Online Model-Free RLSPI Algorithm for Nonlinear Discrete-Time Non-affine Systems*

Yuanheng Zhu and Dongbin Zhao

State Key Laboratory of Management and Control for Complex Systems,
Institute of Automation Chinese Academy of Sciences, Beijing, China
zyh7716155@163.com, dongbin.zhao@gmail.com

Abstract. Policy iteration, as one kind of reinforcement learning methods is applied here to solve the optimal problem of nonlinear discrete-time non-affine system with continuous-state and continuous-action space. By applying action-value function or Q function, the implementation of policy iteration avoids the dependence on system dynamics. Online model-free recursive least-squares policy iteration (RLSPI) algorithm is proposed with continuous policy approximation. It is the first attempt to develop online LSPI algorithm for nonlinear discrete-time non-affine systems with continuous policy. A nonlinear discrete-time system is simulated to verify the efficiency of our algorithm.

Keywords: online policy iteration, Q function, nonlinear discrete-time non-affine system, linear parametrization, RLSPI.

1 Introduction

Reinforcement learning (RL) refers to one kind of methods that try to find optimal or near-optimal policies for complicated systems or agents [1-3]. Policy iteration (PI) is one powerful instrument of RL to solve optimal problems. PI [4] includes two steps: policy evaluation and policy improvement. With iteration of these two steps, PI improves policy constantly and finally achieves the optimal one [5].

As RL developed, function approximation (FA) technique was introduced to RL to solve continuous optimal problems and promoted the development of RL. Such as approximate SARSA [6], TD(λ) [7] and so on. Especially recent years, a new branch of RL, adaptive dynamic programming (ADP) was proposed [2]. Some overviews about ADP are given in [2], [3], [8] and [9].

As FA technique is used for the approximation of value function, parameters of approximation have to be learned based on data. And a lot of online algorithms have developed. SARSA is an online algorithm which modifies value function based on temporal difference (TD) error with gradient method. Si and Wang [10] applied ADP for online learning of under-actuated control systems and presented great

* This work was supported in part by National Natural Science Foundation of China (Nos. 61273136, and 61034002,), and Beijing Natural Science Foundation No. 4122083.

M. Lee et al. (Eds.): ICONIP 2013, Part II, LNCS 8227, pp. 242–249, 2013.

performance. However, those algorithms make a limited use of data, which does not benefit their application.

To solve those problems, Busoniu *et al.* [11] extended offline least-squares policy iteration (LSPI) [12] to an online LSPI algorithm. And this algorithm employed Q function featured as model-free and the results revealed great performance for online learning. However, a batch least-squares method was used and only discrete-action policies were applied.

In this paper, we focus on a brand new research field of online model-free recursive learning with continuous-action policy approximation for nonlinear discrete-time non-affine systems. An online model-free RLSPI algorithm is proposed using linear function approximation for continuous state and action systems. To the limit of our knowledge, it is the first attempt to combine continuous policy approximation with LSPI for online learning.

This paper is organized as follows. In Section 2, PI method using Q function is introduced to solve optimal control problem of nonlinear discrete-time non-affine system. Then an online model-free algorithm, RLSPI, is proposed in Section 3 to solve this kind of problems online. And a nonlinear example is simulated with the new algorithm. In the end, we have our conclusion.

2 PI for Nonlinear Discrete-Time Non-affine Optimal Problem

2.1 Nonlinear Discrete-Time Non-affine Optimal Problem

In this paper, the nonlinear discrete-time non-affine system is denoted by $x_{k+1} = f(x_k, u_k)$, where $x_k, x_{k+1} \in R^n$, $u_k \in R^m$, $f : R^n \times R^m \to R^n$, and k is the step index. Assume the system is controllable on a compact set Ω and 0 is the equilibrium point. Suppose a negative definite function $r(x_k, u_k)$ is used as the reward at each step, and $h : R^n \to R^m$ represents a policy.

Given a policy h, the following definition specifies its action-value function, or Q function

$$Q^h(x_k, u_k) = r(x_k, u_k) + Q^h(f(x_k, u_k), h(f(x_k, u_k))) . \tag{1}$$

Q^h are negative definite. And the optimal control problem is to find the maximum Q function and optimal policy, namely

$$Q^*(x_k, u_k) = \max_h Q^h(x_k, u_k) , \tag{2}$$

$$h^*(x_k) = \arg\max_u Q^*(x_k, u) . \tag{3}$$

It is important to note that the Q function defined here is undiscounted. So a definition is introduced from [13] to guarantee the validity

Definition 1. (Admissible Policy) A control policy h is defined to be admissible, denoted by $h \in \psi(\Omega)$, if h is continuous on Ω, $h(0)=0$, h stabilizes the system on Ω and for $\forall x_k \in \Omega$, $Q^h(x_k,u_k)$ is finite.

