $\tilde{\theta}_n = 0$

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

$$|\vartheta_t - \theta^*| \le |\vartheta_0 - \theta^*| e^{-0.34t}$$

Two Sources of Error. Example: Tabular Q-Learning

$g \geq 1/(1-\gamma)$ required, if using $\rho = 1$*

$$\alpha_n = \frac{g}{n(x,u)}$$

Awesome performance! (?)

Generic tabular Q-learning example. Discount factor γ

Two Sources of Error. Example: Tabular Q-Learning

$g \geq 1/(1-\gamma)$ required, if using $\rho = 1$* *Stay tuned!*

Generic tabular Q-learning example. Discount factor γ

*See Devraj & M 2017 (extension in Wainwright 2019); see also Szepesvári 1997

Zap

Taming Nonlinear Dynamics

What if stability of $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ is unknown? [typically the case in RL]

Newton Raphson Flow [Smale 1976]

Idea: Interpret $\bar{f}$ as the "parameter": $\frac{d}{dt}\bar{f}_t = \mathcal{V}(\bar{f}_t)$ and design $\mathcal{V}$

Taming Nonlinear Dynamics

What if stability of $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ is unknown? [typically the case in RL]

Newton Raphson Flow [Smale 1976]

Idea: Interpret $\bar{f}$ as the "parameter": $\frac{d}{dt}\bar{f}_t = -\bar{f}_t$

$$\frac{d}{dt}\bar{f}(\vartheta_t) = -\bar{f}(\vartheta_t) \qquad \textit{giving} \quad \bar{f}(\vartheta_t) = \bar{f}(\vartheta_0)e^{-t}$$

Linear dynamics

Taming Nonlinear Dynamics

What if stability of $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ is unknown? [typically the case in RL]

Newton Raphson Flow [Smale 1976]

Idea: Interpret $\bar{f}$ as the "parameter": $\frac{d}{dt}\bar{f}_t = -\bar{f}_t$

$$\frac{d}{dt}\bar{f}(\vartheta_t) = -\bar{f}(\vartheta_t) \qquad \frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$$

SA translation: Zap Stochastic Approximation

Zap Algorithm

Designed to emulate Newton-Raphson flow

$\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t), \quad A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$

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

$$\theta_{n+1} = \theta_n + \alpha_{n+1}[-\widehat{A}_{n+1}]^{-1} f(\theta_n, \xi_{n+1})$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \beta_{n+1}(A_{n+1} - \widehat{A}_n), \qquad A_{n+1} = \partial_\theta f(\theta_n, \xi_{n+1})$$

Zap Algorithm

Designed to emulate Newton-Raphson flow

$\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t), \quad A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$

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

$$\theta_{n+1} = \theta_n + \alpha_{n+1}[-\widehat{A}_{n+1}]^{-1} f(\theta_n, \xi_{n+1})$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \beta_{n+1}(A_{n+1} - \widehat{A}_n), \qquad A_{n+1} = \partial_\theta f(\theta_n, \xi_{n+1})$$

Requires $\widehat{A}_{n+1} \approx A(\theta_n) \stackrel{\text{def}}{=} \partial_\theta \bar{f}\,(\theta_n)$

Zap Algorithm

Designed to emulate Newton-Raphson flow

$\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t), \quad A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$

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

$$\theta_{n+1} = \theta_n + \alpha_{n+1}[-\widehat{A}_{n+1}]^{-1} f(\theta_n, \xi_{n+1})$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \beta_{n+1}(A_{n+1} - \widehat{A}_n), \qquad A_{n+1} = \partial_\theta f(\theta_n, \xi_{n+1})$$

$\widehat{A}_{n+1} \approx A(\theta_n)$ requires high-gain, $\dfrac{\beta_n}{\alpha_n} \to \infty, \qquad n \to \infty$

Zap Algorithm

Designed to emulate Newton-Raphson flow

$\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t), \quad A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$

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

$$\theta_{n+1} = \theta_n + \alpha_{n+1}[-\widehat{A}_{n+1}]^{-1} f(\theta_n, \xi_{n+1})$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \beta_{n+1}(A_{n+1} - \widehat{A}_n), \qquad A_{n+1} = \partial_\theta f(\theta_n, \xi_{n+1})$$

$\widehat{A}_{n+1} \approx A(\theta_n)$ requires high-gain, $\dfrac{\beta_n}{\alpha_n} \to \infty, \qquad n \to \infty$

Can use $\alpha_n = 1/n$ (without averaging).

Numerics to come: use this choice, and $\beta_n = (1/n)^\rho$, $\rho \in (0.5, 1)$

Zap Algorithm

Designed to emulate Newton-Raphson flow

$\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t), \quad A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$

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

$$\theta_{n+1} = \theta_n + \alpha_{n+1}[-\widehat{A}_{n+1}]^{-1} f(\theta_n, \xi_{n+1})$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \beta_{n+1}(A_{n+1} - \widehat{A}_n), \qquad A_{n+1} = \partial_\theta f(\theta_n, \xi_{n+1})$$

$\widehat{A}_{n+1} \approx A(\theta_n)$ requires high-gain, $\dfrac{\beta_n}{\alpha_n} \to \infty, \qquad n \to \infty$

Can use $\alpha_n = 1/n$ (without averaging).

Numerics to come: use this choice, and $\beta_n = (1/n)^\rho$, $\rho \in (0.5, 1)$

Stability? *Virtually universal*

Optimal variance, too! (in sense of Ruppert & Polyak)

Thank you Lyapunov and Polyak!

Crank Up Gain: Tame Transients

Average: Tame Variance

Steps to a Successful Design

1. Design $\bar{f}$ for mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ GAS, Hurwitz A^*

Thank you Lyapunov and Polyak!

Crank Up Gain: Tame Transients

Average: Tame Variance

Steps to a Successful Design

1. Design $\bar{f}$ for mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ GAS, Hurwitz A^*
2. Design step-size: $\alpha_n = g/(n_e + n)^\rho$ $\frac{1}{2} < \rho < 1$

Thank you Lyapunov and Polyak!

Crank Up Gain: Tame Transients

Average: Tame Variance

Steps to a Successful Design

1. Design $\bar{f}$ for mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ GAS, Hurwitz A^*
2. Design step-size: $\alpha_n = g/(n_e + n)^\rho$ $\frac{1}{2} < \rho < 1$
3. Perform PR Averaging

$$\theta_N^{\mathrm{PR}} = \frac{1}{N - N_0} \sum_{n=N_0+1}^{N} \theta_n$$

Thank you Lyapunov and Polyak!

Crank Up Gain: Tame Transients

Average: Tame Variance

Steps to a Successful Design

1. Design $\bar{f}$ for mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ GAS, Hurwitz A^*
2. Design step-size: $\alpha_n = g/(n_e + n)^\rho$ $\frac{1}{2} < \rho < 1$
3. Perform PR Averaging
4. Repeat!

Thank you Lyapunov and Polyak!

Crank Up Gain: Tame Transients

Average: Tame Variance

Steps to a Successful Design

1. Design $\bar{f}$ for mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ GAS, Hurwitz A^*
2. Design step-size: $\alpha_n = g/(n_e + n)^\rho$ $\frac{1}{2} < \rho < 1$
3. Perform PR Averaging
4. Repeat! Obtain histogram

$$\left\{\sqrt{N - N_0}\,\tilde{\theta}_N^{\mathrm{PR}\,(m)} : 1 \le m \le M\right\}$$

with $\theta_0^{(m)}$ widely dispersed and N relatively small

Provides estimates of Σ_θ^*

Thank you Lyapunov and Polyak!

Crank Up Gain: Tame Transients

Average: Tame Variance

Steps to a Successful Design

1. Design $\bar{f}$ for mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ GAS, Hurwitz A^*
2. Design step-size: $\alpha_n = g/(n_e + n)^\rho$ $\frac{1}{2} < \rho < 1$
3. Perform PR Averaging
4. Repeat! Obtain histogram

$$\left\{\sqrt{N - N_0}\,\tilde{\theta}_N^{\mathsf{PR}\,(m)} : 1 \le m \le M\right\}$$

with $\theta_0^{(m)}$ widely dispersed and N relatively small

Provides estimates of Σ_θ^*

What you will learn:

- How big N needs to be for a meaningful estimate
- Approximate confidence bounds

Heuristic: $\theta_N^{\mathsf{PR}} \approx \theta^* + \frac{1}{\sqrt{N - N_0}} Z, \quad Z \sim N(0, \Sigma_\theta^*)$

Thank you Lyapunov and Polyak!

Crank Up Gain: Tame Transients

Average: Tame Variance

Steps to a Successful Design

1. Design $\bar{f}$ for mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ GAS, Hurwitz A^*
2. Design step-size: $\alpha_n = g/(n_e + n)^\rho$ $\frac{1}{2} < \rho < 1$
3. Perform PR Averaging
4. Repeat! Obtain histogram

$$\left\{\sqrt{N - N_0}\,\tilde{\theta}_N^{\mathrm{PR}\,(m)} : 1 \le m \le M\right\}$$

with $\theta_0^{(m)}$ widely dispersed and N relatively small

Provides estimates of Σ_θ^*

What you will learn:

- How big N needs to be for a meaningful estimate
- Approximate confidence bounds

Next Steps:

- A flash-crash control course!
- Applications to RL

Skip to Part 2

Control Techniques for Complex Networks page 528

Example 11.5.1. LSTD for the M/M/1 queue **(minimum variance algorithm)**

Nonsense after One Million Samples

Appendix: Understanding Variance

Perturbative Mean Flow

$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$

Prerequisites

- Mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ globally asymptotically stable
  So $\vartheta_t \to \theta^*$ from each initial condition

  A tiny bit more is needed to ensure $\{\theta_n\}$ is bounded – see CS&RL for details!
- $A^* = \partial_\theta \bar{f}\left(\theta^*\right)$ Hurwitz eigenvalues in left half plane of $\mathbb{C}$

Perturbative Mean Flow

$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1})$

Prerequisites

- Mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ globally asymptotically stable
  So $\vartheta_t \to \theta^*$ from each initial condition

  A tiny bit more is needed to ensure $\{\theta_n\}$ is bounded – see CS&RL for details!
- $A^* = \partial_\theta \bar{f}\,(\theta^*)$ Hurwitz eigenvalues in left half plane of $\mathbb{C}$
- $\xi_{n+1} = G_0(\Phi_{n+1})$, with $\mathbf{\Phi}$ a "nice Markov chain"

  Assume here: finite state space and irreducible

Please don't rule out periodicity!

There is a strong theory for quasi-stochastic approximation in which $\mathbf{\Phi}$ is deterministic [98, 99, 100]

P- Mean Flow

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1}) = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \Delta_n\}$$

Perturbative Mean Flow (details for fixed step-size)

Two distinctions from QSA theory of deterministic [98, 99, 100]: 1. Time is discrete, and 2. We obtain an expression for the sum of the NOISE:

$$\sum_{n=0}^{N-1} \Delta_n =$$

Foundation of all is *Poisson's equation*: $\{\widehat{f}_n(\theta) - \mathsf{E}[\widehat{f}_{n+1}(\theta) \mid \mathcal{F}_n]\}\big|_{\theta_{n-1}} = \Delta_{n-1}$

P- Mean Flow

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1}) = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \Delta_n\}$$

Perturbative Mean Flow (details for fixed step-size)

Two distinctions from QSA theory of deterministic [98, 99, 100]: 1. Time is discrete, and 2. We obtain an expression for the sum of the NOISE:

$$\sum_{n=0}^{N-1} \Delta_n = \sum_{n=2}^{N+1} \mathcal{W}_n$$

- $\{\mathcal{W}_n\}$ is white—beautiful theory waiting for us

Foundation of all is *Poisson's equation*: $\{\widehat{f}_n(\theta) - \mathsf{E}[\widehat{f}_{n+1}(\theta) \mid \mathcal{F}_n]\}\big|_{\theta_{n-1}} = \Delta_{n-1}$

P- Mean Flow

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1}) = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \Delta_n\}$$

Perturbative Mean Flow (details for fixed step-size)

Two distinctions from QSA theory of deterministic [98, 99, 100]: 1. Time is discrete, and 2. We obtain an expression for the sum of the NOISE:

$$\sum_{n=0}^{N-1} \Delta_n = \sum_{n=2}^{N+1} \mathcal{W}_n + \Delta \widehat{f}_N$$

- $\{\mathcal{W}_n\}$ is white—beautiful theory waiting for us
- Uniformly bounded—adds $O(1/N)$ in convergence rate.

Foundation of all is *Poisson's equation*: $\{\widehat{f}_n(\theta) - \mathsf{E}[\widehat{f}_{n+1}(\theta) \mid \mathcal{F}_n]\}\big|_{\theta_{n-1}} = \Delta_{n-1}$

P- Mean Flow

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1}) = \theta_n + \alpha_{n+1}\{\bar{f}(\theta_n) + \Delta_n\}$$

Perturbative Mean Flow (details for fixed step-size)

Two distinctions from QSA theory of deterministic [98, 99, 100]: 1. Time is discrete, and 2. We obtain an expression for the sum of the NOISE:

$$\sum_{n=0}^{N-1} \Delta_n = \sum_{n=2}^{N+1} \mathcal{W}_n + \Delta \widehat{f}_N$$

$$- \alpha \sum_{n=1}^{N} \Upsilon_n$$

- $\{\mathcal{W}_n\}$ is white—beautiful theory waiting for us
- Uniformly bounded—adds $O(1/N)$ in convergence rate.
- Bad news for bias: $\Upsilon_n = -\frac{1}{\alpha}\left[\widehat{f}_{n+1}(\theta_n) - \widehat{f}_{n+1}(\theta_{n-1})\right]$

Foundation of all is *Poisson's equation*: $\left\{\widehat{f}_n(\theta) - \mathsf{E}[\widehat{f}_{n+1}(\theta) \mid \mathcal{F}_n]\right\}\Big|_{\theta_{n-1}} = \Delta_{n-1}$

Design Implications $\theta_{n+1} - \theta_n = \alpha\{\bar{f}(\theta_n) + \Delta_n\}$ Constant Step-size

$$\sum_{n=0}^{N-1} \Delta_n = \sum_{n=2}^{N+1} \mathcal{W}_n + \Delta \widehat{f}_N - \alpha \sum_{n=1}^{N} \Upsilon_n$$

- $\mathsf{E}[\|\theta_n\|^2]$ uniformly bounded for $0 < \alpha \le \alpha^0$ some $\alpha^0 > 0$

Design Implications $\theta_{n+1} - \theta_n = \alpha\{\bar{f}(\theta_n) + \Delta_n\}$ Constant Step-size

$$\sum_{n=0}^{N-1} \Delta_n = \sum_{n=2}^{N+1} \mathcal{W}_n + \Delta\widehat{f}_N - \alpha \sum_{n=1}^{N} \Upsilon_n$$

- $\mathsf{E}[\|\theta_n\|^2]$ uniformly bounded for $0 < \alpha \le \alpha^0$ some $\alpha^0 > 0$
- Justifies $\theta_n = X_n + O(\alpha^2)$ (approximation in mean-square)

$$X_{n+1} - X_n = \alpha[A^*[X_n - \theta^*] + \Delta_n]$$

Design Implications $\theta_{n+1} - \theta_n = \alpha\{\bar{f}(\theta_n) + \Delta_n\}$ Constant Step-size

$$\sum_{n=0}^{N-1} \Delta_n = \sum_{n=2}^{N+1} \mathcal{W}_n + \Delta\widehat{f}_N - \alpha \sum_{n=1}^{N} \Upsilon_n$$

- $\mathsf{E}[\|\theta_n\|^2]$ uniformly bounded for $0 < \alpha \le \alpha^0$ some $\alpha^0 > 0$
- Justifies $\theta_n = X_n + O(\alpha^2)$ (approximation in mean-square)

$$\sum_{n=0}^{N-1} \big(X_{n+1} - X_n\big) = \sum_{n=0}^{N-1} \big(\alpha[A^*[X_n - \theta^*] + \Delta_n]\big)$$

Design Implications $\theta_{n+1} - \theta_n = \alpha\{\bar{f}(\theta_n) + \Delta_n\}$ Constant Step-size

$$\sum_{n=0}^{N-1} \Delta_n = \sum_{n=2}^{N+1} \mathcal{W}_n + \Delta\widehat{f}_N - \alpha \sum_{n=1}^{N} \Upsilon_n$$

- $\mathsf{E}[\|\theta_n\|^2]$ uniformly bounded for $0 < \alpha \le \alpha^0$ some $\alpha^0 > 0$
- Justifies $\theta_n = X_n + O(\alpha^2)$ (approximation in mean-square)

$$\sum_{n=0}^{N-1} \big(X_{n+1} - X_n\big) = \sum_{n=0}^{N-1} \big(\alpha[A^*[X_n - \theta^*] + \Delta_n]\big)$$

See a Polyak-Ruppert filter pop out?

Design Implications $\theta_{n+1} - \theta_n = \alpha\{\bar{f}(\theta_n) + \Delta_n\}$ Constant Step-size

$$\sum_{n=0}^{N-1} \Delta_n = \sum_{n=2}^{N+1} \mathcal{W}_n + \Delta \widehat{f}_N - \alpha \sum_{n=1}^{N} \Upsilon_n$$

$$\frac{1}{N} \sum_{n=0}^{N-1} \big(\theta_{n+1} - \theta_n\big) = \alpha \frac{1}{N} \sum_{n=0}^{N-1} \big(A^*[\theta_n - \theta^*] + \Delta_n + O(\alpha^2)\big)$$

Design Implications $\theta_{n+1} - \theta_n = \alpha\{\bar{f}(\theta_n) + \Delta_n\}$ Constant Step-size

$$\sum_{n=0}^{N-1} \Delta_n = \sum_{n=2}^{N+1} \mathcal{W}_n + \Delta \widehat{f}_N - \alpha \sum_{n=1}^{N} \Upsilon_n$$

$$\frac{1}{N} \sum_{n=0}^{N-1} \big(\theta_{n+1} - \theta_n\big) = \alpha \frac{1}{N} \sum_{n=0}^{N-1} \big(A^*[\theta_n - \theta^*] + \Delta_n + O(\alpha^2)\big)$$

Polyak-Ruppert Representation $\theta_N^{\mathsf{PR}} = \dfrac{1}{N} \displaystyle\sum_{n=0}^{N-1} \theta_n$

$$\frac{1}{\alpha} \frac{1}{N} \Delta\theta_N = A^*[\theta_N^{\mathsf{PR}} - \theta^*] + \frac{1}{N} \sum_{n=0}^{N-1} \Delta_n + O(\alpha^2)$$

Design Implications $\theta_{n+1} - \theta_n = \alpha\{\bar{f}(\theta_n) + \Delta_n\}$ Constant Step-size

Polyak-Ruppert Representation $\theta_N^{\mathsf{PR}} = \dfrac{1}{N}\displaystyle\sum_{n=0}^{N-1}\theta_n$

$$\frac{1}{\alpha}\frac{1}{N}\Delta\theta_N = A^*[\theta_N^{\mathsf{PR}} - \theta^*] + \frac{1}{N}\sum_{n=0}^{N-1}\Delta_n + O(\alpha^2)$$

P-R representation = the statistical optimal + statistical annoying:

$$\theta_N^{\mathsf{PR}} = \theta^* + \text{☺} - \text{☹} + O((\alpha N)^{-1} + \alpha^2)$$

$$\text{☺} = [A^*]^{-1}\frac{1}{N}\sum_{n=1}^{N}\mathcal{W}_n \qquad \text{☹} = [A^*]^{-1}\frac{\alpha}{N}\sum_{n=1}^{N}\Upsilon_n$$

Design Implications $\theta_{n+1} - \theta_n = \alpha\{\bar{f}(\theta_n) + \Delta_n\}$ Constant Step-size

Polyak-Ruppert Representation $\theta_N^{\mathsf{PR}} = \dfrac{1}{N}\displaystyle\sum_{n=0}^{N-1}\theta_n$

$$\frac{1}{\alpha}\frac{1}{N}\Delta\theta_N = A^*[\theta_N^{\mathsf{PR}} - \theta^*] + \frac{1}{N}\sum_{n=0}^{N-1}\Delta_n + O(\alpha^2)$$

