Reinforcement Learning in Adaptive Control

Chang-Shao Shen
Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan
Email: b04901088@ntu.edu.tw

Bing-Jyue Chen
Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan
Email: b04901087@ntu.edu.tw

***Abstract*—This paper reviews the current progress on reinforcement learning (RL) based feedback control solutions to regulation and tracking problems to design controllers for discrete-time (DT) linear/nonlinear dynamical systems that combine features of adaptive control and optimal control. Discussion will mainly focus on state feedback methods. Output feedback, however, will also be reviewed. Moreover, lyapunov-based RL control for stability will be covered as well. Core algorithms will be based on Q-learning and actor-critic method.**

***Index Terms*—reinforcement learning (RL), optimal adaptive control**

I. Introduction

Reinforcement learning is a framework in machine learning to tactically modify the actions of an agent based on the observed response from the environment it is interacting with. RL methods stem from the research from psychology of human decision making. One interacts with the environment and learn optimal behaviors to maximize the positive effect of the response.

In control engineering paradigm, RL refers to the process of learning optimal controller based on the response from the environment without knowing the system dynamics, which is also called *action-based learning*. Previous work [1] has shown that reinforcement learning control is direct adaptive optimal control. Optimal control is generally an offline deisgn technique that requires full knowledge of system dynmaics to solve HJB equations. On the other hand, adaptive controllers learn online to control unknown system using data measured along the system trajectories, while not gauranteeing the input to be optimal in the sense of minimizing prescribed cost function. With reinforcement learning, a controller is able to solve HJB equations online without knowledge of system dynamics and adapt the system input to that derived from optimal control theory.

This paper is organized as follows. Section II briefly reviews some background theory in reinforcement learning, i.e., Markov decision process, Bellman equation, and value iteration. Section III presents solutions to discrete-time (DT) optimal control problems, including regulation and tracking problem for both linear and nonlinear systems using RL algorithms. Finally in Section IV, miscellaneous improvement of RL methods including lyapunov-constrained algorithm for stability and output feedback algorithm for partially observable states will be introduced, followed by concluding remarks in Section V.

II. Background Theory

A. Reinforcement Learning

The study of reinforcement learning can be based on Markov decision process (MDP), which can be started by defining optimal decision problems, where decisions are made at stages of a process evolving through time, and total cost is accumulated along the trajectory. Consider the MDP (X, U, P, R), where X is a set of state, and U is a set of actions or control input. $P : X \times U \times X \to [0, 1]$ is the transition probability from state x to state x' under input u, also denoted as $Pr\{x'|x, u\}$. Here we assumed the transition is always deterministic. That is, $Pr\{x'|x, u\}$ is always 1. $R : X \times U \times X \to \mathbb{R}$ is the cost-to-go picked up after state transition from x to x' under input u. The mapping from state to action is called *policy*, denoted as π. A policy gives the conditional probability $\pi(x, u) = Pr\{u|x\}$ of taking action u in state x. Here we also assume the policy is always deterministic. The goal of reinforcement learning is to minimize accumulated cost-to-go along the trajectory by finding optimal policy π^*.

To find the optimal policy, we define the following functions.

$$\begin{aligned} J_k &= \sum_{i=0}^{\infty} \gamma^{i-k} r_i \\ V^\pi(x_k) &= E_\pi\{J_k | x = x_k\} \end{aligned} \tag{1}$$

where J_k is the total infinite-horizon cost starting from time k, and $\gamma \in \mathbb{R},\ 0 < \gamma \leq 1$ is the discount factor, and r_i is the cost-to-go for every state transition. $V^\pi(x_k)$, called *value function*, is the expected value of J_k under policy π. Qualitatively, value function is an estimation of the long term cost a state expects. We can find the optimal policy by minimizing the value function

$$\pi^* = \underset{\pi}{\operatorname{argmin}}\, V^\pi(x_k)$$

And the corresponding optimal value function is

$$V^*(x_k) = \min_\pi V^\pi(x_k)$$

After some derivation, the definition of value function in (1) can be expressed as

$$V^\pi(x_k) = r_k + V^\pi(x_{k+1}) \tag{2}$$

under the assumption of deterministic policy and transition probability. (2) is called *Bellman equation*, which is the fundamental of RL algorithms. It provides a backward recursion for the value at time k in terms of the value at time $k+1$.

