Zap Stochastic Approximation and Reinforcement Learning

Reading Group Network Theory
Lincs

François Durand (NBLF)
Based on works by Ana Bušić (Inria / ENS),
Adithya M. Devraj and Sean Meyn (University of Florida)

October 6, 2020

Motivation:
Stochastic Approximation and RL

What is Stochastic Approximation?

Problem

- W random variable, $\theta \in \mathbb{R}^d$ variable.
- $f(\theta, W) \in \mathbb{R}^d$.
- $\bar{f}(\theta) := \mathsf{E}[f(\theta, W)]$.

Goal: find θ^* s.t.

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

What is Stochastic Approximation?

Problem

- W random variable, $\theta \in \mathbb{R}^d$ variable.
- $f(\theta, W) \in \mathbb{R}^d$.
- $\bar{f}(\theta) := \mathsf{E}[f(\theta, W)]$.

Goal: find θ^* s.t.

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

Traditional example (with $d = 1$)

- θ: dosage of a medicine (e.g. insulin).
- $f(\theta, W)$: effect (e.g. blood sugar level − ideal blood sugar level).
- θ^*: ideal dosage s.t. $\bar{f}(\theta^*) = 0$.

Robbins-Monro Algorithm

Principle

Initial estimate: θ_0 (arbitrary).
Update rule:

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

where the step-size α_{n+1} is part of the algorithm design.

Robbins-Monro Algorithm

Principle

Initial estimate: θ_0 (arbitrary).
Update rule:

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

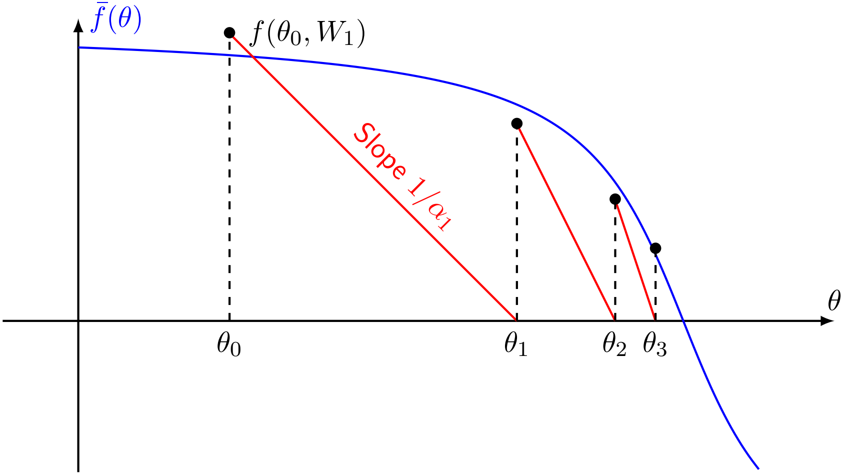
where the step-size α_{n+1} is part of the algorithm design.

Traditional example (continued)

- Try dosage θ_0 with patient 1.
- This gives effect $f(\theta_0, W_1)$: a noisy version of $\bar{f}(\theta_0)$.
- New estimated dosage $\theta_1 = \theta_0 + \alpha_1 f(\theta_0, W_1)$.
- Etc.

Robbins-Monro Algorithm: Illustration

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



Robbins-Monro Algorithm: Convergence

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

The step-size satisfies:

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

Usually we take $\alpha_n = 1/n$.

Monte-Carlo Estimation, Seen as an SA Approach

Problem

We want to estimate $\mathsf{E}[W]$, where W is a random variable.

Monte-Carlo Estimation, Seen as an SA Approach

Problem

We want to estimate $\mathsf{E}[W]$, where W is a random variable.

Conversion to SA problem

Let $f(\theta, W) = W - \theta$. Then $\bar{f}(\theta) = \mathsf{E}[W] - \theta$.
We want to find θ^* s.t. $\bar{f}(\theta^*) = 0$.

Monte-Carlo Estimation, Seen as an SA Approach

Problem

We want to estimate $\mathsf{E}[W]$, where W is a random variable.

Conversion to SA problem

Let $f(\theta, W) = W - \theta$. Then $\bar{f}(\theta) = \mathsf{E}[W] - \theta$.
We want to find θ^* s.t. $\bar{f}(\theta^*) = 0$.