P-R representation = the statistical optimal + statistical annoying:

$$\theta_N^{\mathsf{PR}} = \theta^* + \text{☺} - \text{☹} + O((\alpha N)^{-1} + \alpha^2)$$

$$\text{☺} = [A^*]^{-1}\frac{1}{N}\sum_{n=1}^{N}\mathcal{W}_n \qquad \text{☹} = [A^*]^{-1}\frac{\alpha}{N}\sum_{n=1}^{N}\Upsilon_n$$

Annoyance Υ_n: introduces bias of order $O(\alpha)$ (and variance not understood)

$$\Upsilon_n = -\tfrac{1}{\alpha}\left[\widehat{f}_{n+1}(\theta_n) - \widehat{f}_{n+1}(\theta_{n-1})\right] \approx -\partial_\theta \widehat{f}_{n+1}(\theta_n)\cdot f(\theta_{n-1}, W_n)$$

Design Implications

$\theta_{n+1} - \theta_n = \alpha_{n+1}\{\bar{f}(\theta_n) + \Delta_n\}$ Vanishing Step-size

Slightly different starting point:

$$\sum_{n=0}^{N-1} \frac{1}{\alpha_{n+1}}[\theta_{n+1} - \theta_n] = \sum_{n=0}^{N-1} \{\bar{f}(\theta_n) + \Delta_n\}$$

$$= \sum_{n=0}^{N-1} \{A^*[\theta_n - \theta^*] + \Delta_n\} + O\Big(\underbrace{\sum_{n=0}^{N-1} \alpha_{n+1}^2}_{\text{Assumed bounded}^*} \Big)$$

*Boundedness of sum relaxed in [79]

Design Implications

$\theta_{n+1} - \theta_n = \alpha_{n+1}\{\bar{f}(\theta_n) + \Delta_n\}$ Vanishing Step-size

Statistical memory $\Longrightarrow$ try $\alpha_n = g/(n_e + n)^\rho \quad \frac{1}{2} < \rho < 1$

Ignoring transients (and ignoring summation by parts calculation),

$$\theta_N^{\mathsf{PR}} = \frac{1}{N - N_0} \sum_{n=N_0}^{N-1} \theta_n$$

$$= \theta^* + \text{☺} + O(1/N)$$

$$\text{☺} = \frac{1}{N - N_0} [A^*]^{-1} \sum_{n=N_0+1}^{N} \mathcal{W}_n$$

Optimality of PR-Averaging

$$\mathsf{Cov}(\theta_N^{\mathsf{PR}} - \theta^*) = \frac{1}{N - N_0} \Sigma_N^{\mathsf{PR}}$$

$\Sigma_N^{\mathsf{PR}} \to \Sigma^{\mathsf{PR}} = [A^*]^{-1} \Sigma_{\mathcal{W}} [A^{*T}]^{-1}$ minimal in strongest sense

CS&RL In Three Parts

Part 1 Part 2 Part 3

Part 2: Control Systems & Reinforcement Learning

Sean Meyn

COGNITION & CONTROL IN COMPLEX SYSTEMS

Department of Electrical and Computer Engineering University of Florida

Inria International Chair Inria, Paris

Thanks to to our sponsors: NSF and ARO

Part 2: Q Learning Design

Outline

Return to Part 1 Skip to Part 3

7. Control Systems and Q Learning
8. Zap
9. Zap Zero
10. Conclusions & Future Directions

Resources

References

- Basic Control Concepts: CS&RL Chapters 3, 6 and 7.
- Temporal Difference Methods: CS&RL Chapters 5, 9, and 10.

Key recent papers:

- *The projected Bellman equation* [9]
- *Relative Q-learning* [68]
- Several on *Zap Q-learning* [31, 32]

A local or state back office, private operator, or traffic management center collects and processes data from the roads and vehicles
Backhaul fiber optic cables connect controllers to the back office, ensuring timely data processing
An in-vehicle red light violation warning alerts a driver who is about to run a red light
A roadside unit (RSU) transmits data to the vehicle
A traffic signal controller transfers information on the signal phase (green, yellow, red) and the amount of time remaining until the light changes to the RSU, which then broadcasts that data to the vehicle
On-board equipment receives data from the RSU radio and displays an appropriate alert to the driver

Q Learning

Mountain Car

Deterministic setting $\frac{d}{dt}x = F(x, u)$

Observations are position z and $v = \frac{d}{dt}z$

$$\frac{d^2}{dt^2}z = \frac{\kappa}{m}u - g\sin(\theta(z)), \quad |u| \leq 1$$

$\Longrightarrow$ State space model with state $x = (z; v)$.

Mountain Car

Deterministic setting $\frac{d}{dt}x = F(x,u)$

Observations are position z and $v = \frac{d}{dt}z$

$$\frac{d}{dt}z = v\,, \quad \frac{d}{dt}v = \frac{\kappa}{m}u - g\sin(\theta(z))\,, \quad |u| \le 1$$

$\Longrightarrow$ State space model with state $x = (z; v)$.

Goal: minimize time to reach the top

Minimum time is denoted $J^*(x_0)$

Mountain Car

Deterministic setting $\frac{d}{dt}x = F(x,u)$

Observations are position z and $v = \frac{d}{dt}z$

$$\frac{d}{dt}z = v\,, \quad \frac{d}{dt}v = \frac{\kappa}{m}u - g\sin(\theta(z))\,, \quad |u| \le 1$$

$\Longrightarrow$ State space model with state $x = (z; v)$.

Minimum time is denoted $J^*(x_0)$

Computation of the fixed-policy value function J^n for policy ϕ^n was performed using the numerical technique surveyed in Section 3.2.3. The fact that the computation was successful to compute J_0 implies that the policy ϕ_0 is stabilizing. Prop. 3.4 then implies that each of the polices $\{\phi^n\}$ obtained using PIA is also stabilizing.

Figure 3.3: Convergence of the two basic dynamic programming algorithms for MountainCar.

Recall the Bellman error defined in (3.34). The maximal absolute error at iteration n is

Mountain Car

Deterministic setting $\frac{d}{dt}x = F(x,u)$

Observations are position z and $v = \frac{d}{dt}z$

$$\frac{d}{dt}z = v\,, \quad \frac{d}{dt}v = \frac{\kappa}{m}u - g\sin(\theta(z))\,, \quad |u| \le 1$$

$\Longrightarrow$ State space model with state $x = (z; v)$.

Minimum time is denoted $J^*(x_0)$

Computation of the fixed-policy value function J^n for policy ϕ^n was performed using the numerical technique surveyed in Section 3.2.3. The fact that the computation was successful to compute J_0 implies that the policy ϕ_0 is stabilizing. Prop. 3.4 then implies that each of the polices $\{\phi^n\}$ obtained using PIA is also stabilizing.

Figure 3.3: Convergence of the two basic dynamic programming algorithms for MountainCar.

Recall the Bellman error defined in (3.34). The maximal absolute error at iteration n is

Controlled Lyapunov approach: (minimize $\frac{d}{dt}V(x_t)$)

$$\begin{aligned}\phi(x) &= \arg\min_{|u|\le 1} \nabla V(x)\cdot F(x,u) \\ &= \arg\min_{|u|\le 1} \left\{ V_z(x)v + V_v(x)\left[\frac{\kappa}{m}u - g\sin(\theta(z))\right]\right\}\end{aligned}$$

Mountain Car

Deterministic setting $\frac{d}{dt}x = F(x,u)$

Observations are position z and $v = \frac{d}{dt}z$

$$\frac{d}{dt}z = v\,, \quad \frac{d}{dt}v = \frac{\kappa}{m}u - g\sin(\theta(z))\,, \quad |u| \le 1$$

$\Longrightarrow$ State space model with state $x = (z; v)$.

Minimum time is denoted $J^*(x_0)$

Computation of the fixed-policy value function J^n for policy ϕ^n was performed using the numerical technique surveyed in Section 3.2.3. The fact that the computation was successful to compute J_0 implies that the policy ϕ_0 is stabilizing. Prop. 3.4 then implies that each of the polices $\{\phi^n\}$ obtained using PIA is also stabilizing.

$$\hat{B}_n = \max_x \left| -J_n(x) + \min_u [c(x,u) + J_n(F(x,u))] \right|$$

Figure 3.3: Convergence of the two basic dynamic programming algorithms for MountainCar.

Recall the Bellman error defined in (3.34). The maximal absolute error at iteration n is

Controlled Lyapunov approach:

$$\phi(x) = \arg\min_{|u|\le 1} \nabla V(x) \cdot F(x,u)$$

Motivation is the DP equation, using $c(x,u) \equiv 1$

$$0 = \min_{|u|\le 1}\{c(x,u) + \nabla J^*(x) \cdot F(x,u)\} \qquad \text{(HJB)}$$

Mountain Car

Deterministic setting $\frac{d}{dt}x = F(x,u)$

Observations are position z and $v = \frac{d}{dt}z$

$$\frac{d}{dt}z = v\,, \quad \frac{d}{dt}v = \frac{\kappa}{m}u - g\sin(\theta(z))\,, \quad |u| \le 1$$

$\Longrightarrow$ State space model with state $x = (z; v)$.

Minimum time is denoted $J^*(x_0)$

Computation of the fixed-policy value function J^n for policy ϕ^n was performed using the numerical technique surveyed in Section 3.2.3. The fact that the computation was successful to compute J_0 implies that the policy ϕ_0 is stabilizing. Prop. 3.4 then implies that each of the polices $\{\phi^n\}$ obtained using PIA is also stabilizing.

$$\bar{B}_n = \max_x \left| -J_n(x) + \min_u [c(x,u) + J_n(F(x,u))] \right|$$

Figure 3.3: Convergence of the two basic dynamic programming algorithms for MountainCar.

Recall the Bellman error defined in (3.34). The maximal absolute error at iteration n is

Controlled Lyapunov approach:

$$\phi(x) = \arg\min_{|u|\le 1} \nabla V(x) \cdot F(x,u)$$

Motivation is the DP equation, using $c(x,u) \equiv 1$

$$0 = \min_{|u|\le 1} \{c(x,u) + \nabla J^*(x) \cdot F(x,u)\} \qquad \text{(HJB)}$$

Q-learning: approximates $Q^*(x,u) = c(x,u) + \nabla J^*(x) \cdot F(x,u)$

Stochastic Optimal Control (Review)

MDP Model

$\boldsymbol{X}$ is a 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
- $\gamma < 1$ a discount factor

Assume states and inputs evolve on finite sets

Stochastic Optimal Control (Review)

MDP Model

$\boldsymbol{X}$ is a 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
- $\gamma < 1$ a discount factor

Q function:

$$Q^*(x,u) = \min_{\boldsymbol{U}} \sum_{n=0}^{\infty} \gamma^n \mathsf{E}[c(X_n, U_n) \mid X(0) = x, U(0) = u]$$

Stochastic Optimal Control (Review)

MDP Model

$\boldsymbol{X}$ is a 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
- $\gamma < 1$ a discount factor

Q function:

$$Q^*(x,u) = \min_{\boldsymbol{U}} \sum_{n=0}^{\infty} \gamma^n \mathsf{E}[c(X_n, U_n) \mid X(0) = x, U(0) = u]$$

Bellman equation

$$Q^*(x,u) = c(x,u) + \gamma \mathsf{E}\big[\min_{u'} Q^*(X_{n+1}, u') \mid X_n = x,\ U_n = u\big]$$

Q-Learning

Bellman equation

Find function Q^* that solves

$$c(x,u) + \gamma \mathsf{E}\big[\underline{Q}^*(X_{n+1}) \mid X_n = x,\ U_n = u\big] - Q^*(x,u) = 0$$

$$\underline{H}(x) = \min_u H(x,u)$$

Q-Learning

Bellman equation

Find function Q^* that solves

$$c(x,u) + \gamma \mathsf{E}\big[\underline{Q}^*(X_{n+1}) \mid X_n = x,\ U_n = u\big] - Q^*(x,u) = 0$$

Bellman equation

Find function Q^* that solves ($\mathcal{F}_n$ means history)

$$\mathsf{E}\big[c(X_n,U_n) + \gamma \underline{Q}^*(X_{n+1}) - Q^*(X_n,U_n) \mid \mathcal{F}_n\big] = 0$$

$$\underline{H}(x) = \min_u H(x,u)$$

Q-Learning solves Galerkin Relaxation

Bellman equation

Find function Q^* that solves ($\mathcal{F}_n$ means history)

$$\mathsf{E}\big[c(X_n,U_n)+\gamma \underline{Q}^*(X_{n+1})-Q^*(X_n,U_n)\mid \mathcal{F}_n\big]=0$$

Hidden Goal of Q-Learning

Given $\{Q^\theta:\theta\in\mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*)=0$,

$$\bar{f}(\theta)\stackrel{\text{def}}{=}\mathsf{E}\big[\mathcal{D}^\theta_{n+1}\zeta^\theta_n\big]$$

Temporal difference: $\mathcal{D}^\theta_{n+1}=c(X_n,U_n)+\gamma\underline{Q}^\theta(X_{n+1})-Q^\theta(X_n,U_n)$

Q-Learning

Bellman equation

Find function Q^* that solves ($\mathcal{F}_n$ means history)

$$\mathsf{E}\big[c(X_n, U_n) + \gamma \underline{Q}^*(X_{n+1}) - Q^*(X_n, U_n) \mid \mathcal{F}_n\big] = 0$$

Hidden Goal of Q-Learning

Given $\{Q^\theta : \theta \in \mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*) = 0$,

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^\theta_{n+1} \zeta^\theta_n\big]$$

Temporal difference: $\mathcal{D}^\theta_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)$

Eligibility vector: ζ^θ_n (d-dimensional function of past states and inputs)

Q-Learning

Bellman equation

Find function Q^* that solves ($\mathcal{F}_n$ means history)

$$\mathsf{E}\big[c(X_n, U_n) + \gamma \underline{Q}^*(X_{n+1}) - Q^*(X_n, U_n) \mid \mathcal{F}_n\big] = 0$$

Hidden Goal of Q-Learning

Given $\{Q^\theta : \theta \in \mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*) = 0$,

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^\theta_{n+1} \zeta^\theta_n\big]$$

Temporal difference: $\mathcal{D}^\theta_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)$

Eligibility vector: ζ^θ_n (d-dimensional function of past states and inputs)

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

Q-Learning

Hidden goal $\bar{f}(\theta^*) = 0$

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^\theta_{n+1}\zeta^\theta_n\big]\,, \qquad \mathcal{D}^\theta_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)$$

Typical choice $\zeta^\theta_n = \nabla_\theta Q^\theta(X_n, U_n)$ "Q(0)-learning"

$\Longrightarrow$ prototypical Q-learning algorithm

$$\underline{Q}^\theta_{n+1} = \min_u Q^\theta(X_{n+1}, u)$$

Q-Learning

Hidden goal $\bar{f}(\theta^*) = 0$

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^\theta_{n+1}\zeta^\theta_n\big]\,, \qquad \mathcal{D}^\theta_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)$$

Typical choice $\zeta^\theta_n = \nabla_\theta Q^\theta(X_n, U_n)$ "Q(0)-learning"

$\Longrightarrow$ prototypical Q-learning algorithm

Q Learning Algorithm

Estimates obtained using SA $\mathcal{D}_{n+1} = \mathcal{D}^\theta_{n+1}\big|_{\theta=\theta_n}$, $\zeta_n = \zeta^\theta_n\big|_{\theta=\theta_n}$

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f_{n+1}(\theta_n) \qquad f_{n+1}(\theta_n) = \mathcal{D}_{n+1}\zeta_n$$

$$\underline{Q}^\theta_{n+1} = \min_u Q^\theta(X_{n+1}, u)$$

Q-Learning

Hidden goal $\bar{f}(\theta^*) = 0$

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^\theta_{n+1}\zeta^\theta_n\big]\,, \qquad \mathcal{D}^\theta_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)$$

Typical choice $\zeta^\theta_n = \nabla_\theta Q^\theta(X_n, U_n)$ "Q(0)-learning"

$\Longrightarrow$ prototypical Q-learning algorithm

Q Learning Algorithm

Estimates obtained using SA $\mathcal{D}_{n+1} = \mathcal{D}^\theta_{n+1}\big|_{\theta=\theta_n}$, $\zeta_n = \zeta^\theta_n\big|_{\theta=\theta_n}$

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f_{n+1}(\theta_n) \qquad f_{n+1}(\theta_n) = \mathcal{D}_{n+1}\zeta_n$$

$$\underline{Q}^\theta_{n+1} = Q^\theta(X_{n+1}, \phi^\theta(X_{n+1}))$$

- $\phi^\theta(x) = \arg\min_u Q^\theta(x, u)$ [Q^θ-greedy policy]

Q-Learning

Hidden goal $\bar{f}(\theta^*) = 0$

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^\theta_{n+1}\zeta^\theta_n\big]\,, \qquad \mathcal{D}^\theta_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)$$

Typical choice $\zeta^\theta_n = \nabla_\theta Q^\theta(X_n, U_n)$ "Q(0)-learning"

$\Longrightarrow$ prototypical Q-learning algorithm

Q Learning Algorithm

Estimates obtained using SA $\qquad \mathcal{D}_{n+1} = \mathcal{D}^\theta_{n+1}\big|_{\theta=\theta_n}$, $\zeta_n = \zeta^\theta_n\big|_{\theta=\theta_n}$

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f_{n+1}(\theta_n) \qquad f_{n+1}(\theta_n) = \mathcal{D}_{n+1}\zeta_n$$

$$\underline{Q}^\theta_{n+1} = Q^\theta(X_{n+1}, \phi^\theta(X_{n+1}))$$

- $\phi^\theta(x) = \arg\min_u Q^\theta(x, u)$ $\qquad$ [Q^θ-greedy policy]
- Input $\{U_n\}$ chosen for *exploration*.

Q-Learning *Exploration*

$$\bar{f}(\theta) = \mathsf{E}\big[\big\{c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)\big\}\zeta_n\big]$$

Henceforth choose $\zeta_n = \nabla_\theta Q^\theta(X_n, U_n)\big|_{\theta=\theta_n}$

Q-Learning *Exploration*

$$\bar{f}(\theta) = \mathsf{E}\big[\big\{c(X_n, U_n) + \gamma \underline{Q}^{\theta}(X_{n+1}) - Q^{\theta}(X_n, U_n)\big\}\zeta_n\big]$$

Henceforth choose $\zeta_n = \nabla_\theta Q^\theta(X_n, U_n)\big|_{\theta=\theta_n}$

Q-learning

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n \qquad \mathcal{D}_{n+1} = c_n + \gamma \underline{Q}^{\theta_n}_{n+1} - Q^{\theta_n}_n$$

Is this stochastic approximation? Does it satisfy our assumptions?

Q-Learning *Exploration*

$$\bar{f}(\theta) = \mathsf{E}\big[\{c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)\}\zeta_n\big]$$

Henceforth choose $\zeta_n = \nabla_\theta Q^\theta(X_n, U_n)\big|_{\theta=\theta_n}$

Q-learning

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n \qquad \mathcal{D}_{n+1} = c_n + \gamma \underline{Q}^{\theta_n}_{n+1} - Q^{\theta_n}_n$$

Is this stochastic approximation? Does it satisfy our assumptions?
Assume randomized stationary policy:

$$\mathsf{P}\{U_n = u \mid \mathcal{F}_n; X_n = x\,, \theta_n = \theta\} = \breve{\phi}^\theta(u \mid x)$$

Called *oblivious* if $\breve{\phi}^\theta$ does not depend on θ

Q-Learning *Exploration*

$$\bar{f}(\theta) = \mathsf{E}\big[\{c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)\}\zeta_n\big]$$

Henceforth choose $\zeta_n = \nabla_\theta Q^\theta(X_n, U_n)\big|_{\theta=\theta_n}$

Q-learning

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n \qquad \mathcal{D}_{n+1} = c_n + \gamma \underline{Q}^{\theta_n}_{n+1} - Q^{\theta_n}_n$$

Is this stochastic approximation? Does it satisfy our assumptions?
Assume randomized stationary policy:

$$\mathsf{P}\{U_n = u \mid \mathcal{F}_n; X_n = x\,, \theta_n = \theta\} = \breve{\phi}^\theta(u \mid x)$$

Training policy is designed so that $\Phi_{n+1} = (X_n; U_n; X_{n+1})$ is ergodic

(for each fixed θ)

$$\bar{f}(\theta) = \mathsf{E}_{\pi_\theta}[f(\theta, \Phi_{n+1})]$$

Q-Learning *Exploration*

$$\bar{f}(\theta) = \mathsf{E}\big[\{c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)\}\zeta_n\big]$$

Henceforth choose $\zeta_n = \nabla_\theta Q^\theta(X_n, U_n)\big|_{\theta=\theta_n}$

Q-learning

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n \qquad \mathcal{D}_{n+1} = c_n + \gamma \underline{Q}^{\theta_n}_{n+1} - Q^{\theta_n}_n$$

Is this stochastic approximation? Does it satisfy our assumptions?
Assume randomized stationary policy:

$$\mathsf{P}\{U_n = u \mid \mathcal{F}_n; X_n = x\,, \theta_n = \theta\} = \breve{\phi}^\theta(u \mid x)$$

Training policy is designed so that $\Phi_{n+1} = (X_n; U_n; X_{n+1})$ is ergodic

$$\bar{f}(\theta) = \mathsf{E}_{\pi_\theta}[f(\theta, \Phi_{n+1})]$$

Is $\bar{f}$ Lipschitz continuous? Is the mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ GAS?

