

A Control Perspective of Stochastic Approximation Methods in Machine Learning

Bin Hu

University of Illinois at Urbana-Champaign

Joint work with P. Seiler, L. Lessard, A. Rantzer, S. Wright, and U. Syed

ACC Workshop 2019

Interplay between Control, Optimization, and Machine Learning

Artificial Intelligence Revolution



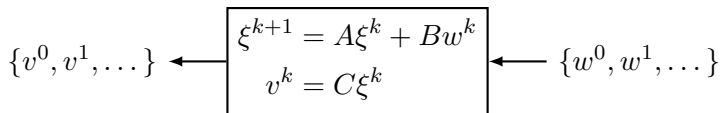
Google
Translate

Break through language barriers

Machine Learning

- Supervised learning
- Unsupervised learning
- Reinforcement learning
- Etc

Tools in Control: Linear Time-Invariant Systems



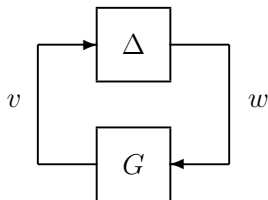
- Given ξ^0 , the above LTI system model maps any input sequence w to a uniquely determined output sequence v :

$$\xi^k = (A)^k \xi^0 + \sum_{t=0}^{k-1} (A)^{k-1-t} B w^t$$

$$v^k = C(A)^k \xi^0 + \sum_{t=0}^{k-1} C(A)^{k-1-t} B w^t$$

- If the spectral radius of A is smaller than 1, then $(A)^k$ converges to a zero matrix, and bounded w leads to bounded state/output.
- Many extensions, e.g. jump systems $\xi^{k+1} = A_{i_k} \xi^k + B_{i_k} w^k$.

Tools in Control: Robust Control Theory



1. Approximate the true system as “a simple system + a perturbation”
2. Δ can be troublesome/nonlinear/stochastic element, uncertainty, or fault
3. Rich control literature including standard textbooks
 - Zhou, Doyle, Glover, “Robust and optimal control,” 1996
4. The Lur’e problem (1944): G is LTI, and Δ is a nonlinearity
5. Many tools: small gain, passivity, dissipativity, Zames-Falb multipliers, etc
6. The integral quadratic constraint (IQC) framework [Megretski, Rantzer, 1997] provides a unified analysis for “LTI G + troublesome Δ ”
7. Recently, the LMI-based analysis has been extended for more general G (linear parameter-varying or Markov jump linear systems)

Outline: Unified Analysis Tools

- **Analyzing Stochastic Methods in Supervised Learning**
- Analyzing Temporal Difference in Reinforcement Learning
- Future Directions

Empirical Risk Minimization (ERM)

- ERM is a general paradigm for supervised learning:

$$\min_{x \in \mathbb{R}^p} g(x) := \frac{1}{n} \sum_{i=1}^n f_i(x)$$

where f_i is smooth and convex, and g is strongly-convex.

- Ridge regression:

$$f_i(x) = (a_i^T x - b_i)^2 + \frac{\lambda}{2} \|x\|^2$$

- ℓ_2 -regularized logistic regression:

$$f_i(x) = \log(1 + e^{-b_i a_i^T x}) + \frac{\lambda}{2} \|x\|^2$$

- ℓ_2 -regularized loss minimization with loss function $l_i(x)$:

$$f_i(x) = l_i(x) + \frac{\lambda}{2} \|x\|^2$$

- Many other possibilities: SVM, Lasso, elastic net, PCA, etc

- ERM is used in supervised learning to select a “best” candidate (parameterized by x) from a class of models.

Methods for ERM

Let g be L -smooth and m -strongly convex.