With principle of optimality, the optimal value function must satisfy

$$V^*(x_k) = \min_{u_k}[r_k + V^*(x_{k+1})] \tag{3}$$

which is a fixed-point equation. Therefore it can be solved by successive approximation using an algorithm called *value iteration*, which will be introduced in the next section.

III. Solutions to DT Optimal control using RL

Consider the following linear and nonlinear systems:

$$x_{k+1} = Ax_k + Bu_k \tag{4}$$

$$x_{k+1} = f(x_k) + g(x_k)u_k \tag{5}$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^m$ is the control input, $A \in \mathbb{R}^{n\times n}$, $f(x_k) \in \mathbb{R}^n$ are the drift dynmaics, $B \in \mathbb{R}^{n\times m}$, $g(x_k) \in \mathbb{R}^{n\times m}$ are the input dynamics. It is assumed that the system is stabilizable and (A, B) are controllable. Output feedback will be discussed in Section IV and thus output dynamics(often denoted as C matrix) is not introduced here.

A. Optimal Regulation Problem

The goal of optimal regulation problem is to design an input that stabilizes the state to a desired target(often zero), while minimizing the predesigned performance function, i.e. long term cost function, which can be defined as

$$J = \sum_{i=0}^{\infty} r(x_i, u_i) = \sum_{i=0}^{\infty}(Q(x_i) + u_i^T R u_i) \tag{6}$$

where $Q(x_i) \succeq 0$, $R = R^T \succ 0$ are predefined state cost-to-go function and input cost matrix respectively. The value function can be defined as

$$V(x_k) = \sum_{i=k}^{\infty} r(x_i, u_i) = \sum_{i=k}^{\infty}(Q(x_i) + u_i^T R u_i), \forall x \tag{7}$$

which can also be formulated as Bellman equation

$$V(x_k) = r(x_k, u_k) + V(x_{k+1}) \tag{8}$$

Thus, with principle of optimality, the problem reduces to finding optimal value function.

$$V^*(x_k) = \min_{u_k}[r(x_k, u_k) + V^*(x_{k+1})] \tag{9}$$

which is the DT HJB equation. For DT linear system as in (4), typical input cost function is defined as $Q(x_k) = x_k^T Q x_k, Q \succ 0$. Then the value function is shown to be quadratic.

$$V(x_k) = x_k^T P x_k \tag{10}$$

where $P = P^T \succ 0$ is called kernel matrix. Putting (10) into (9) and solve the equation yields the DT Algebraic Riccati equation(DARE)

$$Q - P + A^T PA - A^T PB(R + B^T PB)^{-1}B^T PA = 0 \tag{11}$$

and the optimal control input is

$$\begin{aligned} u_k^* &= \underset{u_k}{\operatorname{argmin}}[r((x_k, u_k) + V^*(x_{k+1}))] \\ &= -(R + B^T PB)^{-1}B^T PAx_k \end{aligned} \tag{12}$$

In the case of nonlinear system in (5). The optimal control input is

$$\begin{aligned} u_k^* &= \underset{u_k}{\operatorname{argmin}}[r((x_k, u_k) + V^*(x_{k+1}))] \\ &= -\frac{1}{2}R^{-1}g^T(x)(\frac{\partial V(x_{k+1})}{\partial x_{k+1}}) \end{aligned} \tag{13}$$

The following presents methods for solving HJB equation online using RL.

1) Value Iteration (VI): Algorithm 1 provides a solution to solve HJB equation online using value iteration but requires full knowledge of system dynamics.