2.2 Policy Iteration

Based on Q function, policy evaluation and policy improvement of PI is presented as follows

PI method

1. Policy evaluation: given an admissible control policy $h^{(i)} \in \psi(\Omega)$, calculate relevant Q function by

$$Q^{(i)}(x_k,u_k) = r(x_k,u_k) + Q^{(i)}(f(x_k,u_k),h^{(i)}(f(x_k,u_k))),\ Q^{(i)}(0,0)=0. \tag{4}$$

2. Policy improvement: generate a new improved policy $h^{(i)}$ using

$$h^{(i+1)}(x_k) = \arg\max_u Q^{(i)}(x_k,u). \tag{5}$$

As the policy is improved over and over again, the optimal policy can be finally obtained.

3 An On-Line Model-Free RLSPI Algorithm

LSPI method was first proposed by Lagoudakis and Parr [12] to utilize the linear property of PI method and apply least-squares method for finite state and action sets. Then Busoniu *et al.* [11] extended this method to an online version for infinite and continuous-state space and finite-action sets. However, their algorithm is not suitable for more general continuous action systems.

Different from offline or online LSPI algorithm using batch least-squares at the end of iterations, a new online model-free RLSPI algorithm is proposed which applies RLS method at each step and uses continuous policy approximation for continuous state and action control problems.

3.1 Q Function and Policy Approximation

As state and action spaces are continuous, approximation of Q function and policy is necessary. Here, linear parametrization technique [13] is used.

A linear parametrization of Q function can be expressed by $\hat{Q}(x,u) = \phi^T(x,u)\theta$, where $\phi(x,u) = [\phi_1(x,u),...,\phi_N(x,u)]^T$ is a vector of N basis functions (BFs), and $\theta \in R^N$ is a parameter vector. Like many others works [13], polynomials are adopted

here as BFs. Suppose $x=[x_1,...,x_n]^T$ and $u=[u_1,...,u_m]^T$ and let polynomial BF ϕ_i have the form $\phi_i(x,u)=x_1^{\alpha_1^i}x_2^{\alpha_2^i}\cdots x_n^{\alpha_n^i}u_1^{\beta_1^i}u_2^{\beta_2^i}\cdots u_m^{\beta_m^i}$.

Similarly, denote the linear parametrization of policy by $\hat{h}(x)=\omega^T\varphi(x)$, where $\varphi(x)=[\varphi_1(x),...,\varphi_M(x)]^T$ and $\omega=[\omega_1,...,\omega_m]\in R^{M\times m}$, while each vector $\omega_j\in R^M$ is associated with action u_j. And the polynomial BF φ_j is defined as $\varphi_j(x)=x_1^{\gamma_1^j}x_2^{\gamma_2^j}\cdots x_n^{\gamma_n^j}$.

3.2 Policy Evaluation with Q Function Approximation

With Q function approximation and policy evaluation, the calculation of parameters vector θ can be implemented using online data. Suppose current policy is $\hat{h}^{(i)}$ and try to solve parameters vector $\theta^{(i)}$. $\{(x_t,u_t,x_{t+1})\}$ denotes online data.

For each sample (x_t,u_t,x_{t+1}), combine $\hat{Q}^{(i)}$ and (4) and we have

$$[\phi(x_t,u_t)-\phi(x_{t+1},\hat{h}^{(i)}(x_{t+1}))]^T\theta^{(i)}=r(x_t,u_t). \tag{6}$$

It is obvious that (6) is a linear to $\theta^{(i)}$. Besides, online samples are collected step by step. So RLS method is applied for learning $\theta^{(i)}$. The whole learning process is presented by

$$\begin{aligned}
&z(t)=r(x_t,u_t)\\
&s(t)=\phi(x_t,u_t)-\phi(x_{t+1},\hat{h}^{(i)}(x_{t+1}))\\
&q(t)=P(t)s(t)\left[s(t)^TP(t)s(t)+1\right]^{-1}\\
&P(t+1)=\left[I-q(t)s(t)^T\right]P(t)\\
&\theta_{t+1}^{(i)}=\theta_t^{(i)}+q(t)\left[z(t)-s(t)^T\theta_t^{(i)}\right]
\end{aligned} \tag{7}$$

3.3 Policy Improvement with Policy Approximation

After $\theta^{(i)}$ is achieved, policy improvement continues to extract an improved policy $\hat{h}^{(i+1)}$. However, because of the linear parametrization for Q function and polynomials BFs, it is difficult to solve policy improvement (5) directly and have an explicit solution of $\omega^{(i+1)}$ associated with $\hat{h}^{(i+1)}$. In this way, a gradient-based method is more suitable for policy improvement, which is denoted by $\omega_j=\omega_j+\alpha\frac{\partial Q}{\partial u_j}\frac{\partial u_j}{\partial \omega_j}$, where α is the learning rate.