Q-Learning

Hidden goal $\bar{f}(\theta^*) = 0$

Q-learning with linear function approximation

Estimates obtained using SA

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f_{n+1}(\theta_n)$$

$$f_{n+1}(\theta_n) = \left\{ c_n + \gamma \underline{Q}^\theta_{n+1} - Q^\theta_n \right\}\Big|_{\theta=\theta_n} \zeta_n$$

$$\underline{Q}^\theta_{n+1} = Q^\theta(X_{n+1}), \phi^\theta(X_{n+1}))$$

- $Q^\theta(x,u) = \theta^\intercal \psi(x,u)$
- $\underline{Q}^\theta(x) = \theta^\intercal \psi(x, \phi^\theta(x))$
- $\zeta_n = \nabla_\theta Q^\theta(X_n, U_n)\big|_{\theta=\theta_n} = \psi(X_n, U_n)$

Q-Learning

Hidden goal $\bar{f}(\theta^*) = 0$

Q-learning with linear function approximation

Estimates obtained using SA

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f_{n+1}(\theta_n) \qquad f_{n+1}(\theta_n) = \left\{ c_n + \gamma \underline{Q}^\theta_{n+1} - Q^\theta_n \right\}\Big|_{\theta=\theta_n} \zeta_n$$

$$\underline{Q}^\theta_{n+1} = Q^\theta(X_{n+1}), \phi^\theta(X_{n+1}))$$

- $Q^\theta(x,u) = \theta^\intercal \psi(x,u)$
- $\underline{Q}^\theta(x) = \theta^\intercal \psi(x, \phi^\theta(x))$
- $\zeta_n = \nabla_\theta Q^\theta(X_n, U_n)\big|_{\theta=\theta_n} = \psi(X_n, U_n)$

$$\bar{f}(\theta) = A(\theta)\theta - b \qquad \bar{f}_\infty(\theta) = A(\theta)\theta$$

$$A(\theta) = \mathsf{E}\left[\zeta_n \left[\gamma \psi(X_{n+1}, \phi^\theta(X_{n+1})) - \psi(X_n, U_n)\right]^\intercal\right]$$

$$-b = \mathsf{E}\left[\zeta_n c(X_n, U_n)\right]$$

b independent of θ for oblivious training

Watkins' Q-learning

$$\mathsf{E}\big[\big\{c(X_n,U_n)+\gamma\underline{Q}^{\theta^*}(X_{n+1})-Q^{\theta^*}(X_n,U_n)\big\}\zeta_n\big]=0$$

Watkins' Q-learning

$$\mathsf{E}\big[\{c(X_n,U_n)+\gamma \underline{Q}^{\theta^*}(X_{n+1})-Q^{\theta^*}(X_n,U_n)\}\zeta_n\big]=0$$

Watkin's algorithm

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 $\equiv$ tabular setting)

Convergence of Q^{θ_n} to Q^* holds under mild conditions

Watkins' Q-learning

$$\mathsf{E}\big[\{c(X_n,U_n)+\gamma \underline{Q}^{\theta^*}(X_{n+1})-Q^{\theta^*}(X_n,U_n)\}\zeta_n\big]=0$$

Watkin's algorithm

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 $\equiv$ tabular setting)

Convergence of Q^{θ_n} to Q^* holds under mild conditions

Asymptotic covariance is infinite for $\gamma \geq 1/2$ Devraj's thesis [31, 66]

Using the standard step-size rule $\alpha_n = 1/n(x,u)$

Watkins' Q-learning

$$\mathsf{E}\big[\{c(X_n,U_n)+\gamma \underline{Q}^{\theta^*}(X_{n+1})-Q^{\theta^*}(X_n,U_n)\}\zeta_n\big]=0$$

Watkin's algorithm

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 $\equiv$ tabular setting)

Convergence of Q^{θ_n} to Q^* holds under mild conditions

Asymptotic covariance is infinite for $\gamma \geq 1/2$ Devraj's thesis [31, 66]

Using the standard step-size rule $\alpha_n = 1/n(x,u)$

Until recently, convergence theory only for oblivious policy

Crucial assumption: $\mathsf{E}_\pi[\psi_n\psi_n^T]>0$, meaning (X,U) is an irreducible Markov chain

Asymptotic Covariance of Watkins' Q-Learning

This is what infinite variance looks like

$$\sigma^2 = \lim_{n\to\infty} n\mathsf{E}[\|\theta_n - \theta^*\|^2] = \infty$$

Wild oscillations?

Asymptotic Covariance of Watkins' Q-Learning

This is what infinite variance looks like

$$\sigma^2 = \lim_{n\to\infty} n\mathsf{E}[\|\theta_n - \theta^*\|^2] = \infty$$

Wild oscillations?

Not at all, the sample paths appear frozen

Histogram of parameter estimates after 10^6 iterations.

Example from [31, 66] 2017

Asymptotic Covariance of Watkins' Q-Learning

This is what infinite variance looks like

$$\sigma^2 = \lim_{n\to\infty} n\mathsf{E}[\|\theta_n - \theta^*\|^2] = \infty$$

Wild oscillations?

Not at all, the sample paths appear frozen

Sample paths using a higher gain $[\alpha_n(x,u) = g/n(x,u)]$, or relative Q-learning.

Figure 1: Comparison of Q-learning and Relative Q-learning algorithms for the stochastic shortest path problem of [4]. The relative Q-learning algorithm is unaffected by large discounting.

Example from [31, 66] 2017, and [68], CS&RL, 2021

Numerical Instability Explained

The experiments used a diagonal matrix gain,

$$\theta_{n+1} = \theta_n + \frac{g}{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$G_n^{-1}(i,i)$: average number of times the pair (x^i, u^i) is visited up to time n (set to unity when this is zero).

Numerical Instability Explained

The experiments used a diagonal matrix gain,

$$\theta_{n+1} = \theta_n + \frac{g}{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$G_n^{-1}(i,i)$: average number of times the pair (x^i, u^i) is visited up to time n (set to unity when this is zero).

Mean flow dynamics:

$$\bar{f}(\theta) = A(\theta)\theta - b \qquad \bar{f}_\infty(\theta) = A(\theta)\theta$$

$A(\theta) = -[I - \gamma T_\theta]$ *for a family of transition matrices* $\{T_\theta\}$.

Numerical Instability Explained

The experiments used a diagonal matrix gain,

$$\theta_{n+1} = \theta_n + \frac{g}{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$G_n^{-1}(i,i)$: average number of times the pair (x^i, u^i) is visited up to time n (set to unity when this is zero).

Mean flow dynamics:

$$\bar{f}(\theta) = A(\theta)\theta - b \qquad \bar{f}_\infty(\theta) = A(\theta)\theta$$

$$A(\theta) = -[I - \gamma T_\theta] \quad \textit{for a family of transition matrices } \{T_\theta\}.$$

Challenge: $A(\theta)$ always has an eigenvalue at $-(1-\gamma)$.

Numerical Instability Explained

The experiments used a diagonal matrix gain,

$$\theta_{n+1} = \theta_n + \frac{g}{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$G_n^{-1}(i,i)$: average number of times the pair (x^i, u^i) is visited up to time n (set to unity when this is zero).

Mean flow dynamics:

$$\bar{f}(\theta) = A(\theta)\theta - b \qquad \bar{f}_\infty(\theta) = A(\theta)\theta$$

$$A(\theta) = -[I - \gamma T_\theta] \quad \textit{for a family of transition matrices } \{T_\theta\}.$$

Challenge: $A(\theta)$ always has an eigenvalue at $-(1-\gamma)$.

Resolution: Change your goal. Estimate the relative Q-function

$$H^*(x,u) = Q^*(x,u) - \langle \mu, Q^* \rangle$$

Numerical Instability Explained

The experiments used a diagonal matrix gain,

$$\theta_{n+1} = \theta_n + \frac{g}{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$G_n^{-1}(i,i)$: average number of times the pair (x^i, u^i) is visited up to time n (set to unity when this is zero).

Mean flow dynamics:

$$\bar{f}(\theta) = A(\theta)\theta - b \qquad \bar{f}_\infty(\theta) = A(\theta)\theta$$

$$A(\theta) = -[I - \gamma T_\theta] \quad \textit{for a family of transition matrices } \{T_\theta\}.$$

Challenge: $A(\theta)$ always has an eigenvalue at $-(1-\gamma)$.

Resolution: Change your goal. Or, estimate the advantage function

$$V^*(x,u) = Q^*(x,u) - h^*(x)$$

Advantage Q-Learning

Advantage: $V^*(x,u) = Q^*(x,u) - h^*(x)$

Optimal policy is unchanged: $\phi^*(x) = \arg\min_u V^*(x,u)$

Advantage Q-Learning

Advantage: $V^*(x,u) = Q^*(x,u) - h^*(x)$

Optimal policy is unchanged: $\phi^*(x) = \arg\min_u V^*(x,u)$

$$\underline{V}(x) \stackrel{\text{def}}{=} \min_u V^*(x,u) = 0$$

Advantage Q-Learning

Advantage: $V^*(x,u) = Q^*(x,u) - h^*(x)$

Assume the eligibility vectors satisfy $\mathsf{E}[\zeta_n \mid \mathcal{F}_{n-1}, X_n] = 0$

Then, $0 = \mathsf{E}[\zeta_n\{-V^*(X_n,U_n) + c(X_n,U_n) + \gamma \underline{V}^*(X_{n+1})\}]$

The SA algorithm is identical to Q-learning:

Advantage Q-Learning

Advantage: $V^*(x,u) = Q^*(x,u) - h^*(x)$

Assume the eligibility vectors satisfy $\mathsf{E}[\zeta_n \mid \mathcal{F}_{n-1}, X_n] = 0$

Then, $0 = \mathsf{E}[\zeta_n\{-V^*(X_n,U_n) + c(X_n,U_n) + \gamma \underline{V}^*(X_{n+1})\}]$

The SA algorithm is identical to Q-learning:

Advantage Q Learning Algorithm

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n$$

$$\mathcal{D}_{n+1} = c(X_n,U_n) + \gamma \underline{H}^{\theta_n}(X_{n+1}) - H^{\theta_n}(X_n,U_n)$$

Advantage Q-Learning

Advantage: $V^*(x,u) = Q^*(x,u) - h^*(x)$

Assume the eligibility vectors satisfy $\mathsf{E}[\zeta_n \mid \mathcal{F}_{n-1}, X_n] = 0$

Then, $0 = \mathsf{E}[\zeta_n\{-V^*(X_n,U_n) + c(X_n,U_n) + \gamma \underline{V}^*(X_{n+1})\}]$

The SA algorithm is identical to Q-learning:

Advantage Q Learning Algorithm

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n$$
$$\mathcal{D}_{n+1} = c(X_n,U_n) + \gamma \underline{H}^{\theta_n}(X_{n+1}) - H^{\theta_n}(X_n,U_n)$$

Baird's algorithm updates estimates of the two value functions separately

–I haven't found this simplification in the literature

Advantage Q-Learning

Advantage: $V^*(x,u) = Q^*(x,u) - h^*(x)$

Assume the eligibility vectors satisfy $\mathsf{E}[\zeta_n \mid \mathcal{F}_{n-1}, X_n] = 0$

Then, $0 = \mathsf{E}[\zeta_n\{-V^*(X_n,U_n) + c(X_n,U_n) + \gamma \underline{V}^*(X_{n+1})\}]$

The SA algorithm is identical to Q-learning:

Advantage Q Learning Algorithm

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n$$
$$\mathcal{D}_{n+1} = c(X_n,U_n) + \gamma \underline{H}^{\theta_n}(X_{n+1}) - H^{\theta_n}(X_n,U_n)$$

Baird's algorithm updates estimates of the two value functions separately

–I haven't found this simplification in the literature

The TD version of this algorithm appears in Section 9.5 of CS&RL

Basis to ensure success: $\psi^v(x,u) = \psi(x,u) - \psi_\phi(x)$.

Relative DP Equation

$$H^*(x,u) = Q^*(x,u) - \langle \mu, Q^* \rangle$$

Optimal policy is unchanged: $\phi^*(x) = \arg\min_u H^*(x,u)$

Relative DP Equation

$H^*(x,u) = Q^*(x,u) - \langle \mu, Q^* \rangle$

Choose $\delta > 0$ and a pmf (prob. mass function) $\nu \colon \mathsf{X} \times \mathsf{U} \to [0,1]$
Relative DP equation:

$$\mathsf{E}\big[c(X_n,U_n) + \gamma \underline{H}^*(X_{n+1}) - H^*(X_n,U_n) - \delta\langle \nu\,, H^*\rangle \mid \mathcal{F}_n\big] = 0$$

Relative DP Equation

$H^*(x,u) = Q^*(x,u) - \langle \mu, Q^* \rangle$

Choose $\delta > 0$ and a pmf (prob. mass function) $\nu \colon \mathsf{X} \times \mathsf{U} \to [0,1]$
Relative DP equation:

$$\mathsf{E}\big[c(X_n,U_n) + \gamma \underline{H}^*(X_{n+1}) - H^*(X_n,U_n) - \delta\langle \nu\,, H^*\rangle \mid \mathcal{F}_n\big] = 0$$

Solution: $\mu = \delta\nu/(1-\gamma)$

Relative DP Equation

$H^*(x,u) = Q^*(x,u) - \langle \mu, Q^* \rangle$

Choose $\delta > 0$ and a pmf (prob. mass function) $\nu \colon \mathsf{X} \times \mathsf{U} \to [0,1]$
Relative DP equation:

$$\mathsf{E}\big[c(X_n,U_n) + \gamma \underline{H}^*(X_{n+1}) - H^*(X_n,U_n) - \delta\langle \nu\,, H^*\rangle \mid \mathcal{F}_n\big] = 0$$

Relative Q Learning Algorithm

For a given function class $\{H^\theta\}$:

$$\theta_{n+1} = \theta_n + \alpha_{n+1}\mathcal{D}_{n+1}\zeta_n$$

Only requires a change in the temporal difference:

$$\mathcal{D}_{n+1} = c(X_n,U_n) + \gamma \underline{H}^{\theta_n}(X_{n+1}) - H^{\theta_n}(X_n,U_n) - \delta\langle \nu\,, H^{\theta_n}\rangle$$

Relative Q-Learning — Theory in Tabular Setting

Relative Q Learning Algorithm

$$\theta_{n+1} = \theta_n + \alpha_{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$$\mathcal{D}_{n+1} = c(X_n, U_n) + \gamma \underline{H}^{\theta_n}(X_{n+1}) - H^{\theta_n}(X_n, U_n) - \delta \langle \nu \, , H^{\theta_n} \rangle$$

Assume $\zeta_n = \psi_n = \psi(X_n, U_n)$

$G_n^{-1}(i,i)$: average number of times the pair (x^i, u^i) is visited up to time n:

$$G_n^{-1} = \frac{1}{n+1} \sum_{k=0}^{n} \zeta_k \zeta_k^T$$

($G_n^{-1}(i,i)$ set to unity when this is zero).

Relative Q-Learning — Theory in Tabular Setting

Relative Q Learning Algorithm

$$\theta_{n+1} = \theta_n + \alpha_{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$$\mathcal{D}_{n+1} = c(X_n, U_n) + \gamma \underline{H}^{\theta_n}(X_{n+1}) - H^{\theta_n}(X_n, U_n) - \delta \langle \boldsymbol{\nu}\,, H^{\theta_n} \rangle$$

Conclusions in the tabular setting [68]:

- $A^*_\delta = \partial_\theta \bar{f}\,(\theta^*) = -[I - \gamma P_{\phi^*} + \delta \cdot 1 \otimes \boldsymbol{\nu}] \qquad = A^* - \delta \cdot \bar{\psi}\, \bar{\psi}^T_{\boldsymbol{\nu}}$
- λ_1 for eigenvector 1 is $-(1 - \gamma + \delta)$

$$\Longrightarrow \quad \lambda_1 = -1 \text{ if } \delta = \gamma \text{ is used}$$

$$\bar{\psi} = \langle \pi, \psi \rangle,\ \bar{\psi}_{\boldsymbol{\nu}} = \langle \boldsymbol{\nu}, \psi \rangle$$

Relative Q-Learning — Theory in Tabular Setting

Relative Q Learning Algorithm

$$\theta_{n+1} = \theta_n + \alpha_{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$$\mathcal{D}_{n+1} = c(X_n, U_n) + \gamma \underline{H}^{\theta_n}(X_{n+1}) - H^{\theta_n}(X_n, U_n) - \delta \langle \boldsymbol{\nu}\,, H^{\theta_n} \rangle$$

Conclusions in the tabular setting [68]:

- $A^*_\delta = \partial_\theta \bar{f}\,(\theta^*) = -[I - \gamma P_{\phi^*} + \delta \cdot 1 \otimes \boldsymbol{\nu}] \qquad = A^* - \delta \cdot \bar{\psi}\,\bar{\psi}^T_{\boldsymbol{\nu}}$
- λ_1 for eigenvector 1 is $-(1 - \gamma + \delta)$

$$\implies \quad \lambda_1 = -1 \text{ if } \delta = \gamma \text{ is used}$$

- All other eigenvalues satisfy $\mathrm{Re}(\lambda(A^*_\delta)) \le -(1 - \gamma\rho^*)$,

$$\rho^* = \max\{\mathrm{Re}(\lambda(P_{\phi^*})) : \lambda \ne \lambda_1\}$$

Relative Q-Learning — Theory in Tabular Setting

Relative Q Learning Algorithm

$$\theta_{n+1} = \theta_n + \alpha_{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$$\mathcal{D}_{n+1} = c(X_n, U_n) + \gamma \underline{H}^{\theta_n}(X_{n+1}) - H^{\theta_n}(X_n, U_n) - \delta \langle \boldsymbol{\nu}\,, H^{\theta_n} \rangle$$

Conclusions in the tabular setting [68]:

- $A^*_\delta = \partial_\theta \bar{f}\,(\theta^*) = -[I - \gamma P_{\phi^*} + \delta \cdot 1 \otimes \boldsymbol{\nu}] \qquad = A^* - \delta \cdot \bar{\psi}\,\bar{\psi}^T_{\boldsymbol{\nu}}$
- λ_1 for eigenvector 1 is $-(1 - \gamma + \delta)$

  $\implies \quad \lambda_1 = -1$ if $\delta = \gamma$ is used
- All other eigenvalues satisfy $\mathrm{Re}(\lambda(A^*_\delta)) \le -(1 - \gamma\rho^*)$,

$$\rho^* = \max\{\mathrm{Re}(\lambda(P_{\phi^*})) : \lambda \ne \lambda_1\}$$

- Finite asymptotic variance with $\alpha_n = n^{-1}(1 - \rho^*\gamma)^{-1}$

Relative Q-Learning — Theory in Tabular Setting

Relative Q Learning Algorithm

$$\theta_{n+1} = \theta_n + \alpha_{n+1} G_{n+1} \mathcal{D}_{n+1} \zeta_n$$

$$\mathcal{D}_{n+1} = c(X_n, U_n) + \gamma \underline{H}^{\theta_n}(X_{n+1}) - H^{\theta_n}(X_n, U_n) - \delta \langle \boldsymbol{\nu}\,, H^{\theta_n} \rangle$$

Conclusions in the tabular setting [68]:

- $A^*_\delta = \partial_\theta \bar{f}\,(\theta^*) = -[I - \gamma P_{\phi^*} + \delta \cdot 1 \otimes \boldsymbol{\nu}] \qquad = A^* - \delta \cdot \bar{\psi}\,\bar{\psi}^T_{\boldsymbol{\nu}}$
- λ_1 for eigenvector 1 is $-(1 - \gamma + \delta)$
- All other eigenvalues satisfy $\mathrm{Re}(\lambda(A^*_\delta)) \le -(1 - \gamma\rho^*)$
- Finite asymptotic variance with $\alpha_n = n^{-1}(1 - \rho^*\gamma)^{-1}$

Do not forget the rules!