- Given $\alpha = \frac{1}{L}$, the iteration complexity of the full gradient (FG) method is $\tilde{\mathcal{O}}\left(\frac{nL}{m}\right)$.

$$x^{k+1} = x^k - \alpha \nabla g(x^k) = x^k - \frac{\alpha}{n} \sum_{i=1}^n \nabla f_i(x^k)$$

- Nesterov's method improves the iteration complexity to $\tilde{\mathcal{O}}\left(n\sqrt{\frac{L}{m}}\right)$. The iteration still scales linearly with n .
- Stochastic Gradient (SG) Method [Robbins, Monro, 1951] uses the iteration rule: $x^{k+1} = x^k - \alpha \nabla f_{i_k}(x^k)$. At each k , i_k is sampled uniformly from the set $\mathcal{N} := \{1, 2, \dots, n\}$.
- With well-chosen constant step size, SG linearly converges to some tolerance of the optimum.

Stochastic Average Gradient (SAG) Method

SAG [Roux et al., 2012; Schmidt et al., 2013] uses the iteration rule:

$$x^{k+1} = x^k - \frac{\alpha}{n} \sum_{i=1}^n y_i^{k+1}$$

where at each iteration a random i_k is drawn and

$$y_i^{k+1} := \begin{cases} \nabla f_i(x^k) & \text{if } i = i_k \\ y_i^k & \text{otherwise} \end{cases}$$

Let $\alpha = \frac{1}{16L}$. Then $\mathbb{E}[g(x^k) - g(x^*)] \leq C_0 \left(1 - \min\left\{\frac{1}{8n}, \frac{m}{16L}\right\}\right)^k$.

The iteration complexity becomes $\tilde{\mathcal{O}}\left(n + \frac{L}{m}\right)$, instead of $\tilde{\mathcal{O}}\left(\frac{nL}{m}\right)$.

Stochastic Finite-Sum Methods

- Now there is a large family of variance-reduction methods, e.g. SAGA [Defazio et al., 2014a], Finito [Defazio et al. 2014b], SDCA [Shalev-Shwartz and Zhang, 2013; Shalev-Shwartz, 2016], SVRG [Johnson and Zhang, 2013], and Katyusha [Allen-Zhu, 2016]. Analysis is done in a case-by-case manner.
- For example, SAGA uses

$$x^{k+1} = x^k - \alpha \left(\nabla f_{i_k}(x^k) - y_{i_k}^k + \frac{1}{n} \sum_{i=1}^n y_i^k \right)$$
$$y_i^{k+1} = \begin{cases} \nabla f_i(x^k) & \text{if } i = i_k \\ y_i^k & \text{otherwise} \end{cases}$$

- SAGA and SAG look very similar. But the analysis of SAG is much more difficult! Why?

Stochastic Finite-Sum Methods

Function class

- f_i convex, g strongly-convex
- f_i strongly-convex, g strongly-convex
- many other possibilities

Method

- SAGA
- SAG
- SDCA
- many other possibilities

Bound

- $\mathbb{E}[g(x^k) - g(x^*)] \leq c_1 \rho^k$
- $\mathbb{E}\|x^k - x^*\|^2 \leq c_2 \rho^k$
- Other forms

function class + method \implies bound

Objectives and Ideas

We want to develop a simple analysis routine.

- **Automate** rate analysis of stochastic finite-sum methods.

function class + method \implies bound

- Use numerical semidefinite programs to support search for analytical proofs.
- The unified analysis is based on robust control theory.
- The analysis can be viewed as stochastic counterparts for the analysis in [Lessard, Recht, Packard, 2016].

Finite-Sum Methods as Jump Systems

[Hu, Seiler, Rantzer, COLT 2017]: Finite-sum methods, e.g. SAGA, SAG, Finito, and SDCA, can be modeled as jump systems.