Application of Robbins-Monro Algorithm

$$\begin{aligned}
\theta_{n+1} &= \theta_n + \frac{1}{n+1}(W_{n+1} - \theta_n) \\
&= \frac{n}{n+1}\theta_n + \frac{1}{n+1}W_{n+1} \\
&= \frac{1}{n+1}\sum_{k=1}^{n+1} W_k \qquad \Rightarrow \text{This is Monte-Carlo!}
\end{aligned}$$

Many RL challenges are SA Problems Too

In Monte-Carlo, we want to solve $\mathrm{E}[W - \theta] = 0$.

- W is new data / sample,
- θ_n is an old estimation,
- θ_{n+1} is the new estimation: $\theta_{n+1} = \theta_n + \alpha_{n+1} *$ *observed difference*.

Many RL challenges are SA Problems Too

In Monte-Carlo, we want to solve $\mathsf{E}[W - \theta] = 0$.

- W is new data / sample,
- θ_n is an old estimation,
- θ_{n+1} is the new estimation: $\theta_{n+1} = \theta_n + \alpha_{n+1} *$ *observed difference*.

Many RL algorithms rely on a temporal difference (TD) term of the same form. For example, for Q-Learning:

- New data / sample $= c(X_n, U_n) + \beta \min_u [Q^n(X_{n+1}, u)]$,
- Old estimation $= Q^n(X_n, U_n)$.
- New estimation $= Q^{n+1}(X_n, U_n)$.

We will develop more on this in the section about Q-Learning.

Zap Stochastic Approximation

Difficulties of Stochastic Approximation

Reminder: we search the solution θ^* to

$$\bar{f}(\theta) := \mathsf{E}[f(\theta, W)] = 0\,, \qquad \theta \in \mathbb{R}^d\,, \bar{f} : \mathbb{R}^d \to \mathbb{R}^d$$

Difficulties of Stochastic Approximation

Reminder: we search the solution θ^* to

$$\bar{f}(\theta) := \mathsf{E}[f(\theta, W)] = 0\,, \qquad \theta \in \mathbb{R}^d\,, \bar{f} : \mathbb{R}^d \to \mathbb{R}^d$$

What makes this hard?

Difficulties of Stochastic Approximation

Reminder: we search the solution θ^* to

$$\bar{f}(\theta) := \mathsf{E}[f(\theta, W)] = 0\,, \qquad \theta \in \mathbb{R}^d\,, \bar{f} : \mathbb{R}^d \to \mathbb{R}^d$$

What makes this hard?

1. The distribution of the random variable W may not be known.
2. Computation of the expectation may be expensive: root finding requires multiple evaluations of the expectation for different θ.

Difficulties of Stochastic Approximation

Reminder: we search the solution θ^* to

$$\bar{f}(\theta) := \mathsf{E}[f(\theta, W)] = 0\,, \qquad \theta \in \mathbb{R}^d\,, \bar{f} : \mathbb{R}^d \to \mathbb{R}^d$$

What makes this hard?

1. The distribution of the random variable W may not be known.
2. Computation of the expectation may be expensive: root finding requires multiple evaluations of the expectation for different θ.
3. The recursive algorithms we come up with are often slow, and their variance may be infinite. We will see that it is typically the case for Q-Learning, unfortunately.

Convergence

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

Robbins-Monro Algorithm (reminder): $\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, W_{n+1})$.

Convergence

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

Robbins-Monro Algorithm (reminder): $\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, W_{n+1})$.

Analysis: θ^* : *stationary point* of the ODE $\frac{d}{dt}x(t) = \bar{f}(x(t))$

SA is a noisy Euler approximation:

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

Convergence

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

Robbins-Monro Algorithm (reminder): $\theta_{n+1} = \theta_n + \alpha_{n+1} f(\theta_n, W_{n+1})$.

Analysis: θ^* : *stationary point* of the ODE $\frac{d}{dt}x(t) = \bar{f}(x(t))$

SA is a noisy Euler approximation:

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

Stability of the ODE $\implies \lim_{n\to\infty} \theta_n = \theta^*$.