Algorithm 1 VI Algorithm to solve HJB

1: Select a initial control policy u_k^0.
2: For $j = 0, 1, ...,$ perfrom until convergence:
Value Update:

$$V^{j+1}(x_k) = Q(x_k) + (u_k^j)^T R u_k^j + V^j(x_{k+1})$$

on convergence, set $V^{j+1}(x_k) = V^j(x_k)$.
3: **Policy Improvement:**

$$u_k^{j+1} = -(R + B^T PB)^{-1}B^T PAx_k \quad \text{(linear case)}$$

$$u_k^{j+1} = -\frac{1}{2}R^{-1}g^T(x)(\frac{\partial V(x_{k+1})}{\partial x_{k+1}}) \quad \text{(nonlinear case)}$$

4: Go to 2

2) Q Learning for the DT LQR: Q Learning is model-free learning approach which converges to the global optimal solution under the condition of persistence exciting (PE) signals. Q function is an action-dependent value function, representing how positive a state-action pair is. DT Q function is defined as

$$Q(x_k, u_k) = r(x_k, u_k) + V(x_{k+1})$$

Let $z_k = [x_k^T z_k^T]^T$, generally the Q function can be paramterized as

$$Q(x, u) = W^T\phi(z) = \sum_{i=1}^{N} w_i\varphi_i(z)$$

where $\phi(z) = [\varphi_1(z)\ \varphi_2(z) \dots \varphi_N(z)]$ with $\varphi_i(z)$ as basis functions. N is the number of neurons, and w_i are the neural network weights. This is called *value function approximation (VFA)* technique, often used for continuous state action space in which selecting action from discrete action space and updating value function from discrete state space is computationally expensive or impossible.

For the problem of linear quadratic regulation (LQR), from (8) and (10), the Q function is

$$Q(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P x_{k+1}$$

Apply the dynamics in (4), the Q function becomes

$$Q(x_k, u_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T P A & A^T P B \\ B^T P A & R + B^T P B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}$$

Define a kernel matrix S as

$$Q(x_k, u_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} S_{xx} & S_{xu} \\ S_{ux} & S_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} = z_k^T S z_k$$

Thus, for DT LQR, the Q function is better parameterized as

$$Q(z_k) = z_k^T S z_k$$

Solving $\frac{\partial Q}{\partial u_k} = 0$ yields the optimal control input

$$u_k^* = -S_{uu}^{-1} S_{ux} x_k$$

The value step can be implemented with batch LS, recursive

Algorithm 2 Q Learning algortihm to solve DARE

1: Select a initial control policy u_k^0.
2: For $j = 0, 1, ...,$ perfrom until convergence:
Value Update:

$$z_k^T S^{j+1} z_k = x_k^T Q x_k + u_k^T R u_k + z_{k+1}^T S^j z_{k+1}$$

on convergence, set $S^{j+1} = S^j$.
3: **Policy Improvement:**

$$u_k^* = -S_{uu}^{-1} S_{ux} x_k$$

4: Go to 2

LS, and gradient methods using data collected along the state trajecories.

3) Actor-Critic Approximators: We can extend the idea of VFA described in III-A2 to *policy approximation* to further generalize the system and avoid the requirement of knowledge of plant dynamics. This method is called *Actor-Critic* method, in which two approximators — value function approximator and policy approximator are structured. The value function approximator is maintained by *critic* and estimates the value function by being updated to minimizing the temporal difference error. The policy approximator is maintained by *actor* and estimates the control input from the current state based on the current value function while minimizing the value function. The two approximators can be represented as follows.