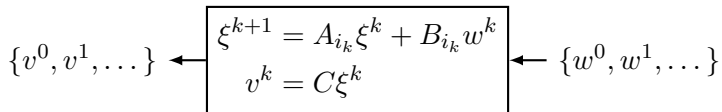
$$\xi^{k+1} = A_{i_k} \xi^k + B_{i_k} w^k$$

$$v^k = C \xi_k$$

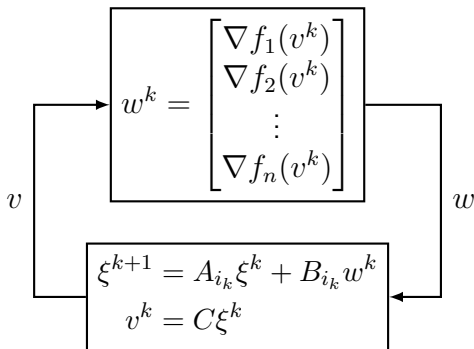
$$w^k = \begin{bmatrix} \nabla f_1(v^k) \\ \nabla f_2(v^k) \\ \vdots \\ \nabla f_n(v^k) \end{bmatrix}$$

- At each step k , the jump parameter i_k is a random variable taking value in a finite set $\mathcal{N} = \{1, \dots, n\}$.
- A_{i_k} , and B_{i_k} are functions of i_k . When $i_k = i \in \mathcal{N}$, clearly we have $A_{i_k} = A_i$ and $B_{i_k} = B_i$.

Feedback View of Stochastic Methods



Stochastic finite-sum methods can be represented as:



Finite-Sum Methods as Jump Systems

Choose $A_{i_k} = \tilde{A}_{i_k} \otimes I_p$, $B_{i_k} = \tilde{B}_{i_k} \otimes I_p$, and $C = \tilde{C} \otimes I_p$ where

Method	\tilde{A}_{i_k}	\tilde{B}_{i_k}	\tilde{C}
SAGA	$\begin{bmatrix} I_n - e_{i_k} e_{i_k}^T & \tilde{0} \\ -\frac{\alpha}{n}(e - n e_{i_k})^T & 1 \end{bmatrix}$	$\begin{bmatrix} e_{i_k} e_{i_k}^T \\ -\alpha e_{i_k}^T \end{bmatrix}$	$\begin{bmatrix} \tilde{0}^T & 1 \end{bmatrix}$
SAG	$\begin{bmatrix} I_n - e_{i_k} e_{i_k}^T & \tilde{0} \\ -\frac{\alpha}{n}(e - e_{i_k})^T & 1 \end{bmatrix}$	$\begin{bmatrix} e_{i_k} e_{i_k}^T \\ -\frac{\alpha}{n} e_{i_k}^T \end{bmatrix}$	$\begin{bmatrix} \tilde{0}^T & 1 \end{bmatrix}$
Finito	$\begin{bmatrix} I_n - e_{i_k} e_{i_k}^T & \tilde{0} \\ -\alpha(e_{i_k} e^T) & I_n - e_{i_k} (e_{i_k}^T - \frac{1}{n} e^T) \end{bmatrix}$	$\begin{bmatrix} e_{i_k} e_{i_k}^T \\ \tilde{0} \tilde{0}^T \end{bmatrix}$	$\begin{bmatrix} -\alpha e^T & \frac{1}{n} e^T \end{bmatrix}$
SDCA	$I_n - \alpha m n e_{i_k} e_{i_k}^T$	$-\alpha m n e_{i_k} e_{i_k}^T$	$\frac{1}{m n} e^T$

Sparsity of B_{i_k} captures the low cost of stochastic methods.

Ref: Hu, Seiler, Rantzer, "A Unified Analysis of Stochastic Optimization Methods Using Jump System Theory and Quadratic Constraints," COLT 2017.

Example: Jump System Formulation for SAGA

Let e_i denote the p -dimensional vector whose entries are all 0 except the i -th entry which is 1. The iteration rule

$$y_i^{k+1} = \begin{cases} \nabla f_i(x^k) & \text{if } i = i_k \\ y_i^k & \text{otherwise} \end{cases}$$

is equivalent to

$$y^{k+1} = ((I_n - e_{i_k}^T e_{i_k}) \otimes I_p) y^k + (e_{i_k}^T e_{i_k} \otimes I_p) w^k$$

where y^k and w^k are given as

$$y^k = \begin{bmatrix} y_1^k \\ \vdots \\ y_n^k \end{bmatrix}, \quad w^k = \begin{bmatrix} \nabla f_1(x^k) \\ \vdots \\ \nabla f_n(x^k) \end{bmatrix}$$