Performance Criteria

SA recursion:

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

Error sequence:

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

Performance Criteria

SA recursion:

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

Error sequence:

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

Two standard approaches to evaluate performance,

1. Finite-n bound:

$$\mathsf{P}\{\|\tilde{\theta}_n\| \geq \varepsilon\} \leq ?$$

2. Asymptotic covariance (CLT):

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

Performance Criteria

SA recursion:

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

Error sequence:

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

Two standard approaches to evaluate performance,

1. Finite-n bound:
$$\mathsf{P}\{\|\tilde{\theta}_n\| \geq \varepsilon\} \leq ?$$
2. Asymptotic covariance (CLT):

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

Asymptotic Covariance

$\Sigma = \lim_{n\to\infty} \Sigma_n = \lim_{n\to\infty} n\mathsf{E}\big[\tilde\theta_n \tilde\theta_n^{\mathsf{T}}\big]$

SA recursion:

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

Asymptotic Covariance

$\Sigma = \lim_{n\to\infty} \Sigma_n = \lim_{n\to\infty} n\mathsf{E}\big[\tilde\theta_n \tilde\theta_n^{\mathsf{T}}\big]$

Linearized SA recursion for the error sequence $\{\tilde\theta_n\}$:

$$\tilde\theta_{n+1} \approx \tilde\theta_n + \frac{1}{n}\Big\{A\tilde\theta_n + \Delta_{n+1}\Big\}$$

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

Asymptotic Covariance

$\Sigma = \lim_{n\to\infty} \Sigma_n = \lim_{n\to\infty} n\mathsf{E}\big[\tilde{\theta}_n \tilde{\theta}_n^{T}\big]$

Scaled, linearized SA recursion for the error sequence:

$$\sqrt{n+1}\tilde{\theta}_{n+1} \approx \sqrt{n}\tilde{\theta}_n + \tfrac{1}{n}\Big\{(A + \tfrac{1}{2}I)\sqrt{n}\tilde{\theta}_n\Big\} + \frac{1}{\sqrt{n}}\Delta_{n+1}$$

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

$$\sqrt{n+1} \approx \sqrt{n} + \tfrac{1}{2\sqrt{n}}$$

Asymptotic Covariance

$\Sigma = \lim_{n\to\infty} \Sigma_n = \lim_{n\to\infty} n\mathsf{E}\big[\tilde{\theta}_n \tilde{\theta}_n^{\intercal}\big]$

SA recursion for $\{\Sigma_n\}$:

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

$$A = \frac{d}{d\theta}\bar{f}\,(\theta^*)$$
$$\Sigma_\Delta = \mathsf{E}[\Delta_{n+1}\Delta_{n+1}^{\intercal}]$$

Asymptotic Covariance

$\Sigma = \lim_{n\to\infty} \Sigma_n = \lim_{n\to\infty} n\mathsf{E}\big[\tilde\theta_n \tilde\theta_n^{\intercal}\big]$

SA recursion for $\{\Sigma_n\}$:

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

$$A = \tfrac{d}{d\theta}\bar f\,(\theta^*)$$
$$\Sigma_\Delta = \mathsf{E}[\Delta_{n+1}\Delta_{n+1}^{\intercal}]$$

Asymptotic Variance Theory

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

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

Optimal Asymptotic Covariance

Basis of Ruppert's Stochastic Newton Raphson, and Polyak-Ruppert Averaging

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

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

Optimal Asymptotic Covariance

Basis of Ruppert's Stochastic Newton Raphson, and Polyak-Ruppert Averaging

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

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

Assume it converges, and linearize:

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

Optimal Asymptotic Covariance

Basis of Ruppert's Stochastic Newton Raphson, and Polyak-Ruppert Averaging

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

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

Assume it converges, and linearize:

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

Asymptotic Variance Theory

- If $\text{Re}\,\lambda(GA) \geq -\frac{1}{2}$ for some eigenvalue then Σ^G is (typically) infinite
- If $\text{Re}\,\lambda(GA) < -\frac{1}{2}$ for all, Σ^G solves the Lyapunov equation:

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

Optimal Asymptotic Covariance

Basis of Ruppert's Stochastic Newton Raphson, and Polyak-Ruppert Averaging

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

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

Assume it converges, and linearize:

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

Optimal Matrix Gain: $G^* := -A^{-1}$

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

$$\boxed{0 = (GA + \tfrac{1}{2}I)\Sigma^G + \Sigma^G (GA + \tfrac{1}{2}I)^T + G\Sigma_\Delta G^T}$$