$$\begin{aligned} V^j(x_k) &= (W_c^j)^T \phi_c(x_k) = \sum_{i=1}^{N_c} w_c^j \varphi_{ci}(x_k), \quad \forall x \\ u_k^j &= (W_a^j)^T \sigma_a(x_k) = \sum_{i=1}^{N_a} w_a^j \varsigma_{ai}(x_k), \quad \forall x \end{aligned} \tag{14}$$

where $\varphi_{ci}(x_k)$ and $\varsigma_{ai}(x_k)$ are the basis functions, $W_c^j = \left[w_{c1}^j w_{c2}^j \dots w_{cNc}^j\right]$, $W_a^j = \left[w_{a1}^j w_{a2}^j \dots w_{aNa}^j\right]$ are the weight vectors of the approximators with N_c being the number of neurons of critic and N_a being the number of neurons of actor. Thus, the Bellman equation of value function approximator can be written as

$$(W_c^{j+1})^T \phi_c(x_k) = r(x_k, u_k^j) + (W_c^{j+1})^T \phi_c(x_{k+1})$$

And the gradient adaptive laws for actor and critic are derived as

$$\begin{aligned} W_c^{j+1} \leftarrow W_c^{j+1} - \alpha_1 \phi_c(x_k)((W_c^{j+1})^T \phi_c(x_k) \\ - r(x_k, u_k^j)) \end{aligned} \tag{15}$$

$$\begin{aligned} W_a^{j+1} \leftarrow W_a^{j+1} - \alpha_2 \sigma_a(x_k) \Big(2R(W_a^{j+1})^T \sigma_a(x_k) + \\ g^T(x_k) \frac{\partial \phi_c(x_{k+1})}{\partial x_{k+1}} W_c^{j+1} \Big)^T \end{aligned} \tag{16}$$

where $\alpha_1, \alpha_2 > 0$ are the tuning gains, also known as learning rates. It can be observed that with the actor-critic structure, state drift dynamics $f(\cdot)$ is not required to obtain an optimal control input.

The structure of the value function approximator plays an important role in convergence and performance of value iteration. If the structure is designed inappropriately, it may never converge to an optimal solution. For linear systems, the form of value function is shown to be quadratic as in (10). For nonlinear systems, the approximation can be single-layer neural network with a large number of neurons or a multi-layer neural to gaurantee a acceptable approximation error.

B. Optimal Tracking

The goal of optimal tracking is to make the state x_k follow the desired tracjectory x_k^d. Define the error e_k as

$$e_k = x_k - x_k^d$$

Optimal control input is designed to minmize the cost functional

$$J = \sum_{i=0}^{\infty} (e_k^T(i) Q e_k(i) + u_k^T(i) R u_k(i))$$

Traditional solutions involve a feedback term and a feedward term, which requires complete knowledge of the system dynamics and the reference dynamics. In [8], a new solution is developed for linear tracking using Q learning without full knowledge of system dynamics. The solution is described as follows. Consider the following augmented states

$$X_k = \begin{bmatrix} x_k \\ x_k^d \end{bmatrix}$$

$$Z_k = \begin{bmatrix} X_k \\ u_k \end{bmatrix}$$

Define Q function as

$$Q(X_k, u_k) = \frac{1}{2} X_k^T Q_1 X_k + \frac{1}{2} u_k^T R u_k + \gamma Q(X_{k+1}, u_{k+1})$$

where $\gamma \in \mathbb{R}$, $0 < \gamma \leq 1$ is the discount factor, $Q_1 = [I - I]^T Q [I - I]$. Q function can also be written as

$$Q(Z_k) = \frac{1}{2} Z_k^T H Z_k$$

And Bellman equation can be derived.

$$Z_k^T H Z_k = X_k^T Q_1 X_k + u_k^T R u_k + \gamma Z_{k+1}^T H Z_{k+1}$$

Then apply algorithm 2 with respect to the above modified Q function leads to optimal tracking solution. Note that an actor-critic structure similar to (14) can also be implemented to obtain optimal control input online.