Example: Jump System Formulation for SAGA

Eventually we can write SAGA as

$$\begin{bmatrix} y^{k+1} \\ x^{k+1} \end{bmatrix} = \begin{bmatrix} (I_n - e_{i_k} e_{i_k}^T) \otimes I_p & \tilde{0} \otimes I_p \\ -\frac{\alpha}{n} (e - n e_{i_k})^T \otimes I_p & I_p \end{bmatrix} \begin{bmatrix} y^k \\ x^k \end{bmatrix} + \begin{bmatrix} (e_{i_k} e_{i_k}^T) \otimes I_p \\ (-\alpha e_{i_k}^T) \otimes I_p \end{bmatrix} w^k$$

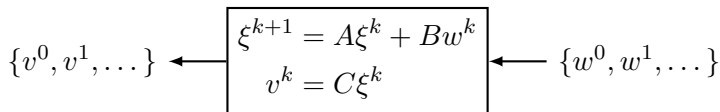
$$v^k = \begin{bmatrix} \tilde{0}^T \otimes I_p & I_p \end{bmatrix} \begin{bmatrix} y^k \\ x^k \end{bmatrix}$$

$$w^k = \begin{bmatrix} \nabla f_1(v^k) \\ \vdots \\ \nabla f_n(v^k) \end{bmatrix}$$

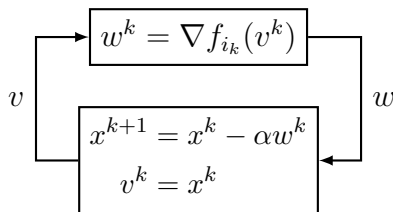
which is exactly in the form of the general jump system model with

$$\xi^k = \begin{bmatrix} y^k \\ x^k \end{bmatrix}.$$

Feedback View of SG and SVRG-like Methods

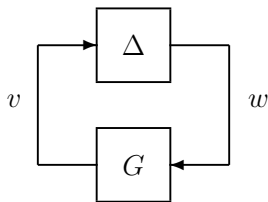


Some finite-sum methods require using jump nonlinearity Δ_{i_k} . In [Hu, Seiler, Lessard, 2017], we model SG ($x^{k+1} = x^k - \alpha \nabla f_{i_k}(x^k)$) as:



Similar models for SVRG and Katyusha in [Hu, Wright, Lessard, 2018]

Summary of Results



Optimization methods are feedback dynamical systems!

- (Hu, Seiler, Rantzer, COLT2017): Convergence rate guarantees of SAGA, SDCA, and Finito with or without individual convexity
- (Hu, Lessard, ICML2017): Unifying linear/sublinear rate analyses for discrete-time/continuous-time algorithms
- (Hu, Seiler, Lessard, 2017): Convergence rates of inexact SGD
- (Hu, Wright, Lessard, ICML2018): Convergence rate guarantees of SVRG and Katyusha with various parameter choices

Dissipativity Theory

In 1972, J.C. Willems pioneered dissipativity theory, a physics-inspired framework to analyze dynamical systems. Originally it was developed for LTI systems but now extended for many other types of systems.

Consider a jump system with i_k uniformly sampled in an i.i.d manner.

$$\xi^{k+1} = A_{i_k} \xi^k + B_{i_k} w^k$$

The dissipation inequality looks like:

$$\underbrace{\mathbb{E}V(\xi^{k+1}) - \mathbb{E}V(\xi^k)}_{\text{averaged increase in stored energy}} \leq \underbrace{\mathbb{E}S(\xi^k, w^k)}_{\text{power supplied}}$$

- $V(\xi)$ is called the **storage function**.
- $S(\xi, w)$ is called the **supply rate**.

