ECE 598: Interplay between Control and Machine Learning **Fall 2020**

Lecture 1

Overview: "Control for Learning" and "Learning for Control"

Lecturer: Bin Hu, Date:08/25/2020

In this lecture, we give an overview of this course. This course covers two main ideas: "control for learning" and "learning for control." In the first half of the course, we will talk about how to apply control theory to unify and automate the analysis and design of machine learning algorithms. In the second half of the course, we will talk about how to combine reinforcement/imitation learning methods with model-based feedback design to solve various control tasks provably.

1.1 Control for Learning

Analysis and design for machine learning algorithms are typically done in a case-by-case manner. To analyzed a different algorithm, or change the underlying assumptions, we would likely require a different analysis technique. We present two examples to illustrate this point.

- Example 1: SAG vs. SAGA. Many supervised learning tasks including classification and prediction can be naturally formulated as the empirical risk minimization (ERM) problem $\min \frac{1}{n}\sum_{i=1}^n f_i(x)$. Here, n denotes the size of the training set and can be quite large. A full gradient computation can be too expansive. Stochastic average gradient (SAG) [5, 6] and SAGA [2] are two popular stochastic iterative methods that can be used to efficiently solve ERM on generalized linear models. At every step k, both SAGA and SAG will draw a random index i_k from the set $\{1, 2, \ldots, n\}$ and only use the i_k-th data point in the training set.

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

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

  Although the update rules for SAG and SAGA are quite similar, the convergence proofs for these two methods are quite different. The analysis for SAG is much more complicated. A small change in the algorithm causes big trouble for analysis.

- Example 2: Markov assumption vs. IID assumption for temporal difference (TD) learning. TD learning is usually used to solve the policy evaluation task in reinforcement learning. The goal of TD learning is to determine the value function of a given policy. The standard TD method (or TD(0)) uses the following update rule:

$$\theta^{k+1} = \theta^k - \alpha\psi(s^k)\left((\psi(s^k) - \gamma\psi(s^{k+1}))^T\theta^k - r(s^k)\right),$$

where $\{s^k\}$ is the underlying Markov chain, ψ is the feature vector, r is the reward, γ is the discounting factor, and θ^k is the weight vector to be estimated. When $\{s^k\}$ is IID, the finite-time analysis of TD(0) is straightforward [1, 4]. However, a more reasonable assumption on $\{s^k\}$ for the MDP setup is that $\{s^k\}$ forms a Markov chain. The analysis of TD learning under this assumption is much more involved and the first finite sample bound was actually obtained in 2019 [8].

One can find many similar examples cases in machine learning. In contrast, control theory is developed to provide general tools for analysis and design of feedback dynamical systems. The field of control theory can be divided into many branches including classical control theory, linear control theory, robust control theory, switching system theory, Markovian jump linear system (MJLS) theory, parameter-varying system theory, nonlinear control theory. Each of these branches covers the analysis and design of a large family of systems with some common features. See Table 1.1 for a few sample results. The stability conditions in Table 1.1 are all in the form of semidefinite programs and can be efficiently verified using existing convex solvers. To analyze the stability of a given system, one only needs to find a positive definite matrix P (or a sequence of positive definite matrices $\{P_j\}$) to satisfy the matrix inequality conditions summarized in Table 1.1. When the state/input matrices or the problem assumptions (e.g. the assumption on ϕ) are changed, one can still use the same conditions. The point is that the tools in control theory are typically developed to handle a large family of dynamical systems in some unified manner.

	LTI systems	MJLS	Lur'e systems
Model	$\xi^{k+1} = A\xi^k + Bu^k$	$\xi^{k+1} = A_{i_k}\xi^k + B_{i_k}u^k$	$\xi^{k+1} = A\xi^k + B\phi(C\xi^k)$
Stability	$A^\mathsf{T}PA - P < 0$	$\sum_{i=1}^n p_{ij}A_i^\mathsf{T}P_iA_i - P_j < 0$	$\begin{bmatrix} A^\mathsf{T}PA - P & A^\mathsf{T}PB \\ B^\mathsf{T}PA & B^\mathsf{T}PB \end{bmatrix} + M < 0$

Table 1.1. Several stability conditions developed in the controls literature.