- $\alpha_n = g n^{-\rho}$ with $g > 0$ and $\frac{1}{2} < \rho < 1$
- $\theta^{\mathrm{PR}}_N = \dfrac{1}{N - N_0} \displaystyle\sum_{n=N_0+1}^{N} \theta_n$ *This will give far better performance*

New in 2023: Exploration can stabilize Q-learning

Approaches to exploration, $U_k \sim \breve{\phi}_k(\,\cdot \mid X_k)$:

- ε-greedy, $U_k = \phi^{\theta_k}(X_k)$ probability $1-\varepsilon$ — small $\varepsilon > 0$
- Gibbs, $\breve{\phi}_k(u \mid x) = \dfrac{1}{\mathcal{Z}} \exp\bigl(-\kappa Q^{\theta_k}(x,u)\bigr)$ — large $\kappa > 0$

  Lipschitz fails (and more)

New in 2023: Exploration can stabilize Q-learning

Approaches to exploration, $U_k \sim \breve{\phi}_k(\,\cdot \mid X_k)$:

- ε-greedy, $U_k = \phi^{\theta_k}(X_k)$ probability $1-\varepsilon$ — small $\varepsilon > 0$
- Gibbs, $\breve{\phi}_k(u \mid x) = \dfrac{1}{\mathcal{Z}} \exp\big(-\kappa Q^{\theta_k}(x,u)\big)$ — large $\kappa > 0$
- Tamed Gibbs, $\breve{\phi}_0^{\theta}(u \mid x) = \dfrac{1}{\mathcal{Z}_\kappa^{\theta}(x)} \exp\big(-\kappa_\theta Q^{\theta}(x,u)\big)$ — New in 2023

New in 2023: Exploration can stabilize Q-learning

Approaches to exploration, $U_k \sim \breve{\phi}_k(\,\cdot \mid X_k)$:

- ε-greedy, $U_k = \phi^{\theta_k}(X_k)$ probability $1-\varepsilon$ — small $\varepsilon > 0$
- Gibbs, $\breve{\phi}_k(u \mid x) = \dfrac{1}{\mathcal{Z}} \exp\big(-\kappa Q^{\theta_k}(x,u)\big)$ — large $\kappa > 0$
- Tamed Gibbs, $\breve{\phi}_0^{\theta}(u \mid x) = \dfrac{1}{\mathcal{Z}_\kappa^\theta(x)} \exp\big(-\kappa_\theta Q^\theta(x,u)\big)$ — large $\kappa_0 > 0$

$$\kappa_\theta \begin{cases} = \frac{1}{\|\theta\|}\kappa_0 & \|\theta\| \geq 1 \\ \geq \frac{1}{2}\kappa_0 & \textit{else} \end{cases}$$

SA recursion satisfies all the assumptions — New in 2023

Theory in [9] for linear function approximation,
but extends to Q^θ Lipschitz in θ (e.g. neural networks with ReLU activation)

Theory for Tamed Gibbs $\breve{\phi}_k(u \mid x) \stackrel{\text{def}}{=} \mathsf{P}\{U_k = u \mid \mathcal{F}_k \,; X_k = x\}$

For ease of analysis: $\breve{\phi}_k(u \mid x) = \breve{\phi}^{\theta_k}(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$

Theory for Tamed Gibbs $\breve{\phi}_k(u \mid x) \stackrel{\text{def}}{=} \mathsf{P}\{U_k = u \mid \mathcal{F}_k\,; X_k = x\}$

For ease of analysis: $\breve{\phi}_k(u \mid x) = \breve{\phi}^{\theta_k}(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$

Assumptions: $Q^\theta(x,u) = \theta^\intercal \psi(x,u)$, and

Theory for Tamed Gibbs $\breve{\phi}_k(u \mid x) \stackrel{\text{def}}{=} \mathsf{P}\{U_k = u \mid \mathcal{F}_k\,; X_k = x\}$

For ease of analysis: $\breve{\phi}_k(u \mid x) = \breve{\phi}^{\theta_k}(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$

Assumptions: $Q^\theta(x,u) = \theta^\intercal \psi(x,u)$, and

For oblivious policy ($\varepsilon = 1$):

1. There is a unique invariant pmf $\pi_{\mathcal{W}}$ for $(\boldsymbol{X}, \boldsymbol{U})$.
2. The covariance is full rank, $R^{\mathcal{W}} > 0$,

$$R^{\mathcal{W}} = \mathsf{E}_{\pi_{\mathcal{W}}}\left[\psi(X_n, U_n)\psi(X_n, U_n)^\intercal\right], \qquad U_n = \mathcal{W}_n \sim \nu_{\mathcal{W}}$$

Theory for Tamed Gibbs $\breve{\phi}_k(u \mid x) \stackrel{\text{def}}{=} \mathsf{P}\{U_k = u \mid \mathcal{F}_k\,; X_k = x\}$

For ease of analysis: $\breve{\phi}_k(u \mid x) = \breve{\phi}^{\theta_k}(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$

Assumptions: $Q^\theta(x,u) = \theta^\intercal \psi(x,u)$, and

For oblivious policy ($\varepsilon = 1$):

1. There is a unique invariant pmf $\pi_{\mathcal{W}}$ for $(\boldsymbol{X}, \boldsymbol{U})$.
2. The covariance is full rank,

$$R^{\mathcal{W}} = \mathsf{E}_{\pi_{\mathcal{W}}}\left[\psi(X_n,U_n)\psi(X_n,U_n)^\intercal\right], \qquad U_n = \mathcal{W}_n \sim \nu_{\mathcal{W}}$$

First step in analysis is to show that 1 and 2 hold for any $\varepsilon > 0$:

Theory for Tamed Gibbs $\breve{\phi}_k(u \mid x) \stackrel{\text{def}}{=} \mathsf{P}\{U_k = u \mid \mathcal{F}_k \,; X_k = x\}$

For ease of analysis: $\breve{\phi}_k(u \mid x) = \breve{\phi}^{\theta_k}(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$

Assumptions: $Q^\theta(x,u) = \theta^\intercal \psi(x,u)$, and

For oblivious policy ($\varepsilon = 1$):

1. There is a unique invariant pmf $\pi_{\mathcal{W}}$ for $(\boldsymbol{X}, \boldsymbol{U})$.
2. The covariance is full rank,

$$R^{\mathcal{W}} = \mathsf{E}_{\pi_{\mathcal{W}}}\left[\psi(X_n, U_n)\psi(X_n, U_n)^\intercal\right], \qquad U_n = \mathcal{W}_n \sim \nu_{\mathcal{W}}$$

First step in analysis is to show that 1 and 2 hold for any $\varepsilon > 0$:

- For each θ is a unique invariant pmf π_θ for $(\boldsymbol{X}, \boldsymbol{U})$
  (with $U_n \sim \breve{\phi}^\theta(u \mid X_n)$ for each n).
- The covariance is full rank,

$$R^{\Theta}(\theta) = \mathsf{E}_{\pi_\theta}\left[\psi(X_n, U_n)\psi(X_n, U_n)^\intercal\right]$$

Theory

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\{c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)\}\zeta_n\big]$$

Stability with sufficient optimism.

There is $\varepsilon_\gamma > 0$ (lower bound given in paper) for which the following hold:

Theory

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\{c(X_n,U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n,U_n)\}\zeta_n\big]$$

Stability with sufficient optimism.

There is $\varepsilon_\gamma > 0$ (lower bound given in paper) for which the following hold:

For each $0 < \varepsilon < \varepsilon_\gamma$, there is $\kappa_{\varepsilon,\gamma}$ such that

- The mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ is *ultimately bounded.*

Proof follows Van Roy's analysis of TD-learning,

$$\frac{d}{dt}\|\vartheta_t\| \le -\delta\|\vartheta_t\|\,, \qquad \textit{if} \;\; \|\vartheta_t\| \ge 1/\delta$$

Theory

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\{c(X_n,U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n,U_n)\}\zeta_n\big]$$

Stability with sufficient optimism.

There is $\varepsilon_\gamma > 0$ (lower bound given in paper) for which the following hold:

For each $0 < \varepsilon < \varepsilon_\gamma$, there is $\kappa_{\varepsilon,\gamma}$ such that

- The mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ is *ultimately bounded*.
- There is at least one solution to the projected Bellman equation

$$\bar{f}(\theta^*) = 0$$

Proof follows from the stability proof + Brouwer's fixed-point theorem

Theory

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\{c(X_n,U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n,U_n)\}\zeta_n\big]$$

Stability with sufficient optimism.

There is $\varepsilon_\gamma > 0$ (lower bound given in paper) for which the following hold:

For each $0 < \varepsilon < \varepsilon_\gamma$, there is $\kappa_{\varepsilon,\gamma}$ such that

- The mean flow $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ is *ultimately bounded*.
- There is at least one solution to the projected Bellman equation

$$\bar{f}(\theta^*) = 0$$

- Under some additional assumptions θ^* is *locally* asymptotically stable

Baird's Example

$$\breve{\phi}_k(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

Baird's Example

$$\breve{\phi}_k(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

From CS&RL

Results for TD(λ)-learning, $\varepsilon = 1$

Baird's Example

$$\breve{\phi}_k(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

Results for Q-learning

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

Baird's Example

$$\breve{\phi}_k(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

Fig. 1. Evolution of the Q-function approximations for two values of discount factor, and using an ε-greedy policy with common value of $\varepsilon = 0.5$.

Baird's Example

$$\breve{\phi}_k(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

Fig. 1. Evolution of the Q-function approximations for two values of discount factor, and using an ε-greedy policy with common value of $\varepsilon = 0.5$.

Fig. 2. Evolution of the Q-function approximations when using an ε-greedy policy. Convergence holds when $\varepsilon > 0$ is sufficiently small.

Baird's Example

$$\breve{\phi}_k(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

If you want to experiment with relative Q learning

$\lambda(A^*)$

Focus only on eigenvalues near imaginary axis

The basis is dimension 14:

$$\psi_i(x,u) = \begin{cases} 2\mathbb{I}\{x=i, u=0\} & 1 \le i \le 6 \\ \mathbb{I}\{x=7, u=0\} + \mathbb{I}\{x=1, u=1\} & i=7 \\ \mathbb{I}\{x+6=i, u=1\} & 8 \le i \le 13 \\ \mathbb{I}\{u=0\}[1+\mathbb{I}\{x=7\}] & i=14 \end{cases}$$

With $\delta = 0$ we see an eigenvalue very close to the origin in $\mathbb{C}$

Baird's Example

$$\breve{\phi}_k(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

If you want to experiment with relative Q learning

$\lambda(A_\delta^*)$

$A_\delta^* = A^* - \delta\bar{\psi} \otimes \nu$

$\delta = 0.0$

$\delta = 0.2$

$\nu = \delta_{7,0}$

$\bar{\psi}_\nu = 2e^{14} + e^7$

The basis is dimension 14:

$$\psi_i(x,u) = \begin{cases} 2\mathbb{I}\{x=i, u=0\} & 1 \le i \le 6 \\ \mathbb{I}\{x=7, u=0\} + \mathbb{I}\{x=1, u=1\} & i = 7 \\ \mathbb{I}\{x+6=i, u=1\} & 8 \le i \le 13 \\ \mathbb{I}\{u=0\}[1+\mathbb{I}\{x=7\}] & i = 14 \end{cases}$$

With $\nu = \delta_{7,0}$ the dominant eigenvalue is unchanged

Baird's Example

$$\breve{\phi}_k(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

If you want to experiment with relative Q learning

$\lambda(A_\delta^*)$

$A_\delta^* = A^* - \delta\bar{\psi} \otimes \nu$

$\delta = 0.2$

× $\delta = 0.0$

× $\nu = \delta_{7,0}$, $\bar{\psi}_\nu = 2e^{14} + e^7$

× $\nu = \delta_{5,1}$, $\bar{\psi}_\nu = e^{11}$

The basis is dimension 14:

$$\psi_i(x,u) = \begin{cases} 2\mathbb{I}\{x=i, u=0\} & 1 \le i \le 6 \\ \mathbb{I}\{x=7, u=0\} + \mathbb{I}\{x=1, u=1\} & i = 7 \\ \mathbb{I}\{x+6=i, u=1\} & 8 \le i \le 13 \\ \mathbb{I}\{u=0\}[1+\mathbb{I}\{x=7\}] & i = 14 \end{cases}$$

With $\nu = \delta_{5,1}$ the eigenvalue moves away from the origin

Baird's Example

$$\breve{\phi}_k(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta_k}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

If you want to experiment with relative Q learning

The basis is dimension 14:

$$\psi_i(x,u) = \begin{cases} 2\mathbb{I}\{x=i, u=0\} & 1 \le i \le 6 \\ \mathbb{I}\{x=7, u=0\} + \mathbb{I}\{x=1, u=1\} & i = 7 \\ \mathbb{I}\{x+6=i, u=1\} & 8 \le i \le 13 \\ \mathbb{I}\{u=0\}[1+\mathbb{I}\{x=7\}] & i = 14 \end{cases}$$

Conclusion: theory is still lacking outside of the tabular setting. While this is sorting itself out, you might try Q-learning for the *advantage function*.

Zap

ODE Design

The ODE method begins with design of the ODE: $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$
Challenges we have faced with Q-learning:

ODE Design

The ODE method begins with design of the ODE: $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$
Challenges we have faced with Q-learning:

- How can we design dynamics for
  1. Stability few results outside of Watkins' tabular setting
  2. $\bar{f}(\theta^*) = 0$ solves a relevant problem or has a solution

ODE Design

The ODE method begins with design of the ODE: $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$
Challenges we have faced with Q-learning:

- How can we design dynamics for
  1. Stability
  2. $\bar{f}(\theta^*) = 0$ solves a relevant problem
- How can we better manage problems introduced by $1/(1-\gamma)$?

Relative Q-Learning is one approach

ODE Design

The ODE method begins with design of the ODE: $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$
Challenges we have faced with Q-learning:

- How can we design dynamics for
  1. Stability
  2. $\bar{f}(\theta^*) = 0$ solves a relevant problem
- How can we better manage problems introduced by $1/(1-\gamma)$?

Relative Q-Learning is one approach

Assuming we have solved 2,
approximate Newton-Raphson flow:

$$\frac{d}{dt}\bar{f}(\vartheta_t) = -\bar{f}(\vartheta_t) \qquad \textit{giving} \quad \bar{f}(\vartheta_t) = \bar{f}(\vartheta_0)e^{-t}$$

ODE Design

The ODE method begins with design of the ODE: $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$
Challenges we have faced with Q-learning:

- How can we design dynamics for
  1. Stability
  2. $\bar{f}(\theta^*) = 0$ solves a relevant problem
- How can we better manage problems introduced by $1/(1-\gamma)$?

Relative Q-Learning is one approach

Assuming we have solved 2,
approximate Newton-Raphson flow:

$$\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t) \qquad \textit{giving} \quad \bar{f}(\vartheta_t) = \bar{f}(\vartheta_0)e^{-t}$$

Zap Algorithm

Designed to emulate Newton-Raphson flow

$$\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t), \quad A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$$

Zap-SA

Zap Algorithm

Designed to emulate Newton-Raphson flow

$\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t), \quad A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$

Zap-SA

$$\theta_{n+1} = \theta_n + \alpha_{n+1} G_{n+1} f_{n+1}(\theta_n) \qquad G_{n+1} = [-\widehat{A}_{n+1}]^{-1}$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \beta_{n+1}(A_{n+1} - \widehat{A}_n) \qquad A_{n+1} = \partial_\theta f_{n+1}(\theta_n)$$

$\widehat{A}_{n+1} \approx A(\theta_n)$ requires high-gain, $\dfrac{\beta_n}{\alpha_n} \to \infty, \qquad n \to \infty$

Zap Algorithm

Designed to emulate Newton-Raphson flow

$\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t), \quad A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$

Zap-SA

$$\theta_{n+1} = \theta_n + \alpha_{n+1} G_{n+1} f_{n+1}(\theta_n) \qquad G_{n+1} = [-\widehat{A}_{n+1}]^{-1}$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \beta_{n+1}(A_{n+1} - \widehat{A}_n) \qquad A_{n+1} = \partial_\theta f_{n+1}(\theta_n)$$

$\widehat{A}_{n+1} \approx A(\theta_n)$ requires high-gain, $\dfrac{\beta_n}{\alpha_n} \to \infty, \qquad n \to \infty$

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

Zap Q-Learning: $f_{n+1}(\theta_n) = \{c(X_n, U_n) + \gamma \underline{Q}^{\theta_n}(X_{n+1}) - Q^{\theta_n}(X_n, U_n)\}\zeta_n$

With linear function approx.: $\zeta_n = \psi(X_n, U_n)$

$$A_{n+1} = \zeta_n \left[\gamma\psi(X_{n+1}, \phi^\theta(X_{n+1})) - \psi(X_n, U_n)\right]^T$$

$$\phi^\theta(x) = \arg\min_u Q^\theta(x, u)$$

Challenges

Q-learning: $\{Q^\theta(x,u) : \theta \in \mathbb{R}^d\,,\ u \in \mathsf{U}\,,\ x \in \mathsf{X}\}$

Find θ^* such that $\bar{f}(\theta^*) = 0$, with

$$\bar{f}(\theta) = \mathsf{E}\big[\{c(X_n,U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n,U_n)\}\zeta_n\big]$$

What makes theory difficult:

1. Does $\bar{f}$ have a root?
2. Does the inverse of A exist?
3. SA theory is weak for a discontinuous ODE $\Longleftarrow$ big technical challenge

Challenges

Q-learning: $\{Q^\theta(x,u) : \theta \in \mathbb{R}^d\,,\ u \in \mathsf{U}\,,\ x \in \mathsf{X}\}$

Find θ^* such that $\bar{f}(\theta^*) = 0$, with

$$\bar{f}(\theta) = \mathsf{E}\big[\{c(X_n,U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n,U_n)\}\zeta_n\big]$$

What makes theory difficult:

1. Does $\bar{f}$ have a root?
2. Does the inverse of A exist?
3. SA theory is weak for a discontinuous ODE $\Longleftarrow$ big technical challenge

$\Longrightarrow$ 3 is resolved by exploiting special structure,
even for NN function approximation [32, 8]

Zap Q-Learning

Optimize Walk to Cafe

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

Convergence of Zap-Q Learning

Discount factor: $\gamma = 0.99$

Zap Q-Learning

Optimize Walk to Cafe

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

Watkins: Infinite covariance with $\alpha_n = 1/n$ or $1/n(x,u)$.

Convergence of Zap-Q Learning

Discount factor: $\gamma = 0.99$

Zap Q-Learning

Optimize Walk to Cafe

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

Watkins: Infinite covariance with $\alpha_n = 1/n$ or $1/n(x,u)$.

Convergence of Zap-Q Learning

Discount factor: $\gamma = 0.99$

Zap Q-Learning

Optimize Walk to Cafe

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

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

Entry #18: $n = 10^4$ $n = 10^6$

Entry #10: $n = 10^4$ $n = 10^6$

CLT gives good prediction of finite-n performance

Discount factor: $\gamma = 0.99$

Zap Q-Learning

Optimize Walk to Cafe

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

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

Entry #18: $n = 10^4$ $n = 10^6$ Entry #10: $n = 10^4$ $n = 10^6$

CLT gives good prediction of finite-n performance

Discount factor: $\gamma = 0.99$

Recall the Rules:

Multiple runs required to estimate CLT variance

Requires **wide** range of initial conditions, but a smaller run length as you fine-tune your algorithm

Zap with Neural Networks

$$0 = \bar{f}(\theta^*) = \mathsf{E}\big[\big\{c(X_n, U_n) + \gamma \underline{Q}^{\theta^*}(X_{n+1}) - Q^{\theta^*}(X_n, U_n)\big\}\zeta_n\big]$$

$\zeta_n = \nabla_\theta Q^\theta(X_n, U_n)\big|_{\theta=\theta_n}$ computed using back-progagation

A few things to note:

- As far as we know, the empirical success of plain vanilla DQN is *extraordinary* (however, nobody reports failure)

Zap with Neural Networks

$$0 = \bar{f}(\theta^*) = \mathsf{E}\big[\big\{c(X_n, U_n) + \gamma \underline{Q}^{\theta^*}(X_{n+1}) - Q^{\theta^*}(X_n, U_n)\big\}\zeta_n\big]$$

$\zeta_n = \nabla_\theta Q^\theta(X_n, U_n)\big|_{\theta=\theta_n}$ computed using back-progagation

A few things to note:

- As far as we know, the empirical success of plain vanilla DQN is *extraordinary* (however, nobody reports failure)
- Zap Q-learning is the only approach for which convergence has been established (under mild conditions)
- We can expect stunning transient performance, based on coupling with the Newton-Raphson flow.