Dissipativity Theory

The exponential dissipation inequality is useful for rate analysis:

$$\mathbb{E}V(\xi^{k+1}) - \rho^2 \mathbb{E}V(\xi^k) \leq \mathbb{E}S(\xi^k, w^k)$$

- A fraction $(1 - \rho^2)$ of the internal energy dissipates at every k .
- If we know $\mathbb{E}S \leq 0$ in advance, then we conclude that $V(\xi)$ is a Lyapunov function (this works for SAGA-like methods):

$$\mathbb{E}V(\xi^{k+1}) \leq \rho^2 \mathbb{E}V(\xi^k)$$

- If we know $\mathbb{E}S \leq M$ in advance, then we conclude a linear convergence to a ball (this works for SG):

$$\mathbb{E}V(\xi^{k+1}) \leq \rho^{2k} \mathbb{E}V(\xi^0) + \frac{M}{1 - \rho^2}$$

- SVRG and Katyusha require a supply rate $\mathbb{E}S \leq c(\xi^0)$.

Dissipation Inequality for Jump Systems

Theorem

Consider the jump system $\xi^{k+1} = A_{i_k} \xi^k + B_{i_k} w^k$ and the quadratic supply rates with symmetric X_j . If $\exists \lambda_j \geq 0$ and a matrix $P \succeq 0$ s.t.

$$\begin{bmatrix} \frac{1}{n} \sum_{i=1}^n A_i^T P A_i - \rho^2 P & \frac{1}{n} \sum_{i=1}^n A_i^T P B_i \\ \frac{1}{n} \sum_{i=1}^n B_i^T P A_i & \frac{1}{n} \sum_{i=1}^n B_i^T P B_i \end{bmatrix} - \sum_{j=1}^J \lambda_j X_j \preceq 0$$

then $\mathbb{E}V^{k+1} - \rho^2 \mathbb{E}V^k \leq \mathbb{E}S^k$ with $V(\xi) = \xi^T P \xi$, $S = \sum_{j=1}^J \lambda_j S_j$.

The proof is based on the following relationship:

$$\begin{aligned} \mathbb{E}[V^{k+1} | \mathcal{F}_{k-1}] &= \mathbb{E}[(\xi^{k+1})^T P \xi^{k+1} | \mathcal{F}_{k-1}] \\ &= \sum_{i=1}^n \mathbb{P}(i_k = i) [A_i \xi^k + B_i w^k]^T P [A_i \xi^k + B_i w^k] \\ &= \begin{bmatrix} \xi^k \\ w^k \end{bmatrix}^T \begin{bmatrix} \frac{1}{n} \sum_{i=1}^n A_i^T P A_i & \frac{1}{n} \sum_{i=1}^n A_i^T P B_i \\ \frac{1}{n} \sum_{i=1}^n B_i^T P A_i & \frac{1}{n} \sum_{i=1}^n B_i^T P B_i \end{bmatrix} \begin{bmatrix} \xi^k \\ w^k \end{bmatrix} \end{aligned}$$

Quadratic Constraints

- For example, rewrite the full gradient method as:

$$\underbrace{(x^{k+1} - x^*)}_{\xi^{k+1}} = \underbrace{(x^k - x^*)}_{\xi^k} - \alpha \underbrace{\nabla g(x^k)}_{w^k}$$

- If g is L -smooth and m -strongly convex, then by co-coercivity:

$$\begin{bmatrix} x - x^* \\ \nabla g(x) \end{bmatrix}^\top \underbrace{\begin{bmatrix} 2mLI_p & -(m+L)I_p \\ -(m+L)I_p & 2I_p \end{bmatrix}}_X \begin{bmatrix} x - x^* \\ \nabla g(x) \end{bmatrix} \leq 0$$

- To construct the dissipation inequality, one only needs to solve the small semidefinite program

$$\left(\begin{bmatrix} (1-\rho^2)p & -\alpha p \\ -\alpha p & \alpha^2 p \end{bmatrix} + \begin{bmatrix} -2mL & m+L \\ m+L & -2 \end{bmatrix} \right) \otimes I_p \leq 0$$