Optimal Variance and Stochastic Newton Raphson (SNR)

$\bar{f}(\theta) = A\theta - b \qquad \frac{\partial}{\partial\theta}(\bar{f}(\theta)) = A$

Stochastic Newton Raphson: Matrix gain algorithm with $G_n \approx G^* = -A^{-1}$:

SNR Algorithm:

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

$$G_n^{-1} = -\frac{1}{n+1}\sum_{k=1}^{n+1} A_k \qquad A_{n+1} = \frac{d}{d\theta} f(\theta_n, W_{n+1})$$

Optimal Variance and Stochastic Newton Raphson (SNR)

$\bar{f}(\theta) = A\theta - b \qquad \frac{\partial}{\partial\theta}(\bar{f}(\theta)) = A$

Stochastic Newton Raphson: Matrix gain algorithm with $G_n \approx G^* = -A^{-1}$:

SNR Algorithm:

$$\theta_{n+1} = \theta_n + \alpha_{n+1}(-\widehat{A}_{n+1})^{-1} f(\theta_n, W_{n+1})$$
$$\widehat{A}_{n+1} = \widehat{A}_n + \alpha_{n+1}(A_{n+1} - \widehat{A}_n)$$

Optimal Variance and Zap-SNR

$A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$ is a function of θ

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

$$\text{Requires} \quad \widehat{A}_{n+1} \approx A(\theta_n) := \frac{d}{d\theta}\bar{f}\,(\theta_n)$$

Optimal Variance and Zap-SNR

$A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$ is a function of θ

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

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

$$\widehat{A}_{n+1} = \widehat{A}_n + \gamma_{n+1}(A_{n+1} - \widehat{A}_n), \qquad A_{n+1} = \frac{d}{d\theta} f(\theta_n, W_{n+1})$$

Optimal Variance and Zap-SNR

$A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$ is a function of θ

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

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

$$\widehat{A}_{n+1} = \widehat{A}_n + \gamma_{n+1}(A_{n+1} - \widehat{A}_n), \qquad A_{n+1} = \frac{d}{d\theta} f(\theta_n, W_{n+1})$$

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

Optimal Variance and Zap-SNR

$A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$ is a function of θ

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

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

$$\widehat{A}_{n+1} = \widehat{A}_n + \gamma_{n+1}(A_{n+1} - \widehat{A}_n), \qquad A_{n+1} = \frac{d}{d\theta} f(\theta_n, W_{n+1})$$

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

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

Optimal Variance and Zap-SNR

$A(\theta) = \frac{\partial}{\partial\theta}\bar{f}(\theta)$ is a function of θ

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

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

$$\widehat{A}_{n+1} = \widehat{A}_n + \gamma_{n+1}(A_{n+1} - \widehat{A}_n), \qquad A_{n+1} = \frac{d}{d\theta} f(\theta_n, W_{n+1})$$

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

ODE for Zap-SNR

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

Application to Q-Learning

Stochastic Optimal Control

MDP Model

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

- For all states x and sets A,

$$\mathsf{P}\{X_{n+1} \in A \mid X_n = x,\ U_n = u, \text{and prior history}\} = P_u(x, A)$$

- $c \colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$ is a cost function
- $\beta < 1$ a discount factor

Stochastic Optimal Control

MDP Model

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

- For all states x and sets A,

$$\mathsf{P}\{X_{n+1} \in A \mid X_n = x,\ U_n = u, \text{and prior history}\} = P_u(x, A)$$

- $c\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$ is a cost function
- $\beta < 1$ a discount factor

Q-function:

$$Q^*(x,u) = \min_{\boldsymbol{U}} \sum_{n=0}^{\infty} \beta^n \mathsf{E}[c(X_n, U_n) \mid X_0 = x, U_0 = u]$$

Stochastic Optimal Control

MDP Model

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

- For all states x and sets A,

$$\mathsf{P}\{X_{n+1} \in A \mid X_n = x,\ U_n = u, \text{and prior history}\} = P_u(x, A)$$

- $c\colon \mathsf{X} \times \mathsf{U} \to \mathbb{R}$ is a cost function
- $\beta < 1$ a discount factor