The key idea forming the basis for the first half of the course is that we can view various machine learning algorithms as feedback control systems and tailor control theory to analyze and design learning algorithms. To illustrate this, we revisit Examples 1 & 2.

- A control perspective on Example 1: Both SAG and SAGA can be viewed as special cases of Markov jump Lur'e systems governed by the model $\xi^{k+1} = A_{i_k}\xi^k + B_{i_k}\phi(C\xi^k)$ where the jump parameter i_k is a random variable sampled from a finite set, and

$(A_{i_k}, B_{i_k}) = (A_i, B_i)$ for $i_k = i$. The random process $\{i_k\}$ is allowed to be either an IID process or a Markov chain. Let e_i be a n-dimensional vector whose i-th entry is 1 and all other entries are 0. Then the Markov jump Lur'e system model can cover SAG, SAGA, and many other stochastic optimization methods as special cases if we can choose (A_{i_k}, B_{i_k}, C) according to Table 1.2 and define the nonlinearity ϕ as $\phi(x) = \begin{bmatrix} \nabla f_1(x)^\mathsf{T} & \nabla f_2(x)^\mathsf{T} & \dots & \nabla f_n(x)^\mathsf{T} \end{bmatrix}^\mathsf{T}$. The behaviors of all these stochastic optimization methods can be studied by applying the Markov jump Lur'e system theory.

Method	A_{i_k}	B_{i_k}	C
SAGA [2]	$\begin{bmatrix} I_n - e_{i_k} e_{i_k}^T & \tilde{0} \\ -\frac{\alpha}{n}(e - n e_{i_k})^T & 1 \end{bmatrix}$	$\begin{bmatrix} e_{i_k} e_{i_k}^T \\ -\alpha e_{i_k}^T \end{bmatrix}$	$\begin{bmatrix} \tilde{0}^T & 1 \end{bmatrix}$
SAG [5, 6]	$\begin{bmatrix} I_n - e_{i_k} e_{i_k}^T & \tilde{0} \\ -\frac{\alpha}{n}(e - e_{i_k})^T & 1 \end{bmatrix}$	$\begin{bmatrix} e_{i_k} e_{i_k}^T \\ -\frac{\alpha}{n} e_{i_k}^T \end{bmatrix}$	$\begin{bmatrix} \tilde{0}^T & 1 \end{bmatrix}$
Finito [3]	$\begin{bmatrix} I_n - e_{i_k} e_{i_k}^T & \tilde{0} \\ -\alpha(e_{i_k} e^T) & I_n - e_{i_k}(e_{i_k}^T - \frac{1}{n} e^T) \end{bmatrix}$	$\begin{bmatrix} e_{i_k} e_{i_k}^T \\ \tilde{0}\tilde{0}^T \end{bmatrix}$	$\begin{bmatrix} -\alpha e^T & \frac{1}{n} e^T \end{bmatrix}$
SDCA [7]	$I_n - \alpha m n e_{i_k} e_{i_k}^T$	$-\alpha m n e_{i_k} e_{i_k}^T$	$\frac{1}{mn} e^T$

Table 1.2. Jump Lur'e system models for SAG, SAGA, and other stochastic finite-sum methods.

- A control perspective on Example 2: We can set $i_k = \begin{bmatrix} (s^{k+1})^\mathsf{T} & (s^k)^\mathsf{T} \end{bmatrix}^\mathsf{T}$, and then the TD learning method becomes a Markov jump linear system $\xi^{k+1} = A_{i_k}\xi^k + B_{i_k}u^k$ with $\xi^k = \theta^k$, $u^k = 1$, $A_{i_k} = I + \alpha\psi(s^k)(\gamma\psi(s^{k+1}) - \psi(s^k))^\mathsf{T}$, and $B_{i_k} = \alpha\psi(s^k)r(s^k)$. Hence we can directly apply the existing MJLS theory to obtain exact closed-form expressions for the mean and covariance matrix of the TD estimation error under both the IID and Markov assumptions.