- Choose (α, ρ, p) to be $(\frac{1}{L}, 1 - \frac{m}{L}, L^2)$ or $(\frac{2}{L+m}, \frac{L-m}{L+m}, \frac{1}{2}(L+m)^2)$ to recover the standard rate results

Summary of Our Approach

1. Use the algorithm matrices (A_i, B_i, C) and the function properties to formulate

$$\begin{bmatrix} \frac{1}{n} \sum_{i=1}^n A_i^T P A_i - \rho^2 P & \frac{1}{n} \sum_{i=1}^n A_i^T P B_i \\ \frac{1}{n} \sum_{i=1}^n B_i^T P A_i & \frac{1}{n} \sum_{i=1}^n B_i^T P B_i \end{bmatrix} - \sum_{j=1}^J \lambda_j X_j \preceq 0$$

2. If the LMI is feasible, then $\mathbb{E}V(\xi^{k+1}) - \rho^2 \mathbb{E}V(\xi^k) \leq \mathbb{E}S(\xi^k, w^k)$
3. Convergence bounds follow from the dissipation inequality and the supply rate condition.
 - Ex: If $\mathbb{E}S \leq 0$, we have $\mathbb{E}V(\xi^k) \leq \rho^{2k} V(\xi^0)$
 - Lur'e-Postnikov Lyapunov function (SAG): $V(\xi) + g(x) - g(x^*)$
 - Cover SG, SVRG, Katyusha, SAGA, SDCA, Finito, and SAG
 - A unified understanding for acceleration and variance reduction

Outline: Unified Analysis Tools

- Analyzing Stochastic Methods in Supervised Learning
- **Analyzing Temporal Difference in Reinforcement Learning**
- Future Directions

High-level Idea: From IID to Markov Noises

- Temporal difference learning can be modeled as

$$\xi^{k+1} = \xi^k + \alpha(A_{i_k}\xi^k + b_{i_k})$$

where i_k is a Markov chain sampled from a finite state space.

- If i_k is IID, then the analysis is similar to the analysis of SG.
- The Markov nature of $\{i_k\}$ makes the analysis complicated. The finite sample bound for TD learning is obtained early this year [Srikant, Ying, COLT 2019]!
- However, the above system is exactly a Markov jump linear system, which has been well studied in the controls literature [Costa, Fragoso, Marques, 2006].

$$\xi^{k+1} = (I + \alpha A_{i_k})\xi^k + \alpha b_{i_k}$$

Policy Evaluation for Reinforcement Learning

- Reinforcement learning studies the control of Markov Decision Processes $(\mathcal{S}, \mathcal{A}, P, c, \gamma)$.
- For a fixed stationary policy, the value function is

$$V(i) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k c(s^k) \mid s^0 = i \right]$$

- Policy evaluation: Given a policy, TD learning aims to learn a linear approximation $V(i) \approx \phi(i)^\top \theta$ from the sampled trajectory.
- The standard TD method (or TD(0)) uses the update rule:

$$\theta^{k+1} = \theta^k - \alpha \phi(s^k) \left((\phi(s^k) - \gamma \phi(s^{k+1}))^\top \theta^k - r(s^k) \right).$$

- Suppose θ^* is the vector that solves the projected Bellman equation. Set $i_k = [(s^{k+1})^\top \quad (s^k)^\top]^\top$, $A_{i_k} = -\phi(s^k)(\phi(s^k) - \phi(s^{k+1}))^\top$, and $b_{i_k} = \phi(s^k) (r(s^k) - (\phi(s^k) - \phi(s^{k+1}))^\top \theta^*)$. Then the TD update is equivalent to $\theta^{k+1} - \theta^* = (I + \alpha A_{i_k}) (\theta^k - \theta^*) + \alpha b_{i_k}$.