IV. Miscellaneous improvements on RL control

A. Lyapunov-constrained Algorithm for Stability

In RL there is a tradeoff between exploitation and exploration of policy. Exploitation ensures the optimization of the policy, while exploration chooses random action with the aim of finding optimal state that may not be reached with greedy optimization of policy. Therefore, the agent can choose an action that lead to instability of the system. To cope with this problem, we can impose lyapunov stability condition to constraint action space into a stability-assured action space. Lyapunov approach starts with the construction of a scalar function known as lyapunov candidate function, which is always positive definite with negative time derivative. That is, a lyapunov candidate function $L(x)$ musy satisfy the following conditions

$$\begin{aligned} &L(0) = 0 \\ &L(x) > 0,\ x \neq 0 \\ &\dot{L}(x) < 0,\ x \notin 0 \end{aligned} \tag{17}$$

In DT system, $\dot{L}(x)$ is interpreted as the difference

$$L(x_{k+1}) - L(x_k)$$

In [7], cost-to-go function $r(x,u) = r(x)$ is selected as lyapunov candidate function $L(x)$ to constraint the action space for stability.

B. Output feedback Algorithm for Partially Observable States

In practice, full state information is usually not available. Thus, methods using only input/output data is needed for realistic use. In control theory, these types of methods are called output feedback, also termed as partially observable Markov decision processes in RL.

In [4], an linear quadratic controller is developed in the form of autoregressivemoving-average (ARMA) controller, in which input/output sequences are applied to realize output feedback.

$$\overline{u}_{k-1,k-N} = \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix},\ \overline{y}_{k-1,k-N} = \begin{bmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-N} \end{bmatrix}$$

$$\overline{z}_{k-1,k-N} = \begin{bmatrix} \overline{u}_{k-1,k-N} \\ \overline{y}_{k-1,k-N} \end{bmatrix}$$

where $\overline{u}_{k-1,k-N}$ and $\overline{y}_{k-1,k-N}$ are the input/output sequences over the time interval $[k-N, k-1]$, $\overline{z}_{k-1,k-N}$ is the augmented state. Then, a augmented kernel matrix for Q function can be developed with $\overline{z}_{k-1,k-N}$ for Q learning.

V. Conclusion

In this paper, we have reviewed serveral methods using reinforcement learning technique to bring together optimal control and adaptive control, with solutions to DT regulation and tracking problems as demonstration. HJB equations, which are generally solved offline and require precise model of the system in optimal control, can be solved online using data collected along the trajectory without full knowledge of the system dynamics with RL-based methods. In addition, we also review improvements on these methods, including stability-assured RL controller with lyapunov criterion and output feedback controller with ARMA.

References

[1] Sutton RS, Barto AG, Williams RJ (1992) Reinforcement learning is direct adaptive optimal control. IEEE Control Systems 12(2): 19-22.

[2] B. Kiumarsi, K. G. Vamvoudakis, H. Modares and F. L. Lewis, "Optimal and Autonomous Control Using Reinforcement Learning: A Survey," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 6, pp. 2042-2062, June 2018.

[3] F. L. Lewis, D. Vrabie and K. G. Vamvoudakis, "Reinforcement Learning and Feedback Control: Using Natural Decision Methods to Design Optimal Adaptive Controllers," in IEEE Control Systems, vol. 32, no. 6, pp. 76-105, Dec. 2012.

[4] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement Learning for Partially Observable Dynamic Processes: Adaptive Dynamic Programming Using Measured Output Data," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 41, no. 1, pp. 14-25, Feb. 2011.

[5] J. Hall, C. E. Rasmussen and J. Maciejowski, "Reinforcement learning with reference tracking control in continuous state spaces," 2011 50th IEEE Conference on Decision and Control and European Control Conference, Orlando, FL, 2011, pp. 6019-6024.

[6] Y. P. Pane, S. P. Nageshrao and R. Babuka, "Actor-critic reinforcement learning for tracking control in robotics," 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, 2016, pp. 5819-5826.

[7] A. Kumar and R. Sharma, "A stable Lyapunov constrained reinforcement learning based neural controller for non linear systems," International Conference on Computing, Communication & Automation, Noida, 2015, pp. 185-189.

[8] Bahare Kiumarsi, Frank L. Lewis, Hamidreza Modares, Ali Karimpour, Mohammad-Bagher Naghibi-Sistani, Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics, Automatica, Volume 50, Issue 4, 2014, Pages 1167-1175.