Zap with Neural Networks

VI. Stunning reliability with Q^θ parameterized by various neural networks

Reliability and stunning transient performance

—from coupling with the Newton-Raphson flow.

Zap Zero

Something new

Zap Zero Circa 2021

Newton-Raphson flow: $\dfrac{d}{dt}\vartheta_t = w_t \qquad w_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

Zap Zero Circa 2021

Newton-Raphson flow: $\dfrac{d}{dt}\vartheta_t = w_t$

$$w_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$$

$$0 = A(\vartheta_t)w_t + \bar{f}(\vartheta_t)$$

Zap Zero Circa 2021

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

Newton-Raphson flow: $\dfrac{d}{dt}\vartheta_t = w_t \qquad w_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$0 = A(\vartheta_t)w_t + \bar{f}(\vartheta_t)$$

Zap Zero ODE:

$$\frac{d}{dt}\vartheta_t = w_t$$

$$\frac{d}{dt}w_t = \beta_t[A(\vartheta_t)w_t + \bar{f}(\vartheta_t)]$$

Inspired by GQ learning algorithm [29]

Zap Zero Circa 2021

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

Newton-Raphson flow: $\dfrac{d}{dt}\vartheta_t = w_t \qquad w_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$0 = A(\vartheta_t)w_t + \bar{f}(\vartheta_t)$$

Zap Zero ODE:

$$\frac{d}{dt}\vartheta_t = w_t$$

$$\frac{d}{dt}w_t = \beta_t[A(\vartheta_t)w_t + \bar{f}(\vartheta_t)] \qquad \beta_t \uparrow \infty$$

Inspired by GQ learning algorithm [29]

Zap Zero Circa 2021

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

Newton-Raphson flow: $\dfrac{d}{dt}\vartheta_t = w_t \qquad w_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$0 = A(\vartheta_t)w_t + \bar{f}(\vartheta_t)$$

Zap Zero ODE:

$$\frac{d}{dt}\vartheta_t = w_t$$

$$\frac{d}{dt}w_t = \beta_t[A(\vartheta_t)w_t + \bar{f}(\vartheta_t)] \qquad \beta_t \uparrow \infty$$

Convergent subject to uniform Hurwitz condition for $A(\theta)$

Zap Zero Circa 2021

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

Newton-Raphson flow: $\dfrac{d}{dt}\vartheta_t = w_t \qquad w_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$0 = A(\vartheta_t)w_t + \bar{f}(\vartheta_t)$$

Zap Zero ODE:

$$\frac{d}{dt}\vartheta_t = w_t$$

$$\frac{d}{dt}w_t = \beta_t[A(\vartheta_t)w_t + \bar{f}(\vartheta_t)] \qquad \beta_t \uparrow \infty$$

See CS&RL [8] for details and caveats for this experiment

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

Uniform Hurwitz condition for $A(\theta)$ is horrible! Not needed for Zap

Let's try harder ...

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

NR flow: $\dfrac{d}{dt}\vartheta_t = -[\vartheta_t + z_t]$ $\qquad \vartheta_t + z_t = [A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

NR flow: $$\frac{d}{dt}\vartheta_t = -[\vartheta_t + z_t] \qquad \vartheta_t + z_t = [A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$$

$$\bar{f}(\vartheta_t) = A(\vartheta_t)[\vartheta_t + z_t]$$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

NR flow: $$\frac{d}{dt}\vartheta_t = -[\vartheta_t + z_t] \qquad \vartheta_t + z_t = [A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$$

$$\bar{f}(\vartheta_t) = A(\vartheta_t)[\vartheta_t + z_t]$$

Variables w_t, z_t and matrix gain L_t designed so $z_t \approx L_t w_t$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

NR flow: $\dfrac{d}{dt}\vartheta_t = -[\vartheta_t + z_t]$ $\qquad \vartheta_t + z_t = [A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$\bar{f}(\vartheta_t) = A(\vartheta_t)[\vartheta_t + z_t]$$

Variables w_t, z_t and matrix gain L_t designed so $z_t \approx L_t w_t$

$$\frac{d}{dt}\vartheta_t = -[\vartheta_t + L_t w_t]$$

$$\frac{d}{dt}w_t =$$

$$\frac{d}{dt}z_t = -\beta_t\big[z_t - L_t w_t\big] \qquad \Leftrightarrow z_t \approx L_t w_t \quad \textit{If stable}$$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

NR flow: $\dfrac{d}{dt}\vartheta_t = -[\vartheta_t + z_t]$

$$\vartheta_t + z_t = [A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$$

$$\bar{f}(\vartheta_t) = A(\vartheta_t)[\vartheta_t + z_t]$$

Variables w_t, z_t and matrix gain L_t designed so $z_t \approx L_t w_t$

$$\frac{d}{dt}\vartheta_t = -[\vartheta_t + L_t w_t] \qquad \Leftrightarrow \vartheta_t + z_t \approx 0$$

$$\frac{d}{dt}w_t =$$

$$\frac{d}{dt}z_t = -\beta_t\big[z_t - L_t w_t\big] \qquad \Leftrightarrow z_t \approx L_t w_t \quad \textit{If stable}$$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

NR flow: $\dfrac{d}{dt}\vartheta_t = -[\vartheta_t + z_t]$ $\qquad \vartheta_t + z_t = [A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$\bar{f}(\vartheta_t) = A(\vartheta_t)[\vartheta_t + z_t]$$

Variables w_t, z_t and matrix gain L_t designed so $z_t \approx L_t w_t$

$$\frac{d}{dt}\vartheta_t = -[\vartheta_t + L_t w_t] \qquad \Leftrightarrow \vartheta_t + z_t \approx 0$$

$$\frac{d}{dt}w_t = -\beta_t\big[A(\vartheta_t)[\vartheta_t + z_t] - \bar{f}(\vartheta_t)\big] \qquad \Leftrightarrow \bar{f}(\vartheta_t) \approx A(\vartheta_t)[\vartheta_t + z_t]$$

$$\frac{d}{dt}z_t = -\beta_t\big[z_t - L_t w_t\big] \qquad \Leftrightarrow z_t \approx L_t w_t \quad \textit{If stable}$$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

NR flow: $\dfrac{d}{dt}\vartheta_t = -[\vartheta_t + z_t]$ $\qquad \vartheta_t + z_t = [A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$\bar{f}(\vartheta_t) = A(\vartheta_t)[\vartheta_t + z_t]$$

Variables w_t, z_t and matrix gain L_t designed so $z_t \approx L_t w_t$

$$\frac{d}{dt}\vartheta_t = -[\vartheta_t + L_t w_t] \qquad \Leftrightarrow \vartheta_t + z_t \approx 0$$

$$\frac{d}{dt}w_t = -\beta_t\big[A(\vartheta_t)[\vartheta_t + z_t] - \bar{f}(\vartheta_t)\big] \qquad \Leftrightarrow \bar{f}(\vartheta_t) \approx A(\vartheta_t)[\vartheta_t + z_t]$$

$$\frac{d}{dt}z_t = -\beta_t\big[z_t - L_t w_t\big] \qquad \Leftrightarrow z_t \approx L_t w_t \quad \textit{If stable}$$

How to choose gain for stability?

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

NR flow: $\dfrac{d}{dt}\vartheta_t = -[\vartheta_t + z_t]$ $\qquad \vartheta_t + z_t = [A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$\bar{f}(\vartheta_t) = A(\vartheta_t)[\vartheta_t + z_t]$$

Variables w_t, z_t and matrix gain L_t designed so $z_t \approx L_t w_t$

$$\frac{d}{dt}\vartheta_t = -[\vartheta_t + L_t w_t] \qquad \Leftrightarrow \vartheta_t + z_t \approx 0$$

$$\frac{d}{dt}w_t = -\beta_t\big[A(\vartheta_t)[\vartheta_t + z_t] - \bar{f}(\vartheta_t)\big] \qquad \Leftrightarrow \bar{f}(\vartheta_t) \approx A(\vartheta_t)[\vartheta_t + z_t]$$

$$\frac{d}{dt}z_t = -\beta_t\big[z_t - L_t w_t\big] \qquad \Leftrightarrow z_t \approx L_t w_t \quad \textit{If stable}$$

How to choose gain for stability?

$L_t = M A(\vartheta_t)^T$ works whenever Zap works! $\qquad M > 0$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$\frac{d}{dt}\vartheta_t = -[\vartheta_t + L_t w_t] \qquad \Leftrightarrow \vartheta_t + z_t \approx 0$$

$$\frac{d}{dt}w_t = -\beta_t\big[A(\vartheta_t)[\vartheta_t + z_t] - \bar{f}(\vartheta_t)\big] \qquad \Leftrightarrow \bar{f}(\vartheta_t) \approx A(\vartheta_t)[\vartheta_t + z_t]$$

$$\frac{d}{dt}z_t = -\beta_t\big[z_t - L_t w_t\big] \qquad \Leftrightarrow z_t \approx L_t w_t \quad \textit{If stable}$$

$L_t = MA(\vartheta_t)^\intercal$ works whenever Zap works! $\qquad M > 0$

Explanation from singular perturbation theory, fast dynamics are stable:

$$\frac{d}{dt}\begin{bmatrix} W_t^\theta \\ Z_t^\theta \end{bmatrix} = \begin{bmatrix} 0 & -A(\theta) \\ MA(\theta)^\intercal & -I \end{bmatrix}\begin{bmatrix} W_t^\theta \\ Z_t^\theta \end{bmatrix} - \begin{bmatrix} A(\theta)\theta - \bar{f}(\theta) \\ 0 \end{bmatrix}$$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$\frac{d}{dt}\vartheta_t = -[\vartheta_t + L_t w_t] \qquad \Leftrightarrow \vartheta_t + z_t \approx 0$$

$$\frac{d}{dt}w_t = -\beta_t\big[A(\vartheta_t)[\vartheta_t + z_t] - \bar{f}(\vartheta_t)\big] \qquad \Leftrightarrow \bar{f}(\vartheta_t) \approx A(\vartheta_t)[\vartheta_t + z_t]$$

$$\frac{d}{dt}z_t = -\beta_t\big[z_t - L_t w_t\big] \qquad \Leftrightarrow z_t \approx L_t w_t \quad \textit{If stable}$$

$L_t = MA(\vartheta_t)^\intercal$ works whenever Zap works! $\qquad M > 0$

Zap Zero Algorithm

$$\theta_{n+1} = \theta_n - \alpha_{n+1}[\theta_n + L_n w_n]$$

$$w_{n+1} = w_n - \beta_{n+1}\big[\widehat{A}_{n+1}[\theta_n + z_n] - f_{n+1}(\theta_n)\big]$$

$$z_{n+1} = z_n - \beta_{n+1}\big[z_n - M\widehat{A}_{n+1}^\intercal w_n\big]$$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$\frac{d}{dt}\vartheta_t = -[\vartheta_t + L_t w_t] \qquad \Leftrightarrow \vartheta_t + z_t \approx 0$$

$$\frac{d}{dt}w_t = -\beta_t\big[A(\vartheta_t)[\vartheta_t + z_t] - \bar{f}(\vartheta_t)\big] \qquad \Leftrightarrow \bar{f}(\vartheta_t) \approx A(\vartheta_t)[\vartheta_t + z_t]$$

$$\frac{d}{dt}z_t = -\beta_t\big[z_t - L_t w_t\big] \qquad \Leftrightarrow z_t \approx L_t w_t \quad \textit{If stable}$$

$L_t = M A(\vartheta_t)^\intercal$ works whenever Zap works! $\qquad M > 0$

Why $3 \times d$ dimensional?

$$\theta_{n+1} = \theta_n - \alpha_{n+1}[\theta_n + L_n w_n]$$

$$w_{n+1} = w_n - \beta_{n+1}\big[\widehat{A}_{n+1}[\theta_n + z_n] - f_{n+1}(\theta_n)\big]$$

$$z_{n+1} = z_n - \beta_{n+1}\big[z_n - M\widehat{A}_{n+1}^\intercal w_n\big]$$

Zap Zero 2023

Approximates $\frac{d}{dt}\vartheta_t = -[A(\vartheta_t)]^{-1}\bar{f}(\vartheta_t)$

$$\frac{d}{dt}\vartheta_t = -[\vartheta_t + L_t w_t] \qquad \Leftrightarrow \vartheta_t + z_t \approx 0$$

$$\frac{d}{dt}w_t = -\beta_t\big[A(\vartheta_t)[\vartheta_t + z_t] - \bar{f}(\vartheta_t)\big] \qquad \Leftrightarrow \bar{f}(\vartheta_t) \approx A(\vartheta_t)[\vartheta_t + z_t]$$

$$\frac{d}{dt}z_t = -\beta_t\big[z_t - L_t w_t\big] \qquad \Leftrightarrow z_t \approx L_t w_t \quad \textit{If stable}$$

$L_t = M A(\vartheta_t)^T$ works whenever Zap works! $\qquad M > 0$

Zap Zero Algorithm © Why $3 \times d$ dimensional?

$$\theta_{n+1} = \theta_n - \alpha_{n+1}[\theta_n + L_n w_n]$$

$$w_{n+1} = w_n - \beta_{n+1}\big[A_{n+1}[\theta_n + z_n] - f_{n+1}(\theta_n)\big]$$

$$z_{n+1} = z_n - \beta_{n+1}\big[z_n - M A_{n+1}^T w_n\big], \qquad A_{n+1} = \partial_\theta f_{n+1}(\theta_n)$$

To eliminate $\widehat{A}_n$ (estimate of $A(\theta_n)$)

Conclusions

Perhaps a surprise:

RL&SA remain fertile areas for research!

Recaps:

- Reinforcement Learning is cursed by dimension, variance, and nonlinear (algorithm) dynamics
- Second order methods can ensure stability—use them when you can
- Current formulations of Q-learning are not always well motivated—*projected Bellman equation?*

Recaps:

- Reinforcement Learning is cursed by dimension, variance, and nonlinear (algorithm) dynamics
- Second order methods can ensure stability—use them when you can
- Current formulations of Q-learning are not always well motivated—*projected Bellman equation?*

The future

- Beyond the projected Bellman equation,
  such as LP approaches [61, 62, 63, 64]

Recaps:

- Reinforcement Learning is cursed by dimension, variance, and nonlinear (algorithm) dynamics
- Second order methods can ensure stability—use them when you can
- Current formulations of Q-learning are not always well motivated—*projected Bellman equation?*

The future

- Beyond the projected Bellman equation,
  such as LP approaches [61, 62, 63, 64]
- Zap Zero and marriage with momentum

Recaps:

- Reinforcement Learning is cursed by dimension, variance, and nonlinear (algorithm) dynamics
- Second order methods can ensure stability—use them when you can
- Current formulations of Q-learning are not always well motivated—*projected Bellman equation?*

The future

- Beyond the projected Bellman equation,
  such as LP approaches [61, 62, 63, 64]
- Zap Zero and marriage with momentum
- Please don't forget about EXPLORATION!
  Extensions of Zap are unexplored for parameter dependent training

Recaps:

- Reinforcement Learning is cursed by dimension, variance, and nonlinear (algorithm) dynamics
- Second order methods can ensure stability—use them when you can
- Current formulations of Q-learning are not always well motivated—*projected Bellman equation?*

The future

- Beyond the projected Bellman equation,
  such as LP approaches [61, 62, 63, 64]
- Zap Zero and marriage with momentum
- Please don't forget about EXPLORATION!
  Extensions of Zap are unexplored for parameter dependent training

Thank you for your endurance!

I welcome questions and comments now and in the future

CS&RL In Three Parts

Part 1 Part 2 Part 3

Part 3: TD Learning and Actor-Critic Methods

Sean Meyn

COGNITION & CONTROL IN COMPLEX SYSTEMS

Department of Electrical and Computer Engineering University of Florida

Inria International Chair Inria, Paris

Thanks to to our sponsors: NSF and ARO

TD Learning & Actor Critic Methods

Outline

Return to Part 1 Return to Part 2

11 Resources & Background

12 TD Learning

13 Actor-Critic Method

14 Conclusions

15 References

Part 3: TD&AC **Resources**

References

Today's lecture based on
[8] CS&RL, Chapter 10: Setting the Stage, Return of the Actors

The notes section contains key historical references, including

[39] Schweitzer's perturbation theorem, which is a basis of Glynn's work in the 80's [42] and early work on actor critic methods by Konda and Borkar [90].

[48, 47, 17] The dissertations of Van Roy, Marbach and Konda (all with Tsitsiklis).

Sutton and Barto were inspirational at this time [50], and earlier work of Williams led to REINFORCE [43] (related to extremum seeking control and SPSA [99, 44]).

TD Learning

Note: avoiding the term SARSA

Stochastic Optimal Control (Review)

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
- $\gamma < 1$ a discount factor

Q function:

$$Q^*(x,u) = \min_{\boldsymbol{U}} \sum_{n=0}^{\infty} \gamma^n \mathsf{E}[c(X_n, U_n) \mid X(0) = x, U(0) = u]$$

Stochastic Optimal Control (Review)

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
- $\gamma < 1$ a discount factor

Q function:

$$Q^*(x,u) = \min_{\boldsymbol{U}} \sum_{n=0}^{\infty} \gamma^n \mathsf{E}[c(X_n, U_n) \mid X(0) = x, U(0) = u]$$

Bellman equation:

$$Q^*(x,u) = c(x,u) + \gamma \mathsf{E}\big[\min_{u'} Q^*(X_{n+1}, u') \mid X_n = x,\ U_n = u\big]$$

TD Learning

Given policy ϕ, approximate the fixed-policy Q-function

$$U(k) = \phi(X(k)),\ k \geq 1$$

$$Q^{\phi}(x,u) = \sum_{k=0}^{\infty} \gamma^k \mathsf{E}[c(X(k),U(k)) \mid X(0) = x\,,\ U(0) = u]$$

Motivation: 1. Policy Improvement $\quad \phi^{+}(x) = \arg\min_{u} Q^{\phi}(x,u)$

2. Sensitivity formula to be seen ...

TD Learning

Given policy ϕ, approximate the fixed-policy Q-function

$$U(k) = \phi(X(k)),\ k \geq 1$$

2. Sensitivity formula to be seen ... Requires $\gamma = 1$:

$$Q^{\phi}(x,u) = \sum_{k=0}^{\infty} \mathsf{E}[\tilde{c}(X(k),U(k)) \mid X(0) = x\,,\ U(0) = u]$$

$$= \text{const.} + \mathsf{E}\Big[\sum_{k=0}^{\tau_\bullet - 1} \tilde{c}(X(k),U(k)) \mid X(0) = x\,,\ U(0) = u\Big]$$

TD Learning

Given policy ϕ, approximate the fixed-policy Q-function

$$U(k) = \phi(X(k)),\ k \geq 1$$

DP equation:

$$Q^\phi(x,u) = c(x,u) + \gamma \mathsf{E}[\underline{Q}^\phi(X(k+1)) \mid X(k) = x\,,\ U(k) = u]$$

$$Q^\phi(X(k),U(k)) = c(X(k),U(k)) + \gamma \mathsf{E}[\underline{Q}^\phi(X(k+1)) \mid \mathcal{F}_k]$$

with new definition, $\underline{H}(x) = H(x,\phi(x))$.

TD Learning

Given policy ϕ, approximate the fixed-policy Q-function

$$U(k) = \phi(X(k)),\ k \geq 1$$

$$Q^\phi(X(k), U(k)) = c(X(k), U(k)) + \gamma \mathsf{E}[\underline{Q}^\phi(X(k+1)) \mid \mathcal{F}_k]$$

with new definition, $\underline{H}(x) = H(x, \phi(x))$.