Q-function:

$$Q^*(x,u) = \min_{\boldsymbol{U}} \sum_{n=0}^{\infty} \beta^n \mathsf{E}[c(X_n, U_n) \mid X_0 = x, U_0 = u]$$

Bellman equation:

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

Stochastic Optimal Control Seen As an SA Problem

Problem

Find function Q^* that solves

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

Stochastic Optimal Control Seen As an SA Problem

Problem

Find function Q^* that solves

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

Q-learning

Given $\{Q^\theta : \theta \in \mathbb{R}^d\}$, find θ^* that solves

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

The family $\{Q^\theta\}$ and "*eligibility vectors*" $\{\zeta_n\}$, $\zeta_n \in \mathbb{R}^d$ are part of algorithm design.

Stochastic Optimal Control Seen As an SA Problem

Problem

Find function Q^* that solves

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

Q-learning

Given $\{Q^\theta : \theta \in \mathbb{R}^d\}$, find θ^* that solves

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

The family $\{Q^\theta\}$ and "*eligibility vectors*" $\{\zeta_n\}$, $\zeta_n \in \mathbb{R}^d$ are part of algorithm design.

Example: $\zeta_n = \nabla_\theta Q^\theta(X_n,U_n)$

Stochastic Optimal Control Seen As an SA Problem

This is Stochastic Approximation!

Q-learning

Given $\{Q^\theta : \theta \in \mathbb{R}^d\}$, find θ^* that solves

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

The family $\{Q^\theta\}$ and "*eligibility vectors*" $\{\zeta_n\}$, $\zeta_n \in \mathbb{R}^d$ are part of algorithm design.

Example: $\zeta_n = \nabla_\theta Q^\theta(X_n,U_n)$

Watkins' Q-learning ($=$ "Vanilla" Tabular Q-Learning)

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

Watkins' Q-learning (= "Vanilla" Tabular Q-Learning)

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

Watkin's algorithm is Stochastic Approximation

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

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

Watkins' Q-learning (= "Vanilla" Tabular Q-Learning)

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

Watkin's algorithm is Stochastic Approximation

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

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

Algorithm:

$$\theta_{n+1}=\theta_n+\alpha_{n+1}\big(c(X_n,U_n)+\beta\underline{Q}^{\theta^*}(X_{n+1})-Q^{\theta^*}(X_n,U_n)\big)\zeta_n$$

Watkins' Q-learning (= "Vanilla" Tabular Q-Learning)

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

Watkin's algorithm is Stochastic Approximation

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

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

Converges, but has infinite asymptotic variance if $\beta > \frac{1}{2}$:

$$\lambda_{\max}\big(\boldsymbol{A}(\theta^*)\big) > -\tfrac{1}{2}$$

[Devraj & Meyn, 2017]

Watkins' Q-learning (= "Vanilla" Tabular Q-Learning)

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

Watkin's algorithm is Stochastic Approximation

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

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

Convergence rate for $\beta > \frac{1}{2}$:

$$\mathcal{O}(1/n^{1-\beta})$$

[Devraj & Meyn, 2017]

Watkins' Q-learning (= "Vanilla" Tabular Q-Learning)

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

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

Watkin's algorithm is Stochastic Approximation

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

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

Convergence rate for $\beta > \frac{1}{2}$:

$$\mathcal{O}(1/n^{1-\beta})$$

[Devraj & Meyn, 2017]

Linear Parametrization of Q-Learning

Definition

$Q^\theta(x,u) = \theta^T \psi(x,u)$, where:

- $\theta \in \mathbb{R}^d$ denotes the parameter vector,
- $\psi(x,u)$ represents the features of (x,u).

Linear Parametrization of Q-Learning

Definition

$Q^\theta(x,u) = \theta^T \psi(x,u)$, where:

- $\theta \in \mathbb{R}^d$ denotes the parameter vector,
- $\psi(x,u)$ represents the features of (x,u).

Particular Case: Tabular Q-Learning

- $\psi_i(x,u) = \mathbb{I}(x = x^i, u = u^i)$,
- (x^i, u^i) enumerate all state-action pairs,
- $1 \leq i \leq d$, where $d = |\text{states}| * |\text{actions}|$.