The key message is that many algorithms in supervised, reinforcement, and unsupervised learning can be viewed as feedback control systems. The first half of the course focuses on tailoring control theory for analysis and design of machine learning algorithms.

1.2 Learning for Control

Suppose we want to control a general nonlinear system $x^{k+1} = f(x^k, u^k, w^k)$ where x^k is the state, u^k is the control action, and w^k is the process noise. The objective is to design a mapping from the state measurement to the actuation input, i.e. $u^k = K(x^k)$. Here K is called "controller" or "policy." Such a control design problem can be viewed as a constrained optimization problem $\min_{K\in\mathcal{K}} \mathcal{J}(K)$, where the decision variable K is determined by the controller parameterization, the cost function $\mathcal{J}(K)$ is a pre-specified control performance measure, and the feasible set $\mathcal{K}$ carries the information of the constraints on the controller.

- Optimization variable K: In the simplest case, K is parameterized as a static matrix, i.e. $x^k = Ku^k$. In the machine learning community, it is more popular to parameterize K as a neural network.
- Objective function $\mathcal{J}(K)$: $\mathcal{J}(K)$ is specified to measure the performance of a given controller K. Popular choices in the controls literature include $\mathcal{H}_2$ or $\mathcal{H}_\infty$-norm (or some related upper bounds) of the closed-loop systems. For standard reinforcement learning models that are based on Markov decision processes (MDPs), the cost $\mathcal{J}(K)$ can be handcrafted in a case-by-case manner and always has an additive structure over time. For imitation learning models, the goal is to fit the control policy K based a given set of measurement/action pairs demonstrated by experts, and the cost $\mathcal{J}(K)$ is set up to measure the discrepancy between the true control action and the estimated outputs of the controller.
- Feasible set $\mathcal{K}$: Constraints on the decision variable K are typically posed to account for either the stability, robustness, safety, or structural concerns on the system. A common, though sometimes implicit, example in continuous control tasks is the stability constraint, i.e. K is required to stabilize the closed-loop dynamics. In the control field, there are also other constraints related to robustness or safety concerns in control design. Examples include the famous $\mathcal{H}_\infty$-constraint and various IQC-based constraints from the robust control literature. In the reinforcement learning field, the constraints are often imposed on either the expected long-term cost or some risk-related constraint.

Then we can apply gradient descent method or its variants to search for the optimal K. If the model is unknown, machine learning methods can be applied to estimate the gradient $\nabla\mathcal{J}$ from data. Many reinforcement learning methods such as policy gradient, natural policy gradient, trust region policy optimization (TRPO), proximal policy optimization (PPO), and evolutionary strategies (ES) are based on such an idea. The second half of the course focuses on how to combine machine learning methods with model-based control techniques to handle problems with stability/robustness/safety constraints. A few sample topics are listed below.

- Guarantees of reinforcement/imitation learning methods on linear control problems

$$K' = K - \alpha \mathbf{S}\left(\nabla\mathcal{J}(K)\right)$$

- Robustness constraints: robust control theory
- Safety constraints: model predictive control, constrained policy optimization
- Repetitive tasks: iterative learning control
- Combination of model-free and model-based methods

Bibliography

[1] G. Dalal, B. Szörényi, G. Thoppe, and S. Mannor. Finite sample analyses for TD (0) with function approximation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[2] A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, 2014.

[3] A. Defazio, J. Domke, and T. Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1125–1133, 2014.

[4] C. Lakshminarayanan and C. Szepesvari. Linear stochastic approximation: How far does constant step-size and iterate averaging go? In *International Conference on Artificial Intelligence and Statistics*, pages 1347–1355, 2018.

[5] N. Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets. In *Advances in Neural Information Processing Systems*, 2012.

[6] M. Schmidt, N. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *ArXiv preprint*, 2013.

[7] S. Shalev-Shwartz. SDCA without duality, regularization, and individual convexity. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 747–754, 2016.

[8] R. Srikant and L. Ying. Finite-time error bounds for linear stochastic approximation and TD learning. *arXiv preprint arXiv:1902.00923*, 2019.