Galerkin relaxation: θ^* is a root of the function

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q^\theta(X(k), U(k)) + c(X(k), U(k)) + \gamma \underline{Q}^\theta(X(k+1))\}\zeta_k]$$

Eligibility vector in $\mathbb{R}^d$

TD Learning

Given policy ϕ, approximate the fixed-policy Q-function

$$U(k) = \phi(X(k)),\ k \geq 1$$

$$Q^\phi(X(k), U(k)) = c(X(k), U(k)) + \gamma \mathsf{E}[\underline{Q}^\phi(X(k+1)) \mid \mathcal{F}_k]$$

with new definition, $\underline{H}(x) = H(x, \phi(x))$.

Galerkin relaxation: θ^* is a root of the function

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q^\theta(X(k), U(k)) + c(X(k), U(k)) + \gamma \underline{Q}^\theta(X(k+1))\}\zeta_k]$$

Linear family: $\{Q^\theta = \theta^\intercal \psi\}$

- $\frac{d}{dt}\vartheta = \bar{f}(\vartheta)$ is stable (clever choice of ζ_k, and with covariance Σ_ψ full rank)
- Alternatively, compute θ^* by a single matrix inversion

TD Learning

Given policy ϕ, approximate the fixed-policy Q-function

$$U(k) = \phi(X(k)),\ k \geq 1$$

$$Q^\phi(X(k), U(k)) = c(X(k), U(k)) + \gamma \mathsf{E}[\underline{Q}^\phi(X(k+1)) \mid \mathcal{F}_k]$$

with new definition, $\underline{H}(x) = H(x, \phi(x))$.

Galerkin relaxation: θ^* is a root of the function

$$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q^\theta(X(k), U(k)) + c(X(k), U(k)) + \gamma \underline{Q}^\theta(X(k+1))\}\zeta_k]$$

How to estimate $\theta^\star$ for neural network (nonlinear) architecture?

TD(λ) $\quad \bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q_k^\theta + c_k + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k]$

Choose $\lambda \in [0,1]$ as you please, and then for each n

$$\zeta_n = \sum_{k=0}^{n} (\gamma\lambda)^{n-k} \nabla_\theta [Q_k^\theta]\big|_{\theta=\theta_n}$$

Interpretation for linear function class: $\nabla_\theta[Q_k^\theta] = \psi_k = \psi(X_k, U_k)$

Basis observations passed through a low pass linear filter

TD(λ) $\quad \bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q_k^\theta + c_k + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k]$

Choose $\lambda \in [0,1]$ as you please, and then for each n

$$\zeta_n = \sum_{k=0}^{n} (\gamma\lambda)^{n-k} \nabla_\theta [Q_k^\theta]\big|_{\theta=\theta_n}$$

TD(λ) $\equiv$ SA algorithm:

$$\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, \xi_{n+1}) = \theta_n + \alpha_{n+1}\{-Q_n^{\theta_n} + c_n + \gamma \underline{Q}_{n+1}^{\theta_n}\}\zeta_n$$

$$\zeta_{n+1} = \gamma\lambda\zeta_n + \nabla_\theta [Q_n^\theta]\big|_{\theta=\theta_{n+1}}$$

with $\xi_{n+1} = (X_{n+1}, X_n, U_n, \zeta_n)$

LSTD and Zap $\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q_k^\theta + c_k + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k]$

We require the linearization matrix:

$$A(\theta, \xi_{k+1}) = \partial_\theta \{-Q_k^\theta + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k$$

LSTD and Zap $\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q_k^\theta + c_k + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k]$

We require the linearization matrix:

$$
\begin{aligned}
A(\theta, \xi_{k+1}) &= \partial_\theta \{-Q_k^\theta + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k \\
&= \big\{-\psi(X_k, U_k) + \gamma \psi\big(X_{k+1}, \phi(X_{k+1})\big)\big\}\zeta_k \quad \text{linear function class}
\end{aligned}
$$

LSTD and Zap $\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q^\theta_k + c_k + \gamma \underline{Q}^\theta_{k+1}\}\zeta_k]$

We require the linearization matrix:

$$A(\theta, \xi_{k+1}) = \partial_\theta\{-Q^\theta_k + \gamma \underline{Q}^\theta_{k+1}\}\zeta_k$$
$$= \big\{-\psi(X_k, U_k) + \gamma\psi\big(X_{k+1}, \phi(X_{k+1})\big)\big\}\zeta_k \quad \text{linear function class}$$

$A^* = \mathsf{E}\big[\{-\psi(X_k, U_k) + \gamma\psi\big(X_{k+1}, \phi(X_{k+1})\big)\}\zeta_k\big]$ is Hurwitz in this case

Only requirement is that $\Sigma_\psi > 0$

LSTD and Zap $\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q_k^\theta + c_k + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k]$

We require the linearization matrix:

$$A(\theta, \xi_{k+1}) = \partial_\theta\{-Q_k^\theta + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k$$
$$= \big\{-\psi(X_k, U_k) + \gamma\psi\big(X_{k+1}, \phi(X_{k+1})\big)\big\}\zeta_k \quad \text{linear function class}$$

$$\theta_{n+1} = \theta_n + \alpha_{n+1}[-\widehat{A}_{n+1}]^{-1} f(\theta_n, \xi_{n+1})$$
$$f(\theta_n, \xi_{n+1}) = \{-Q_n^{\theta_n} + c_n + \gamma \underline{Q}_{n+1}^{\theta_n}\}\zeta_n$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \alpha_{n+1}(A_{n+1} - \widehat{A}_n)$$
$$A_{n+1} = \{-\psi(X_n, U_n) + \gamma\psi(X_{n+1}, \phi(X_{n+1})\}\zeta_n$$

Zap SA doesn't require two time-scale algorithm when $A(\theta) \equiv A^*$

LSTD and Zap $\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}[\{-Q_k^\theta + c_k + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k]$

We require the linearization matrix:

$$A(\theta, \xi_{k+1}) = \partial_\theta\{-Q_k^\theta + \gamma \underline{Q}_{k+1}^\theta\}\zeta_k$$
$$= \big\{-\psi(X_k, U_k) + \gamma\psi\big(X_{k+1}, \phi(X_{k+1})\big)\big\}\zeta_k \quad \text{linear function class}$$

LSTD(λ)-Learning

$$\theta_{n+1} = \theta_n + \alpha_{n+1}[-\widehat{A}_{n+1}]^{-1} f(\theta_n, \xi_{n+1})$$
$$f(\theta_n, \xi_{n+1}) = \{-Q_n^{\theta_n} + c_n + \gamma \underline{Q}_{n+1}^{\theta_n}\}\zeta_n$$

$$\widehat{A}_{n+1} = \widehat{A}_n + \alpha_{n+1}(A_{n+1} - \widehat{A}_n)$$
$$A_{n+1} = \{-\psi(X_n, U_n) + \gamma\psi(X_{n+1}, \phi(X_{n+1})\}\zeta_n$$

Optimal convergence rate when $\alpha_n = 1/n$

What Is LSTD(λ) Solving?

Beautiful Theory for $\lambda = 1$

What Is LSTD(1) Solving?

Eternal gratitude to John Tsitsiklis!

Answered in the dissertation of Van Roy [17]:

$$\min_{\theta} \|Q^{\theta} - Q^{\Phi}\|_{\pi}^{2} = \sum_{x,u} \big(Q^{\theta}(x,u) - Q^{\Phi}(x,u)\big)^{2} \pi(x,u)$$

See [20, 19] and Chapters 9 & 10 of CS&RL

What Is LSTD(1) Solving?

Eternal gratitude to John Tsitsiklis!

Answered in the dissertation of Van Roy [17]:

$$\min_{\theta} \|Q^{\theta} - Q^{\phi}\|_{\pi}^2 = \sum_{x,u} \big(Q^{\theta}(x,u) - Q^{\phi}(x,u)\big)^2 \pi(x,u)$$

See [20, 19] and Chapters 9 & 10 of CS&RL

Foundation for actor-critic algorithms in the dissertation of Konda [48]

$$\theta_{n+1} = \theta_n - \alpha_{n+1} \breve{\nabla}_{\Gamma}(n) \qquad \bar{f}(\theta) = -\nabla \Gamma(\theta)$$

Unbiased gradient observation: $\breve{\nabla}_{\Gamma}(n) \stackrel{\text{def}}{=} \Lambda_n^{\theta_n} Q_n^{\theta_n}$

Score function: $\Lambda^{\theta}(x,u) = \nabla_{\theta} \log[\breve{\phi}^{\theta}(u \mid x)]$

See also [90, 49] and Chapter 10 of CS&RL

LSTD(1) Optimal Covariance May Be Large

From CTCN [33], *estimating relative value function for M/M/1 queue*:

LSTD(1) Optimal Covariance May Be Large

From CTCN [33], *estimating relative value function for M/M/1 queue*:

Architecture linear, two dimensional, and ideal in every sense

LSTD(1) Optimal Covariance May Be Large

From CTCN [33], *estimating relative value function for M/M/1 queue*:

Architecture linear, two dimensional, and ideal in every sense

Algorithm performance is far from ideal

State Weighting

Weighted Objective

$$\min_{\theta} \|Q^{\theta} - Q^{\Phi}\|^2_{\pi,\Omega} = \sum_{x,u} \frac{1}{\Omega(x,u)} \left(Q^{\theta}(x,u) - Q^{\Phi}(x,u)\right)^2 \pi(x,u)$$

State Weighting

Weighted Objective

$$\min_{\theta} \|Q^{\theta} - Q^{\Phi}\|^2_{\pi,\Omega} = \sum_{x,u} \frac{1}{\Omega(x,u)} \left(Q^{\theta}(x,u) - Q^{\Phi}(x,u)\right)^2 \pi(x,u)$$

Zap $\equiv$ LSTD(1) $\qquad \theta_n = [-\widehat{A}_n]^{-1} b_n$

$$\widehat{A}_{n+1} = \widehat{A}_n + \frac{1}{n+1}\Big[-\frac{1}{\Omega_{n+1}}\psi_{n+1}\psi_{n+1}^{\mathsf{T}} - \widehat{A}_n\Big]$$

$$b_{n+1} = b_n + \frac{1}{n+1}\big[\zeta_n c_n - b_n\big]$$

$$\zeta_{n+1} = \gamma\zeta_n + \frac{1}{\Omega_{n+1}}\psi_{n+1}$$

State Weighting

Design state weighting so that $f(\theta, \xi)$ becomes bounded in ξ:

State Weighting

Design state weighting so that $f(\theta, \xi)$ becomes bounded in ξ:

State Weighting

Design state weighting so that $f(\theta, \xi)$ becomes bounded in ξ:

Far more flexible and better motivated than introducing λ

$$\zeta_{n+1} = \gamma \zeta_n + \frac{1}{\Omega_{n+1}} \psi_{n+1}$$

Relative LSTD

Skip to AC

Far more flexible and well motivated than introducing λ?

$$\zeta_{n+1} = \gamma\zeta_n + \frac{1}{\Omega_{n+1}}\psi(X_{n+1})$$

Relative LSTD

Skip to AC

Far more flexible and well motivated than introducing λ?

$$\zeta_{n+1} = \gamma\zeta_n + \frac{1}{\Omega_{n+1}}\psi(X_{n+1})$$

Performance will degrade as γ gets close to one

Relative LSTD

Skip to AC

Far more flexible and well motivated than introducing λ?

$$\zeta_{n+1} = \gamma\zeta_n + \frac{1}{\Omega_{n+1}}\psi(X_{n+1})$$

Performance will degrade as γ gets close to one

Change your objective: $\min_{\theta} \|H^\theta - H^\Phi\|^2_{\pi,\Omega}$

with $H(x,u) = Q(x,u) - Q(x^\bullet, u^\bullet)$ (some fixed normalization)

Relative LSTD

Skip to AC

Far more flexible and well motivated than introducing λ?

$$\zeta_{n+1} = \gamma\zeta_n + \frac{1}{\Omega_{n+1}}\psi(X_{n+1})$$

Performance will degrade as γ gets close to one

Change your objective: $\min_\theta \|H^\theta - H^\Phi\|^2_{\pi,\Omega}$
with $H(x,u) = Q(x,u) - Q(x^\bullet, u^\bullet)$ (some fixed normalization)

Relative LSTD

Replace $\psi(x,u)$ by $\tilde{\psi}(x,u) \stackrel{\text{def}}{=} \psi(x,u) - \psi(x^\bullet, u^\bullet)$ where it appears

Relative LSTD

Skip to AC

Far more flexible and well motivated than introducing λ?

$$\zeta_{n+1} = \gamma\zeta_n + \frac{1}{\Omega_{n+1}}\psi(X_{n+1})$$

Performance will degrade as γ gets close to one

Change your objective: $\min_\theta \|H^\theta - H^\Phi\|^2_{\pi,\Omega}$
with $H(x,u) = Q(x,u) - Q(x^\bullet, u^\bullet)$ (some fixed normalization)

Relative LSTD

Replace $\psi(x,u)$ by $\tilde{\psi}(x,u) \stackrel{\text{def}}{=} \psi(x,u) - \psi(x^\bullet, u^\bullet)$ where it appears

Covariance is optimal, and uniformly bounded over $0 \le \gamma < 1$ [68]

Relative LSTD

Figure 1: Comparison of Q-learning and Relative Q-learning algorithms for the stochastic shortest path problem of [4]. The relative Q-learning algorithm is unaffected by large discounting.

Relative LSTD

Replace $\psi(x,u)$ by $\tilde{\psi}(x,u) \stackrel{\text{def}}{=} \psi(x,u) - \psi(x^\bullet, u^\bullet)$ where it appears

Covariance is optimal, and uniformly bounded over $0 \le \gamma < 1$ [68]

Actor-Critic Method

See Notes section of Chapter 10:

Actor-critic methods Glynn's research in the 1980s [142, 143] introduced likelihood ratio methods for stochastic gradient descent without bias, based in part on [313] (i.e., the sensitivity theory surveyed in Section 6.8). Just over one decade later this was extended and applied to obtain the first unbiased stochastic gradient descent approach for reinforcement learning [239, 240] (see also [32, 31]), followed soon after with new insights in [344]. This work was the start of the actor-critic revolution that followed.

Two time-scale stochastic approximation and associated variance theory was still evolving in the late 90s. Konda's research with Borkar [189, 190] and then Tsitsiklis [188, 193] helped to shed light on this topic, and [188, 192] introduced a major advancement in actor-critic theory and application: the compatible features property (10.47) was introduced in this work, based on the prior L_2 theory surveyed above [355, 356, 354, 353].

The introduction of the advantage function as a means to accelerate actor-critic algorithms was proposed in [172], following [23]. Prop. 9.2 of [172] was later applied to obtain the *trust region policy optimization* (TRPO) algorithm of [311], which spawned many other approaches.

The regenerative structure leading to the i.i.d. samples $\{S_n\}$ appearing in (10.24) suggests that the policy should be frozen between regeneration times in algorithm implementation, similar to the way qSGD was applied for the Mountain Car example in Section 4.7.1. This was one approach in [239, 240], and similar "episodic" approaches are used in [311, 312].

Actor-Critic Method

Temporal Difference Objectives

$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^{\theta}_{n+1}\zeta^{\theta}_{n}\big]$

Temporal difference: $\mathcal{D}^{\theta}_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^{\theta}(X_{n+1}) - Q^{\theta}(X_n, U_n)$

Hidden Goal of Q-Learning

Given $\{Q^{\theta} : \theta \in \mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*) = 0$

$\underline{H}(x) = \min_u H(x, u)$

Temporal Difference Objectives

$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^{\theta}_{n+1}\zeta^{\theta}_{n}\big]$

Temporal difference: $\mathcal{D}^{\theta}_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^{\theta}(X_{n+1}) - Q^{\theta}(X_n, U_n)$

Hidden Goal of Q-Learning

Given $\{Q^{\theta} : \theta \in \mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*) = 0$

$\underline{H}(x) = \min_u H(x, u)$

Goals of TD-Learning

Given $\{Q^{\theta} : \theta \in \mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*) = 0$

with new definition, $\underline{H}(x) = H(x, \phi(x))$

Temporal Difference Objectives

$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^\theta_{n+1}\zeta^\theta_n\big]$

Temporal difference: $\mathcal{D}^\theta_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)$

Hidden Goal of Q-Learning

Given $\{Q^\theta : \theta \in \mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*) = 0$

$\underline{H}(x) = \min_u H(x, u)$

Goals of TD-Learning

Given $\{Q^\theta : \theta \in \mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*) = 0$

with new definition, $\underline{H}(x) = H(x, \phi(x))$

TD(1): solves $\min_\theta \|Q^\theta - Q^\phi\|^2_\pi$

Temporal Difference Objectives

$\bar{f}(\theta) \stackrel{\text{def}}{=} \mathsf{E}\big[\mathcal{D}^\theta_{n+1}\zeta^\theta_n\big]$

Temporal difference: $\mathcal{D}^\theta_{n+1} = c(X_n, U_n) + \gamma \underline{Q}^\theta(X_{n+1}) - Q^\theta(X_n, U_n)$

Hidden Goal of Q-Learning

Given $\{Q^\theta : \theta \in \mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*) = 0$

$\underline{H}(x) = \min_u H(x, u)$

Goals of TD-Learning

Given $\{Q^\theta : \theta \in \mathbb{R}^d\}$, find θ^* that solves $\bar{f}(\theta^*) = 0$

with new definition, $\underline{H}(x) = H(x, \phi(x))$

TD(1): solves $\min_\theta \|Q^\theta - Q^\phi\|^2_\pi$

So What?

Reconsider our objective

For training in Q-learning we considered $\breve{\phi}_k(u \mid x) = \breve{\phi}^{\theta_k}(u \mid x)$

$$\breve{\phi}^{\theta}(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

with $\breve{\phi}_0^{\theta}$ an approximation of ϕ^{θ}.

Reconsider our objective

For training in Q-learning we considered $\breve{\phi}_k(u \mid x) = \breve{\phi}^{\theta_k}(u \mid x)$

$$\breve{\phi}^{\theta}(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

with $\breve{\phi}_0^{\theta}$ an approximation of ϕ^{θ}.

Why use Q-learning? We can apply SGD with objective

$$\Gamma(\theta) = \sum_{n=0}^{\infty} \gamma^n \mathsf{E}_{\mu}[c(X_n, U_n)]\,, \qquad U_n \sim \breve{\phi}^{\theta}(\,\cdot \mid X_n)$$

with your choice of μ.

Reconsider our objective

For training in Q-learning we considered $\breve{\phi}_k(u \mid x) = \breve{\phi}^{\theta_k}(u \mid x)$

$$\breve{\phi}^{\theta}(u \mid x) = (1-\varepsilon)\breve{\phi}_0^{\theta}(u \mid x) + \varepsilon \nu_{\mathcal{W}}(u)$$

with $\breve{\phi}_0^{\theta}$ an approximation of ϕ^{θ}.

Why use Q-learning? We can apply SGD with objective

$$\Gamma(\theta) = \sum_{n=0}^{\infty} \gamma^n \mathsf{E}_{\mu}[c(X_n, U_n)]\,, \qquad U_n \sim \breve{\phi}^{\theta}(\,\cdot \mid X_n)$$

with your choice of μ.