Q(λ) Algorithm

1. $d_{n+1} = c(X_n, U_n) + \beta \underline{Q}^{\theta_n}(X_{n+1}) - Q^{\theta_n}(X_n, U_n)$
2. $\theta_{n+1} = \theta_n + \alpha_{n+1}\zeta_n d_{n+1}$
3. $\zeta_{n+1} = \lambda\beta\zeta_n + \psi(X_{n+1}, U_{n+1})$

Zap-Q(λ) Algorithm

1. $d_{n+1} = c(X_n, U_n) + \beta \underline{Q}^{\theta_n}(X_{n+1}) - Q^{\theta_n}(X_n, U_n)$
2. $A_{n+1} = \zeta_n [\beta \psi(X_{n+1}, \phi_n(X_{n+1})) - \psi(X_n, U_n)]^T$
3. $\widehat{A}_{n+1} = \widehat{A}_n + \gamma_{n+1}[A_{n+1} - \widehat{A}_n]$
4. $\theta_{n+1} = \theta_n + \alpha_{n+1} \widehat{A}_{n+1}^{-1} \zeta_n d_{n+1}$
5. $\zeta_{n+1} = \lambda \beta \zeta_n + \psi(X_{n+1}, U_{n+1})$

Zap-Q(λ) Algorithm: Issues and Possible Solutions

Work in Progress...

Issue 1: $\widehat{A}_{n+1}$ is proven to be eventually invertible, but is generally not invertible during the early stages of the algorithm.

$\Rightarrow$ Use Moore-Penrose pseudoinverse $\widehat{A}_{n+1}^{+}$.

Zap-Q(λ) Algorithm: Issues and Possible Solutions

Work in Progress...

Issue 1: $\widehat{A}_{n+1}$ is proven to be eventually invertible, but is generally not invertible during the early stages of the algorithm.

$\Rightarrow$ Use Moore-Penrose pseudoinverse $\widehat{A}^{+}_{n+1}$.

Issue 2: Computing $\widehat{A}^{-1}_{n+1}$ (or $\widehat{A}^{+}_{n+1}$) is expensive.

$\Rightarrow$ In fact we do not need $\widehat{A}^{+}_{n+1}$ itself but only $\widehat{A}^{+}_{n+1}\zeta_n$. This can be done by solving a least squares problem: find X that minimizes $\|\widehat{A}_{n+1}X - \zeta_n\|_2$. Still expensive...

$\Rightarrow$ Since $\widehat{A}_{n+1}$ is updated by adding a matrix of rank 1 at each step, it can be computed cheaply by Sherman-Morrison-Woodbury formula.

Conclusion

Conclusion

Take-aways:

- *Reinforcement Learning is not just cursed by dimension, but also by variance!*
- RL algorithms in their raw form are NO GOOD without careful gain selection.

Conclusion

Take-aways:

- *Reinforcement Learning is not just cursed by dimension, but also by variance!*
- RL algorithms in their raw form are NO GOOD without careful gain selection.

Current/future works:

- Implementation in the Stable-Baselines framework.
- Q-learning with function-approximation: obtain conditions for a stable algorithm in a general setting.

This Presentation

- A. M. Devraj and S. P. Meyn, *Zap Q-learning.* *Advances in Neural Information Processing Systems (NIPS).* Dec. 2017.
- A. M. Devraj and S. P. Meyn, *Fastest convergence for Q-learning.* Available on *ArXiv*. Jul. 2017.
- A. M. Devraj, A. Bušić, and S. Meyn. *Optimal Matrix Momentum Stochastic Approximation and Applications to Q-learning*. *ArXiv e-prints*, Feb. 2019.
- S. Chen, A. M. Devraj, A. Bušić, and S. Meyn. *Zap Q-learning for Optimal Stopping Time Problems*. *ArXiv e-prints*, Apr. 2019.
- S. Chen, A. M. Devraj, A. Bušić, and S. Meyn. *Zap Q-learning with Nonlinear Function Approximation*. *ArXiv e-prints*, Oct. 2019.

Selected References I

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

[2] A. M. Devraj, A. Bušić and S. P. Meyn. *Zap Meets Momentum: Stochastic Approximation Algorithms with Optimal Convergence Rate. ArXiv* , September 2018.

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

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

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

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

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

Selected References II

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

[9] 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.

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

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

[12] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[13] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, 1983.

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

Selected References III

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

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

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

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

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

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

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

Selected References IV

[23] 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.

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

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

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

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

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

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