Existing Results for TD Learning

- ODE technique: asymptotic convergence under IID assumption and diminishing stepsize
- Finite sample bounds under IID assumption: [Dalal, Szrnyi, Thoppe, Mannor, AAI2018], [Lakshminarayanan, Szepesvari, Alstats2018]
- Finite sample bounds under Markov assumption but with an extra projection step: [Bhandari, Russo, Singal, COLT2018]
- Finite sample bounds under Markov assumption for small stepsize: [Srikant, Ying, COLT2019]

$$\mathbb{E}\|\xi^k\|^2 \leq C_0\rho^{2k} + C_1,$$

where $\rho^2 = 1 - c\alpha + O(\alpha^2)$ for some c , $C_1 = O(\alpha)$, and C_0 depends how fast the Markov chain $\{i_k\}$ mixes.

Existing Results from MJLS Theory

- Consider a MJLS where $p_{ij} = \mathbb{P}(i_{k+1} = j | i_k = i)$ and $y^k = 1$.

$$\xi^{k+1} = H_{i_k} \xi^k + G_{i_k} y^k.$$

- Let us define $\mu^k = \mathbb{E} \xi^k$ and $Q^k = \mathbb{E} (\xi^k (\xi^k)^\top)$. We further set $q_i^k = \mathbb{E} (\xi^k \mathbf{1}_{\{i_k=i\}})$ and $Q_i^k = \mathbb{E} (\xi^k (\xi^k)^\top \mathbf{1}_{\{i_k=i\}})$. We have $\mu^k = \sum_{i=1}^n q_i^k$ and $Q^k = \sum_{i=1}^n Q_i^k$. We augment q_i^k and Q_i^k as

$$q^k = \begin{bmatrix} q_1^k \\ \vdots \\ q_n^k \end{bmatrix}, \quad Q^k = [Q_1^k \quad Q_2^k \quad \dots \quad Q_n^k].$$

Then we have

$$q_j^{k+1} = \sum_{i=1}^n p_{ij} (H_i q_i^k + G_i p_i^k),$$
$$Q_j^{k+1} = \sum_{i=1}^n p_{ij} \left(H_i Q_i^k H_i^\top + 2 \text{sym}(H_i q_i^k G_i^\top) + p_i^k G_i G_i^\top \right).$$

Existing Results from MJLS Theory

$$\begin{bmatrix} q^{k+1} \\ \text{vec}(Q^{k+1}) \end{bmatrix} = \begin{bmatrix} \mathcal{H}_{11} & 0 \\ \mathcal{H}_{21} & \mathcal{H}_{22} \end{bmatrix} \begin{bmatrix} q^k \\ \text{vec}(Q^k) \end{bmatrix} + \begin{bmatrix} u_q^k \\ u_Q^k \end{bmatrix}$$

$$\mathcal{H}_{11} = \begin{bmatrix} p_{11}H_1 & \dots & p_{n1}H_n \\ \vdots & \ddots & \vdots \\ p_{1n}H_1 & \dots & p_{nn}H_n \end{bmatrix}, \mathcal{H}_{22} = \begin{bmatrix} p_{11}H_1 \otimes H_1 & \dots & p_{n1}H_n \otimes H_n \\ \vdots & \ddots & \vdots \\ p_{1n}H_1 \otimes H_1 & \dots & p_{nn}H_n \otimes H_n \end{bmatrix}$$

$$\mathcal{H}_{21} = \begin{bmatrix} p_{11}(H_1 \otimes G_1 + G_1 \otimes H_1) & \dots & p_{n1}(H_n \otimes G_n + G_n \otimes H_n), \\ \vdots & \ddots & \vdots \\ p_{1n}(H_1 \otimes G_1 + G_1 \otimes H_1) & \dots & p_{nn}(H_n \otimes G_n + G_n \otimes H_n) \end{bmatrix},$$

$$u_q^k = \begin{bmatrix} p_{11}G_1 & \dots & p_{n1}G_n \\ \vdots & \ddots & \vdots \\ p_{1n}G_1 & \dots & p_{nn}G_n \end{bmatrix} \begin{bmatrix} p_1^k I_{n_\xi} \\ \vdots \\ p_n^k I_{n_\xi} \end{bmatrix},$$

$$u_Q^k = \begin{bmatrix} p_{11}G_1 \otimes G_1 & \dots & p_{n1}G_n \otimes G_n \\ \vdots & \ddots & \vdots \\ p_{1n}G_1 \otimes G_1 & \dots & p_{nn}G_n \otimes G_n \end{bmatrix} \begin{bmatrix} p_1^k I_{n_\xi^2} \\ \vdots \\ p_n^k I_{n_\xi^2} \end{bmatrix}.$$