Exposition is more attractive for the average cost criterion,

$$\Gamma(\theta) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \mathsf{E}_{\mu}[c(X_n, U_n)]\,, \qquad U_n \sim \breve{\phi}^{\theta}(\,\cdot \mid X_n)$$

Schweitzer's Sensitivity Formula

Objective: $\Gamma(\theta) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \mathsf{E}_{\mu}[c(X_n, U_n)]\,, \qquad U_n \sim \breve{\phi}^{\theta}(\,\cdot\mid X_n)$

Schweitzer's Sensitivity Formula

Objective: $\Gamma(\theta) = \sum_z c(z)\varpi_\theta(z)$

ϖ_θ invariant pmf for $\Phi_k = (X_k; U_k)$

sum over all $z = (x; u) \in \mathsf{Z} = \mathsf{X} \times \mathsf{U}$

Schweitzer's Sensitivity Formula

Objective: $\Gamma(\theta) = \sum_z c(z)\varpi_\theta(z)$

ϖ_θ invariant pmf for $\Phi_k = (X_k; U_k)$

sum over all $z = (x; u) \in \mathsf{Z} = \mathsf{X} \times \mathsf{U}$

Denote Q_θ (note subscript) the solution to Poisson's equation,

$$\mathsf{E}[Q_\theta(\Phi_{k+1}) \mid \Phi_k] = Q_\theta(\Phi_k) - c(\Phi_k) + \Gamma(\theta)$$

Schweitzer's Sensitivity Formula

Objective: $\Gamma(\theta) = \sum_z c(z)\varpi_\theta(z)$

ϖ_θ invariant pmf for $\Phi_k = (X_k; U_k)$

sum over all $z = (x; u) \in \mathsf{Z} = \mathsf{X} \times \mathsf{U}$

Denote Q_θ (note subscript) the solution to Poisson's equation,

$$\sum_{z'} T_\theta(z, z')Q_\theta(z') = Q_\theta(z) - c(z) + \Gamma(\theta)$$

Schweitzer's Sensitivity Formula

Objective: $\Gamma(\theta) = \sum_z c(z)\varpi_\theta(z)$

ϖ_θ invariant pmf for $\Phi_k = (X_k; U_k)$

sum over all $z = (x; u) \in \mathsf{Z} = \mathsf{X} \times \mathsf{U}$

Denote Q_θ (note subscript) the solution to Poisson's equation,

$$\sum_{z'} T_\theta(z, z')Q_\theta(z') = Q_\theta(z) - c(z) + \Gamma(\theta)$$

with T_θ the transition matrix for $\{\Phi_k\}$, so that

$$\varpi_\theta(z') = \sum_z \varpi_\theta(z)T_\theta(z, z')$$

Schweitzer's Sensitivity Formula

Objective: $\Gamma(\theta) = \sum_z c(z)\varpi_\theta(z)$

ϖ_θ invariant pmf for $\Phi_k = (X_k; U_k)$

sum over all $z = (x; u) \in \mathsf{Z} = \mathsf{X} \times \mathsf{U}$

Denote Q_θ the solution to Poisson's equation,

$$\sum_{z'} T_\theta(z, z')Q_\theta(z') = Q_\theta(z) - c(z) + \Gamma(\theta)$$

Schweitzer 1968 [39]:

$$\nabla\Gamma(\theta) = \mathsf{E}_{\varpi_\theta}\left[\Lambda^\theta(\Phi_k)Q_\theta(\Phi_k)\right]$$

Score function: $\Lambda^\theta(z') = \nabla \log T_\theta(z, z') = \nabla_\theta \log[\breve{\phi}^\theta(u' \mid x')]$

Application to the Gibbs Policy

Schweitzer 1968 [39]: $\nabla\Gamma(\theta) = \mathsf{E}_{\varpi_\theta}\left[\Lambda^\theta(\Phi_k)Q_\theta(\Phi_k)\right]$

Score function: $\Lambda^\theta(z') = \nabla \log T_\theta(z, z') = \nabla_\theta \log[\breve{\phi}^\theta(u' \mid x')]$

Application to the Gibbs Policy

Schweitzer 1968 [39]: $\nabla\Gamma(\theta) = \mathsf{E}_{\varpi_\theta}\left[\Lambda^\theta(\Phi_k)Q_\theta(\Phi_k)\right]$

Score function: $\Lambda^\theta(z') = \nabla \log T_\theta(z, z') = \nabla_\theta \log[\breve{\phi}^\theta(u' \mid x')]$

Gibbs in notation of CS&RL:

Gibbs policy Given a d-dimensional basis vector ψ^0, consider for each θ the policy

$$\breve{\phi}^\theta(u \mid x) = \frac{1}{\kappa(\theta, x)} \exp\big(\theta^\intercal \psi^0(x, u)\big) \tag{10.52}$$

where κ is a normalizing constant (recall (9.46)). We then have from the definitions,

Lemma 10.21. *For the Gibbs policy,*

$$\Lambda^\theta(x, u) = \widetilde{\psi}_\theta(x, u) \stackrel{\text{def}}{=} \psi^0(x, u) - \underline{\psi}^0_\theta(x)\,, \qquad \underline{\psi}^0_\theta(x) \stackrel{\text{def}}{=} \sum_v \breve{\phi}^\theta(v \mid x)\psi^0(x, v) \qquad \square$$

Application to the Gibbs Policy

Schweitzer 1968 [39]: $\nabla\Gamma(\theta) = \mathsf{E}_{\varpi_\theta}\left[\Lambda^\theta(\Phi_k)Q_\theta(\Phi_k)\right]$

Score function: $\Lambda^\theta(z') = \nabla \log T_\theta(z, z') = \nabla_\theta \log[\breve{\phi}^\theta(u' \mid x')]$

Gibbs in notation of CS&RL:

Gibbs policy Given a d-dimensional basis vector ψ^0, consider for each θ the policy

$$\breve{\phi}^\theta(u \mid x) = \frac{1}{\kappa(\theta, x)} \exp\big(\theta^\intercal \psi^0(x, u)\big) \tag{10.52}$$

where κ is a normalizing constant (recall (9.46)). We then have from the definitions,

Lemma 10.21. *For the Gibbs policy,*

$$\Lambda^\theta(x, u) = \widetilde{\psi}_\theta(x, u) \stackrel{\text{def}}{=} \psi^0(x, u) - \underline{\psi}^0_\theta(x)\,, \qquad \underline{\psi}^0_\theta(x) \stackrel{\text{def}}{=} \sum_v \breve{\phi}^\theta(v \mid x)\psi^0(x, v) \qquad \square$$

Stationary point: $0 = \nabla\Gamma(\theta)\big|_{\theta=\theta^*} = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)Q_\theta(\Phi_k)\right]\big|_{\theta=\theta^*}$

Application to the Gibbs Policy

Schweitzer 1968 [39]: $\nabla\Gamma(\theta) = \mathsf{E}_{\varpi_\theta}\left[\Lambda^\theta(\Phi_k)Q_\theta(\Phi_k)\right]$

Score function: $\Lambda^\theta(z') = \nabla \log T_\theta(z, z') = \nabla_\theta \log[\breve{\phi}^\theta(u' \mid x')]$

Gibbs in notation of CS&RL:

Gibbs policy Given a d-dimensional basis vector ψ^0, consider for each θ the policy

$$\breve{\phi}^\theta(u \mid x) = \frac{1}{\kappa(\theta, x)} \exp\big(\theta^\intercal \psi^0(x, u)\big) \tag{10.52}$$

where κ is a normalizing constant (recall (9.46)). We then have from the definitions,

Lemma 10.21. *For the Gibbs policy,*

$$\Lambda^\theta(x, u) = \widetilde{\psi}_\theta(x, u) \stackrel{\text{def}}{=} \psi^0(x, u) - \underline{\psi}^0_\theta(x)\,, \qquad \underline{\psi}^0_\theta(x) \stackrel{\text{def}}{=} \sum_v \breve{\phi}^\theta(v \mid x)\psi^0(x, v) \qquad \square$$

Stationary point: $0 = \nabla\Gamma(\theta)\big|_{\theta=\theta^*} = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)Q_\theta(\Phi_k)\right]\big|_{\theta=\theta^*}$

Nothing like the projected Bellman equation

Actor-Critic for the Gibbs Policy

Gibbs policy Given a d-dimensional basis vector ψ^0, consider for each θ the policy

$$\breve{\phi}^{\theta}(u \mid x) = \frac{1}{\kappa(\theta, x)} \exp\big(\theta^{\mathsf{T}} \psi^0(x, u)\big) \tag{10.52}$$

where κ is a normalizing constant (recall (9.46)). We then have from the definitions,

Lemma 10.21. *For the Gibbs policy,*

$$\Lambda^{\theta}(x, u) = \widetilde{\psi}_{\theta}(x, u) \stackrel{\text{def}}{=} \psi^0(x, u) - \underline{\psi}^0_{\theta}(x)\,, \qquad \underline{\psi}^0_{\theta}(x) \stackrel{\text{def}}{=} \sum_{v} \breve{\phi}^{\theta}(v \mid x) \psi^0(x, v) \qquad \square$$

Stationary point: $0 = \nabla \Gamma(\theta)\big|_{\theta=\theta^*} = \mathsf{E}_{\varpi_\theta}\big[\widetilde{\psi}_\theta(\Phi_k) Q_\theta(\Phi_k)\big]\big|_{\theta=\theta^*}$

Actor-Critic for the Gibbs Policy

Stationary point: $0 = \nabla\Gamma(\theta)\big|_{\theta=\theta^*} = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)Q_\theta(\Phi_k)\right]\big|_{\theta=\theta^*}$
Konda and Tsitsiklis [48, 49]:

$$\nabla\Gamma(\theta) = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)\widehat{Q}_\theta(\Phi_k)\right]$$

with $\widehat{Q}_\theta$ the projection of Q_θ onto the span of $\{\widetilde{\psi}^i_\theta\}$ in $L_2(\varpi_\theta)$:

$$\widehat{Q}_\theta = \arg\min\left\{\mathsf{E}_{\varpi_\theta}\left[\{Q_\theta(\Phi_k) - Q(\Phi_k)\}^2\right] : Q = \omega^\intercal\widetilde{\psi}_\theta\,,\ \omega \in \mathbb{R}^d\right\}$$

$$\widehat{Q}_\theta = \omega_\theta^\intercal\widetilde{\psi}_\theta$$

Actor-Critic for the Gibbs Policy

Stationary point: $0 = \nabla\Gamma(\theta)\big|_{\theta=\theta^*} = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)Q_\theta(\Phi_k)\right]\big|_{\theta=\theta^*}$
Konda and Tsitsiklis [48, 49]:

$$\nabla\Gamma(\theta) = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)\widehat{Q}_\theta(\Phi_k)\right]$$

with $\widehat{Q}_\theta$ the projection of Q_θ onto the span of $\{\widetilde{\psi}^i_\theta\}$ in $L_2(\varpi_\theta)$:

$$\widehat{Q}_\theta = \omega_\theta^{\mathsf{T}}\widetilde{\psi}_\theta$$

Magic happens: ω_θ is the output of the TD(1) algorithm!

Actor-Critic for the Gibbs Policy

Stationary point: $0 = \nabla\Gamma(\theta)\big|_{\theta=\theta^*} = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)Q_\theta(\Phi_k)\right]\big|_{\theta=\theta^*}$
Konda and Tsitsiklis [48, 49]:

$$\nabla\Gamma(\theta) = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)\widehat{Q}_\theta(\Phi_k)\right]$$

with $\widehat{Q}_\theta$ the projection of Q_θ onto the span of $\{\widetilde{\psi}^i_\theta\}$ in $L_2(\varpi_\theta)$:

$$\widehat{Q}_\theta = \omega_\theta^T\widetilde{\psi}_\theta$$

Magic happens: ω_θ is the output of the TD(1) algorithm!

$$\theta_{n+1} = \theta_n - \alpha_{n+1}G_n\widetilde{\psi}_{\theta_n}(\Phi_k)\widehat{Q}_n(\Phi_k)\,, \qquad \widehat{Q}_n = \omega_n^T\widetilde{\psi}_{\theta_n}$$
$$\omega_{n+1} = \omega_n + \beta_{n+1}\{\text{TD(1) update based on } \theta_n\}$$

Actor-Critic for the Gibbs Policy

Stationary point: $0 = \nabla\Gamma(\theta)\big|_{\theta=\theta^*} = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)Q_\theta(\Phi_k)\right]\big|_{\theta=\theta^*}$
Konda and Tsitsiklis [48, 49]:

$$\nabla\Gamma(\theta) = \mathsf{E}_{\varpi_\theta}\left[\widetilde{\psi}_\theta(\Phi_k)\widehat{Q}_\theta(\Phi_k)\right]$$

with $\widehat{Q}_\theta$ the projection of Q_θ onto the span of $\{\widetilde{\psi}^i_\theta\}$ in $L_2(\varpi_\theta)$:

$$\widehat{Q}_\theta = \omega_\theta^T\widetilde{\psi}_\theta$$

Magic happens: ω_θ is the output of the TD(1) algorithm!

$$\theta_{n+1} = \theta_n - \alpha_{n+1}G_n\widetilde{\psi}_{\theta_n}(\Phi_k)\widehat{Q}_n(\Phi_k)\,, \qquad \widehat{Q}_n = \omega_n^T\widetilde{\psi}_{\theta_n}$$
$$\omega_{n+1} = \omega_n + \beta_{n+1}\{\text{TD(1) update based on } \theta_n\}$$

$G_n > 0$ *choose your favorite matrix gain*

Actor-Critic for the Gibbs Policy

$H_\theta^\omega \equiv \widehat{Q}$

Natural Actor-Critic Algorithm with Zap

The function class is defined using $\psi_\theta = \Lambda^\theta$.

For initialization $R_0 > 0$ $(d \times d)$, $\eta_0 \in \mathbb{R}$ and $\theta_0, \omega_0, \zeta_0 \in \mathbb{R}^d$,

$$\theta_{n+1} = \theta_n - \boxed{\alpha_{n+1}} G_n \check{\nabla}_\Gamma^v(n), \qquad \check{\nabla}_\Gamma^v(n) \stackrel{\text{def}}{=} \Lambda_n H_{\theta_n}^{\omega_n}(\Phi(n)), \quad G_n = R_n^{-1} \tag{10.55a}$$

$$\Lambda_n = \Lambda^{\theta_n}(\Phi(n)) \tag{10.55b}$$

$$\Phi(n+1) \sim T_{\theta_n}(z, \cdot), \quad \textit{with } z = \Phi(n) \tag{10.55c}$$

$$\left.\begin{aligned}
\mathcal{D}_{n+1} &= -H_{\theta_n}^{\omega_n}(\Phi(n)) + \tilde{c}_n + \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} H_{\theta_n}^{\omega_n}(\Phi(n+1)) \\
\omega_{n+1} &= \omega_n + \boxed{\beta_{n+1}} G_n \zeta_n \mathcal{D}_{n+1} \\
\zeta_{n+1} &= \lambda \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} \zeta_n + \psi_{\theta_{n+1}}(\Phi(n+1)) \\
\eta_{n+1} &= \eta_n + \beta_{n+1} \bar{c}_{n+1}, \qquad \bar{c}_{n+1} = c(\Phi(n+1)) - \eta_n \\
R_{n+1} &= R_n + \beta_{n+1}[\Lambda_{n+1}\Lambda_{n+1}^\intercal - R_n]
\end{aligned}\right\} \tag{10.55d}$$

Two time-scale algorithm

The inverse that defines the gain matrix $G_n = R_n^{-1}$ can be computed efficiently using the Matrix Inversion Lemma (A.1), or an algorithm can be obtained without matrix inversion by adapting the first-order Zap algorithm (8.52).

Actor-Critic for the Gibbs Policy

$H_\theta^\omega \equiv \widehat{Q}$

Natural Actor-Critic Algorithm with Zap

The function class is defined using $\psi_\theta = \Lambda^\theta$.

For initialization $R_0 > 0$ $(d \times d)$, $\eta_0 \in \mathbb{R}$ and $\theta_0, \omega_0, \zeta_0 \in \mathbb{R}^d$,

$$\theta_{n+1} = \theta_n - \alpha_{n+1} G_n \check{\nabla}_\Gamma^v(n)\,, \qquad \check{\nabla}_\Gamma^v(n) \stackrel{\text{def}}{=} \Lambda_n H_{\theta_n}^{\omega_n}(\Phi(n))\,, \quad G_n = R_n^{-1} \tag{10.55a}$$

$$\Lambda_n = \Lambda^{\theta_n}(\Phi(n)) \tag{10.55b}$$

$$\Phi(n+1) \sim T_{\theta_n}(z, \cdot)\,, \quad \textit{with } z = \Phi(n) \tag{10.55c}$$

$$\left.\begin{aligned}
\mathcal{D}_{n+1} &= -H_{\theta_n}^{\omega_n}(\Phi(n)) + \tilde{c}_n + \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} H_{\theta_n}^{\omega_n}(\Phi(n+1)) \\
\omega_{n+1} &= \omega_n + \beta_{n+1} G_n \zeta_n \mathcal{D}_{n+1} \\
\zeta_{n+1} &= \boxed{\lambda} \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} \zeta_n + \psi_{\theta_{n+1}}(\Phi(n+1)) \\
\eta_{n+1} &= \eta_n + \beta_{n+1} \bar{c}_{n+1}\,, \qquad \bar{c}_{n+1} = c(\Phi(n+1)) - \eta_n \\
R_{n+1} &= R_n + \beta_{n+1}[\Lambda_{n+1}\Lambda_{n+1}^\intercal - R_n]
\end{aligned}\right\} \boxed{\mathrm{TD}(\lambda)} \tag{10.55d}$$

The inverse that defines the gain matrix $G_n = R_n^{-1}$ can be computed efficiently using the Matrix Inversion Lemma (A.1), or an algorithm can be obtained without matrix inversion by adapting the first-order Zap algorithm (8.52).

Actor-Critic for the Gibbs Policy

$H_\theta^\omega \equiv \widehat{Q}$

Natural Actor-Critic Algorithm with Zap

The function class is defined using $\psi_\theta = \Lambda^\theta$.

For initialization $R_0 > 0$ $(d \times d)$, $\eta_0 \in \mathbb{R}$ and $\theta_0, \omega_0, \zeta_0 \in \mathbb{R}^d$,

$$\theta_{n+1} = \theta_n - \alpha_{n+1} G_n \check{\nabla}_\Gamma^v(n), \qquad \check{\nabla}_\Gamma^v(n) \stackrel{\text{def}}{=} \Lambda_n H_{\theta_n}^{\omega_n}(\Phi(n)), \quad G_n = R_n^{-1} \tag{10.55a}$$

$$\Lambda_n = \Lambda^{\theta_n}(\Phi(n)) \tag{10.55b}$$

$$\Phi(n+1) \sim T_{\theta_n}(z, \cdot), \quad \textit{with } z = \Phi(n) \tag{10.55c}$$

$$\left.\begin{aligned}
\mathcal{D}_{n+1} &= -H_{\theta_n}^{\omega_n}(\Phi(n)) + \tilde{c}_n + \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} H_{\theta_n}^{\omega_n}(\Phi(n+1)) \\
\omega_{n+1} &= \omega_n + \beta_{n+1} G_n \zeta_n \mathcal{D}_{n+1} \\
\zeta_{n+1} &= \boxed{\lambda} \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} \zeta_n + \psi_{\theta_{n+1}}(\Phi(n+1)) \\
\eta_{n+1} &= \eta_n + \beta_{n+1} \tilde{c}_{n+1}, \qquad \tilde{c}_{n+1} = c(\Phi(n+1)) - \eta_n \\
R_{n+1} &= R_n + \beta_{n+1}[\Lambda_{n+1}\Lambda_{n+1}^{\mathrm{T}} - R_n]
\end{aligned}\right\} \boxed{\mathrm{TD}(\lambda)} \tag{10.55d}$$

The inverse that defines the gain matrix $G_n = R_n^{-1}$ can be computed efficiently using the Matrix Inversion Lemma (A.1), or an algorithm can be obtained without matrix inversion by adapting the first-order Zap algorithm (8.52).

Practitioners use $\lambda = 0$

Also, Note: with basis $\widetilde{\psi}_\theta$ we are estimating the *advantage function*

(for those in the know).

Actor-Critic for the Gibbs Policy

$H_\theta^\omega \equiv \widehat{Q}$

Natural Actor-Critic Algorithm with Zap

The function class is defined using $\psi_\theta = \Lambda^\theta$.

For initialization $R_0 > 0$ $(d \times d)$, $\eta_0 \in \mathbb{R}$ and $\theta_0, \omega_0, \zeta_0 \in \mathbb{R}^d$,

$$\theta_{n+1} = \theta_n - \alpha_{n+1} G_n \widehat{\nabla}_\Gamma^v(n), \qquad \widehat{\nabla}_\Gamma^v(n) \stackrel{\text{def}}{=} \Lambda_n H_{\theta_n}^{\omega_n}(\Phi(n)), \quad G_n = R_n^{-1} \tag{10.55a}$$

$$\Lambda_n = \Lambda^{\theta_n}(\Phi(n)) \tag{10.55b}$$

$$\Phi(n+1) \sim T_{\theta_n}(z, \cdot), \quad \textit{with } z = \Phi(n) \tag{10.55c}$$

$$\left.\begin{aligned}
\mathcal{D}_{n+1} &= -H_{\theta_n}^{\omega_n}(\Phi(n)) + \tilde{c}_n + \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} H_{\theta_n}^{\omega_n}(\Phi(n+1)) \\
\omega_{n+1} &= \omega_n + \beta_{n+1} G_n \zeta_n \mathcal{D}_{n+1} \\
\zeta_{n+1} &= \boxed{\lambda} \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} \zeta_n + \psi_{\theta_{n+1}}(\Phi(n+1)) \\
\eta_{n+1} &= \eta_n + \beta_{n+1} \tilde{c}_{n+1}, \qquad \tilde{c}_{n+1} = c(\Phi(n+1)) - \eta_n \\
R_{n+1} &= R_n + \beta_{n+1}[\Lambda_{n+1}\Lambda_{n+1}^{\mathsf{T}} - R_n]
\end{aligned}\right\} \boxed{\mathrm{TD}(\lambda)} \tag{10.55d}$$

The inverse that defines the gain matrix $G_n = R_n^{-1}$ can be computed efficiently using the Matrix Inversion Lemma (A.1), or an algorithm can be obtained without matrix inversion by adapting the first-order Zap algorithm (8.52).