Applying MJLS Theory to TD Learning

$$\begin{bmatrix} q^{k+1} \\ \text{vec}(Q^{k+1}) \end{bmatrix} = \begin{bmatrix} \mathcal{H}_{11} & 0 \\ \mathcal{H}_{21} & \mathcal{H}_{22} \end{bmatrix} \begin{bmatrix} q^k \\ \text{vec}(Q^k) \end{bmatrix} + \begin{bmatrix} u_q^k \\ u_Q^k \end{bmatrix}$$

Recall that $H_i = I + \alpha A_i$, and $G_i = b_i$ for TD learning.

1. We have $\mathcal{H}_{22} = (P^\top \otimes I_{n_\xi^2}) \text{diag}((I_{n_\xi} + \alpha A_i) \otimes (I_{n_\xi} + \alpha A_i))$ and

$$\lambda_{\max}(\mathcal{H}_{22}) = 1 + 2\lambda_{\max \text{ real}}(\bar{A})\alpha + O(\alpha^2).$$

where $\bar{A} = \sum_{i=1}^n p_i^\infty A_i$ and p^∞ is the unique stationary distribution of the Markov chain under the ergodicity assumption.

2. Suppose $\sigma(\mathcal{H}_{22}) < 1$. Assume $p^k \rightarrow p^\infty$. Then

$$q^\infty = \lim_{k \rightarrow \infty} q^k = \alpha(I - \mathcal{H}_{11})^{-1}((P^\top \text{diag}(p_i^\infty)) \otimes I_{n_\xi})b,$$

$$\text{vec}(Q^\infty) = \lim_{k \rightarrow \infty} \text{vec}(Q^k) = O(\alpha)$$

3. Given the geometric ergodicity, i.e. $\|p^k - p^\infty\| \leq C\tilde{\rho}^k$, then we have

$$\left\| \begin{bmatrix} q^k \\ \text{vec}(Q^k) \end{bmatrix} - \begin{bmatrix} q^\infty \\ \text{vec}(Q^\infty) \end{bmatrix} \right\| \leq C_0 \max\{\sigma(\mathcal{H}_{11}) + \varepsilon, \sigma(\mathcal{H}_{22}) + \varepsilon, \tilde{\rho}\}^k.$$

Applying MJLS Theory to TD Learning

1. Key idea: Some **augmented versions** of the mean and covariance matrix of TD learning exactly track a deterministic LTI dynamical system, and hence can be fully understood.

$$q_i^k = \mathbb{E} \left(\xi^k \mathbf{1}_{\{i_k=i\}} \right)$$
$$Q_i^k = \mathbb{E} \left(\xi^k (\xi^k)^\top \mathbf{1}_{\{i_k=i\}} \right)$$

2. The same analysis can be applied to a large family of TD algorithms including GTD, GTD2, TDC, DTD, and ATD.
3. More details can be found in our recent arxiv paper entitled **“Characterizing the Exact Behaviors of Temporal Difference Learning Algorithms Using Markov Jump Linear System Theory.”**

<https://arxiv.org/pdf/1906.06781.pdf>

Outline: Unified Analysis Tools

- Analyzing Stochastic Methods in Supervised Learning
- Analyzing Temporal Difference in Reinforcement Learning
- **Future Directions**

Future Directions

- Tailoring control theory to understand generalization in supervised learning: implicit regularization, non-convexity, etc
- Control-oriented analysis for reinforcement learning: Q-learning, policy gradient, actor-critic
- Convergence of stochastic versions of policy gradient on robust control tasks: How to handle robustness constraints in stochastic optimization without projection?

Thanks!

If you are interested, feel free to send an email to binhu7@illinois.edu