Practitioners use $\boxed{\lambda = 0}$

For Gibbs, a limit of the algorithm is a pair (θ^*, ω^*) satisfying

$$0 = \mathsf{E}\big[\widetilde{\psi}_{\theta^*}(\Phi_n)\widehat{Q}(\Phi_n)\big], \qquad \widehat{Q} = \omega^{*\mathsf{T}}\widetilde{\psi}_{\theta^*}$$

$$0 = \mathsf{E}\big[\{c(X_n, U_n) + \mathbb{I}\{\Phi_{n+1} \neq z^\bullet\}\widehat{Q}(\Phi_{n+1}) - \widehat{Q}(\Phi_n)\}\widetilde{\psi}_{\theta^*}(\Phi_n)\big]$$

Projected Bellman equation appears

Actor-Critic for the Gibbs Policy

$H_\theta^\omega \equiv \widehat{Q}$

Natural Actor-Critic Algorithm with Zap

The function class is defined using $\psi_\theta = \Lambda^\theta$.

For initialization $R_0 > 0$ $(d \times d)$, $\eta_0 \in \mathbb{R}$ and $\theta_0, \omega_0, \zeta_0 \in \mathbb{R}^d$,

$$\theta_{n+1} = \theta_n - \alpha_{n+1} G_n \widehat{\nabla}_\Gamma^v(n), \qquad \widehat{\nabla}_\Gamma^v(n) \stackrel{\text{def}}{=} \Lambda_n H_{\theta_n}^{\omega_n}(\Phi(n)), \quad G_n = R_n^{-1} \tag{10.55a}$$

$$\Lambda_n = \Lambda^{\theta_n}(\Phi(n)) \tag{10.55b}$$

$$\Phi(n+1) \sim T_{\theta_n}(z, \cdot), \quad \textit{with } z = \Phi(n) \tag{10.55c}$$

$$\left.\begin{aligned}
\mathcal{D}_{n+1} &= -H_{\theta_n}^{\omega_n}(\Phi(n)) + \tilde{c}_n + \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} H_{\theta_n}^{\omega_n}(\Phi(n+1)) \\
\omega_{n+1} &= \omega_n + \beta_{n+1} G_n \zeta_n \mathcal{D}_{n+1} \\
\zeta_{n+1} &= \boxed{\lambda} \mathbb{1}\{\Phi(n+1) \neq z^\bullet\} \zeta_n + \psi_{\theta_{n+1}}(\Phi(n+1)) \\
\eta_{n+1} &= \eta_n + \beta_{n+1} \tilde{c}_{n+1}, \qquad \tilde{c}_{n+1} = c(\Phi(n+1)) - \eta_n \\
R_{n+1} &= R_n + \beta_{n+1}[\Lambda_{n+1}\Lambda_{n+1}^{\mathsf{T}} - R_n]
\end{aligned}\right\} \boxed{\mathrm{TD}(\lambda)} \tag{10.55d}$$

The inverse that defines the gain matrix $G_n = R_n^{-1}$ can be computed efficiently using the Matrix Inversion Lemma (A.1), or an algorithm can be obtained without matrix inversion by adapting the first-order Zap algorithm (8.52).

Practitioners use $\boxed{\lambda = 0}$

For Gibbs, a limit of the algorithm is a pair (θ^*, ω^*) satisfying

$$0 = \mathsf{E}\big[\widetilde{\psi}_{\theta^*}(\Phi_n)\widehat{Q}(\Phi_n)\big], \qquad \widehat{Q} = \omega^{*\mathsf{T}}\widetilde{\psi}_{\theta^*}$$

$$0 = \mathsf{E}\big[\{c(X_n, U_n) + \mathbb{I}\{\Phi_{n+1} \neq z^\bullet\}\widehat{Q}(\Phi_{n+1}) - \widehat{Q}(\Phi_n)\}\widetilde{\psi}_{\theta^*}(\Phi_n)\big]$$

Projected Bellman equation appears

Is there theory to justify $\lambda \neq 1$?

A. Cooper and S. Meyn. Reinforcement learning design for quickest change detection. IEEE CDC and arXiv:2403.14109, 2024.

Figure 2: Statistics of the gradient estimate as a function of θ, based on the Actor Critic method: (a) Empirical variance of the gradient estimates. (b) Gradient estimates using $N = 10^4$ episodes. (c) Objective and its approximation obtained from integrating the gradient estimate.

Conclusions & Future Directions

Conclusions

For the stochastic control problems I have considered,

- I'm not impressed by actor-critic methods!
- I commonly find massive variance.
- This is tamed by taking $\lambda = 0$, but why should that be successful?

Conclusions

For the stochastic control problems I have considered,

- I'm not impressed by actor-critic methods!
- I commonly find massive variance.
- This is tamed by taking $\lambda = 0$, but why should that be successful?

Future work:

- Find new formulations that have the firm theory of Actor-Critic methods without the noise.
- Develop new formulations of Q-learning

(work in progress [61, 62, 63, 64]).

Conclusions

For the stochastic control problems I have considered,

- I'm not impressed by actor-critic methods!
- I commonly find massive variance.
- This is tamed by taking $\lambda = 0$, but why should that be successful?

Future work:

- Find new formulations that have the firm theory of Actor-Critic methods without the noise.
- Develop new formulations of Q-learning

(work in progress [61, 62, 63, 64]).

Thank you once more for your endurance!

Control Systems
& Reinforcement Learning
Sean Meyn
CAMBRIDGE
UNIVERSITY PRESS

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

Control Background I

[1] K. J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

[2] A. Fradkov and B. T. Polyak. Adaptive and robust control in the USSR. *IFAC–PapersOnLine*, 53(2):1373–1378, 2020. 21th IFAC World Congress.

[3] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos. *Nonlinear and adaptive control design*. John Wiley & Sons, Inc., 1995.

[4] K. J. Åström. Theory and applications of adaptive control—a survey. *Automatica*, 19(5):471–486, 1983.

[5] K. J. Åström. Adaptive control around 1960. *IEEE Control Systems Magazine*, 16(3):44–49, 1996.

[6] L. Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22(4):551–575, 1977.

[7] N. Matni, A. Proutiere, A. Rantzer, and S. Tu. From self-tuning regulators to reinforcement learning and back again. In *Proc. of the IEEE Conf. on Dec. and Control*, pages 3724–3740, 2019.

RL Background I

[8] S. Meyn. *Control Systems and Reinforcement Learning*. Cambridge University Press, Cambridge, 2021.

[9] S. Meyn. *The projected Bellman equation in reinforcement learning*. *IEEE Transactions on Automatic Control*, pages 8323–8337, 2024.

[10] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press. On-line edition at `http://www.cs.ualberta.ca/~sutton/book/the-book.html`, Cambridge, MA, 2nd edition, 2018.

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

[12] D. P. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, Belmont, MA, 2019.

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

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

[15] J. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202, 1994.

RL Background II

[16] T. Jaakola, M. Jordan, and S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201, 1994.

[17] B. Van Roy. *Learning and Value Function Approximation in Complex Decision Processes*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1998.

[18] J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming. *Mach. Learn.*, 22(1-3):59–94, 1996.

[19] 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.

[20] J. N. Tsitsiklis and B. V. Roy. Average cost temporal-difference learning. *Automatica*, 35(11):1799–1808, 1999.

[21] 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.

[22] 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.

RL Background III

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

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

[25] 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.

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

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

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

[29] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton. Toward off-policy learning control with function approximation. In *Proc. ICML*, pages 719–726, USA, 2010. Omnipress.

RL Background IV

[30] 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.

[31] A. M. Devraj and S. P. Meyn. Zap Q-learning. In *Proc. of the Intl. Conference on Neural Information Processing Systems*, pages 2232–2241, 2017.

[32] S. Chen, A. M. Devraj, F. Lu, A. Busic, and S. Meyn. Zap Q-Learning with nonlinear function approximation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems, and arXiv e-prints 1910.05405*, volume 33, pages 16879–16890, 2020.

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

DQN:

[34] M. Riedmiller. *Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method.* In J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, editors, *Machine Learning: ECML 2005*, pages 317–328, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

RL Background V

[35] S. Lange, T. Gabel, and M. Riedmiller. *Batch reinforcement learning.* In *Reinforcement learning*, pages 45–73. Springer, 2012.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. *Playing Atari with deep reinforcement learning. ArXiv*, abs/1312.5602, 2013.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. *Human-level control through deep reinforcement learning. Nature*, 518:529–533, 2015.

[38] O. Anschel, N. Baram, and N. Shimkin. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In *Proc. of ICML*, pages 176–185. JMLR.org, 2017.

Actor Critic / Policy Gradient

[39] P. J. Schweitzer. Perturbation theory and finite Markov chains. *J. Appl. Prob.*, 5:401–403, 1968.

[40] C. D. Meyer, Jr. The role of the group generalized inverse in the theory of finite Markov chains. *SIAM Review*, 17(3):443–464, 1975.

RL Background VI

[41] P. W. Glynn. Stochastic approximation for Monte Carlo optimization. In *Proceedings of the 18th conference on Winter simulation*, pages 356–365, 1986.

[42] P. W. Glynn. Likelihood ratio gradient estimation: An overview. In *Proc. of the Winter Simulation Conference*, pages 366–375, 1987.

[43] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[44] J. C. Spall. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica*, 33(1):109–112, 1997.

[45] T. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in neural information processing systems*, pages 345–352, 1995.

[46] X.-R. Cao and H.-F. Chen. Perturbation realization, potentials, and sensitivity analysis of Markov processes. *IEEE Transactions on Automatic Control*, 42(10):1382–1393, Oct 1997.

[47] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of Markov reward processes. *IEEE Trans. Automat. Control*, 46(2):191–209, 2001.

RL Background VII

[48] V. Konda. *Actor-critic algorithms*. PhD thesis, Massachusetts Institute of Technology, 2002.

[49] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[50] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[51] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[52] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[53] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of Markov reward processes. *IEEE Trans. Automat. Control*, 46(2):191–209, 2001.

[54] S. M. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

RL Background VIII

[55] H. Mania, A. Guy, and B. Recht. Simple random search provides a competitive approach to reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1800–1809, 2018.

MDPs, LPs and Convex Q:

[56] A. S. Manne. Linear programming and sequential decisions. *Management Sci.*, 6(3):259–267, 1960.

[57] C. Derman. *Finite State Markovian Decision Processes*, volume 67 of *Mathematics in Science and Engineering*. Academic Press, Inc., 1970.

[58] V. S. Borkar. Convex analytic methods in Markov decision processes. In *Handbook of Markov decision processes*, volume 40 of *Internat. Ser. Oper. Res. Management Sci.*, pages 347–375. Kluwer Acad. Publ., Boston, MA, 2002.

[59] D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Res.*, 51(6):850–865, 2003.

[60] D. P. de Farias and B. Van Roy. *A cost-shaping linear program for average-cost approximate dynamic programming with performance guarantees. Math. Oper. Res.*, 31(3):597–620, 2006.

RL Background IX

[61] P. G. Mehta and S. P. Meyn. *Q-learning and Pontryagin's minimum principle.* In *Proc. of the IEEE Conf. on Dec. and Control*, pages 3598–3605, Dec. 2009.

[62] J. Bas Serrano, S. Curi, A. Krause, and G. Neu. Logistic Q-learning. In A. Banerjee and K. Fukumizu, editors, *Proc. of The Intl. Conference on Artificial Intelligence and Statistics*, volume 130, pages 3610–3618, 13–15 Apr 2021.

[63] F. Lu, P. G. Mehta, S. P. Meyn, and G. Neu. Convex Q-learning. In *American Control Conf.*, pages 4749–4756. IEEE, 2021.

[64] F. Lu, P. G. Mehta, S. P. Meyn, and G. Neu. Convex analytic theory for convex Q-learning. In *IEEE Conference on Decision and Control*, pages 4065–4071, Dec 2022.

Gator Nation:

[65] A. M. Devraj, A. Bušić, and S. Meyn. Fundamental design principles for reinforcement learning algorithms. In K. G. Vamvoudakis, Y. Wan, F. L. Lewis, and D. Cansever, editors, *Handbook on Reinforcement Learning and Control*, Studies in Systems, Decision and Control series (SSDC, volume 325). Springer, 2021.

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

RL Background X

[67] A. M. Devraj. *Reinforcement Learning Design with Optimal Learning Rate*. PhD thesis, University of Florida, 2019.

[68] A. M. Devraj and S. P. Meyn. Q-learning with uniformly bounded variance: Large discounting is not a barrier to fast learning. *IEEE Trans Auto Control (and arXiv:2002.10301)*, 2021.

[69] A. M. Devraj, A. Bušić, and S. Meyn. On matrix momentum stochastic approximation and applications to Q-learning. In *Allerton Conference on Communication, Control, and Computing*, pages 749–756, Sep 2019.

Stochastic Miscellanea I

[70] S. Asmussen and P. W. Glynn. *Stochastic Simulation: Algorithms and Analysis*, volume 57 of *Stochastic Modelling and Applied Probability*. Springer-Verlag, New York, 2007.

[71] P. W. Glynn and S. P. Meyn. *A Liapounov bound for solutions of the Poisson equation. Ann. Probab.*, 24(2):916–931, 1996.

[72] 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.

[73] R. Douc, E. Moulines, P. Priouret, and P. Soulier. *Markov Chains*. Springer, 2018.

Stochastic Approximation I

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

[75] 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.

[76] 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.

[77] V. Borkar, S. Chen, A. Devraj, I. Kontoyiannis, and S. Meyn. The ODE method for asymptotic statistics in stochastic approximation and reinforcement learning. To appear, Annals of Applied Probability (preprint at arXiv e-prints:2110.14427), 2021.

[78] A. Durmus, E. Moulines, A. Naumov, and S. Samsonov. Finite-time high-probability bounds for Polyak–Ruppert averaged iterates of linear stochastic approximation. *Mathematics of Operations Research*, 2024.

[79] C. K. Lauand and S. Meyn. Revisiting step-size assumptions in stochastic approximation. *arXiv 2405.17834*, 2024.

Stochastic Approximation II

[80] C. K. Lauand and S. Meyn. The curse of memory in stochastic approximation. In *Proc. IEEE Conference on Decision and Control (extended version available at arXiv 2309.02944)*, pages 7803–7809, 2023.

[81] A. Cooper and S. Meyn. Reinforcement learning design for quickest change detection—extended paper. *arXiv:2403.14109*, 2024.

[82] M. Benaïm. Dynamics of stochastic approximation algorithms. In *Séminaire de Probabilités, XXXIII*, pages 1–68. Springer, Berlin, 1999.

[83] V. Borkar and S. P. Meyn. Oja's algorithm for graph clustering, Markov spectral decomposition, and risk sensitive control. *Automatica*, 48(10):2512–2519, 2012.

[84] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466, 09 1952.

[85] J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control.* John Wiley & Sons, 2003.

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

Stochastic Approximation III

[87] 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.

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

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

[90] V. R. Konda and V. S. Borkar. Actor-critic–type learning algorithms for Markov decision processes. *SIAM Journal on control and Optimization*, 38(1):94–123, 1999.

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

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

Stochastic Approximation IV

[93] W. Mou, C. Junchi Li, M. J. Wainwright, P. L. Bartlett, and M. I. Jordan. On Linear Stochastic Approximation: Fine-grained Polyak-Ruppert and Non-Asymptotic Concentration. *arXiv e-prints*, page arXiv:2004.04719, Apr. 2020.

Optimization and ODEs I

[94] W. Su, S. Boyd, and E. Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in neural information processing systems*, pages 2510–2518, 2014.

[95] B. Shi, S. S. Du, W. Su, and M. I. Jordan. Acceleration via symplectic discretization of high-resolution differential equations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5744–5752. Curran Associates, Inc., 2019.

[96] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[97] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, 1983.

QSA and Extremum Seeking Control I

[98] C. K. Lauand and S. Meyn. Extremely fast convergence rates for extremum seeking control with Polyak-Ruppert averaging. *arXiv 2206.00814*, 2022.

[99] C. K. Lauand and S. Meyn. Quasi-stochastic approximation: Design principles with applications to extremum seeking control. *IEEE Control Systems Magazine*, 43(5):111–136, Oct 2023.

[100] C. K. Lauand and S. Meyn. Markovian foundations for quasi stochastic approximation. *SIAM Journal on Control and Optimization (pre-publication version arXiv 2207.06371)*, 63(1):402–430, 2025.

[101] B. Lapeybe, G. Pages, and K. Sab. Sequences with low discrepancy generalisation and application to Robbins-Monro algorithm. *Statistics*, 21(2):251–272, 1990.

[102] S. Laruelle and G. Pagès. Stochastic approximation with averaging innovation applied to finance. *Monte Carlo Methods and Applications*, 18(1):1–51, 2012.

[103] S. Shirodkar and S. Meyn. Quasi stochastic approximation. In *Proc. of the 2011 American Control Conference (ACC)*, pages 2429–2435, July 2011.

[104] S. Chen, A. Devraj, A. Bernstein, and S. Meyn. Revisiting the ODE method for recursive algorithms: Fast using quasi stochastic approximation. *Journal of Systems Science and Complexity*, 34(5):1681–1702, 2021.

QSA and Extremum Seeking Control II

[105] Y. Chen, A. Bernstein, A. Devraj, and S. Meyn. Model-Free Primal-Dual Methods for Network Optimization with Application to Real-Time Optimal Power Flow. In *Proc. of the American Control Conf.*, pages 3140–3147, Sept. 2019.

[106] S. Bhatnagar and V. S. Borkar. Multiscale chaotic spsa and smoothed functional algorithms for simulation optimization. *Simulation*, 79(10):568–580, 2003.

[107] S. Bhatnagar, M. C. Fu, S. I. Marcus, and I.-J. Wang. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 13(2):180–209, 2003.

[108] M. Le Blanc. Sur l'electrification des chemins de fer au moyen de courants alternatifs de frequence elevee [On the electrification of railways by means of alternating currents of high frequency]. *Revue Generale de l'Electricite*, 12(8):275–277, 1922.

[109] Y. Tan, W. H. Moase, C. Manzie, D. Nešić, and I. M. Y. Mareels. Extremum seeking from 1922 to 2010. In *Proceedings of the 29th Chinese Control Conference*, pages 14–26, July 2010.

[110] P. F. Blackman. Extremum-seeking regulators. In *An Exposition of Adaptive Control*. Macmillan, 1962.

QSA and Extremum Seeking Control III

[111] J. Sternby. Adaptive control of extremum systems. In H. Unbehauen, editor, *Methods and Applications in Adaptive Control*, pages 151–160, Berlin, Heidelberg, 1980. Springer Berlin Heidelberg.

[112] J. Sternby. Extremum control systems–an area for adaptive control? In *Joint Automatic Control Conference*, number 17, page 8, 1980.

[113] K. B. Ariyur and M. Krstić. *Real Time Optimization by Extremum Seeking Control*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[114] M. Krstić and H.-H. Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4):595 – 601, 2000.

[115] S. Liu and M. Krstic. Introduction to extremum seeking. In *Stochastic Averaging and Stochastic Extremum Seeking*, Communications and Control Engineering. Springer, London, 2012.

[116] O. Trollberg and E. W. Jacobsen. On the convergence rate of extremum seeking control. In *European Control Conference (ECC)*, pages 2115–2120. 2014.

[117] Y. Bugeaud. *Linear forms in logarithms and applications*, volume 28 of *IRMA Lectures in Mathematics and Theoretical Physics*. EMS Press, 2018.

QSA and Extremum Seeking Control IV

[118] G. Wüstholz, editor. *A Panorama of Number Theory—Or—The View from Baker's Garden*. Cambridge University Press, 2002.