To guarantee the accuracy of $\hat{h}^{(i+1)}$ or $\omega^{(i+1)}$ on the whole state space Ω, a training set which is evenly distributed over the state space is defined beforehand, $\{X_s\}$, $s=1,...,N_s$. For any BF ϕ_i, its partial differential to action u_j has the following form

$$\frac{\partial \phi_i}{\partial u_j}(x,u)=\begin{cases}\beta_j^i \dfrac{\phi_i(x,u)}{u_j}, & \text{if } u_j \neq 0\\ 0, & \text{if } u_j = 0\end{cases} \tag{8}$$

So partial differential of $\hat{Q}^{(i)}$ can be formulated by $\frac{\partial \hat{Q}^{(i)}}{\partial u_j}(x,u)=\phi_{u_j}^T(x,u)\theta^{(i)}$ where $\phi_{u_j}=\left[\frac{\partial \phi_1}{\partial u_j},...,\frac{\partial \phi_N}{\partial u_j}\right]^T$. In this way, the gradient-based updating formula for $\omega_j^{(i+1)}$ on training set $\{X_s\}$ is obtained

$$\omega_{j,s}^{(i+1)}=\omega_{j,s-1}^{(i+1)}+\alpha\phi_{u_j}^T(X_s,\varphi^T(X_s)\omega_{j,s-1}^{(i+1)})\theta^{(i)}\varphi(X_s) \tag{9}$$

where $j=1,...,m$. It is noted that in order to generate an accurate parameter $\omega^{(i+1)}$, updating formula (9) on training set $\{X_s\}$ can be implemented for sufficient times.

3.4 Exploration

Exploration is necessary for online algorithm to find optimal polices. Here, we introduce ε-greedy exploration and reset scheme in [11]. At each step t, it has $1-\varepsilon_t$ probability to apply the current policy directly and ε_t probability to add uniform random exploration noise n_t to the action. Besides, at the beginning of the algorithm, the exploration probability is relatively large to encourage exploration. As the algorithm runs, the proportion of the exploitation increases. A decay exploration is denoted by $\varepsilon_t=\varepsilon_0\varepsilon_d^t$, where ε_0 is the initial value and ε_d is the decay factor with $0<\varepsilon_d<1$.

However, as the policy is admissible and the exploration noise is small, the system can still be stabilized after enough steps. At that time, the added exploration noise is not sufficient to drive the state away from the equilibrium and a new trial starting from non-equilibrium points is more benefit for the exploration. Namely, after every T_{trial} steps, the state is reset away from equilibrium.

Algorithm 1 presents our online RLSPI algorithm. It should be noted that during the implementation, no information of system dynamics is needed and the algorithm only relies on online data.

Algorithm 1. Online RLSPI algorithm

initialize $\theta^{(0)} \leftarrow 0$, $\omega^{(0)} \leftarrow h_0$, $P(0) = aI_{N\times N}$, $i = 0$ and x_0
for every step $t = 0,1,2,...$ do
$$u_t = \begin{cases} \hat{h}^{(i)}(x_t) & \text{at } 1-\varepsilon_t \text{ probability} \\ \hat{h}^{(i)}(x_t) + n_t & \text{at } \varepsilon_t \text{ probability} \end{cases}$$
apply u_t and measure next state x_{t+1} and reward r_t
policy evaluation $\theta^{(i)}$ by (7)
if $t = (i+1)K_{update}$ then
policy improvement $\omega^{(i+1)}$ using (9) on $\{X_s\}$ and $i = i+1$
end if
end for

4 Simulation Example

The nonlinear discrete-time system to test our algorithm is a mass-spring system with two states and one action. The dynamics of this system is presented in [14]. We define the reward function by a negative definite quadratic function with respect to state and action, $r(x_k, u_k) = -x_k^T Q x_k - u_k^T R u_k$, where $Q = 0.5I_{2\times 2}, R = 1$.

As the system is nonlinear, the associated Q function or policy is complicated and a plenty of BFs are required to achieve high precision. In this way, up to 6-order $\phi(x,u)$ and up to 5-order $\varphi(x)$ are used

$$\varphi(x,u) = \begin{bmatrix} x_1^2, x_2^2, u^2, x_1x_2, x_1u, x_2u, x_1^4, x_2^4, u^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, u[x_1^3, x_1^2x_2, x_1x_2^2, x_2^3], \\ u^2[x_1^2, x_1x_2, x_2^2], u^3[x_1, x_2], x_1^6, x_2^6, u^6, x_1^5x_2, x_1^4x_2^2, x_1^3x_2^3, x_1^2x_2^4, x_1x_2^5, \\ u[x_1^5, x_1^4x_2, x_1^3x_2^2, x_1^2x_2^3, x_1x_2^4, x_2^5], u^2[x_1^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, x_2^4], \\ u^3[x_1^3, x_1^2x_2, x_1x_2^2, x_2^3], u^4[x_1^2, x_1x_2, x_2^2], u^5[x_1, x_2] \end{bmatrix}^T$$

$$\varphi(x) = \left[x_1, x_2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, x_1^5, x_1^4x_2, x_1^3x_2^2, x_1^2x_2^3, x_1x_2^4, x_2^5 \right]^T$$

For this system, the initial state of each trial is set to $[-0.2, 0.2]^T$. And the length of each trial T_{trial} is 1000 steps. Besides, the policy evaluation length K_{update} is set to 200 and the training set $\{X_s\}$ selects $\{-0.1, -0.08, ..., 0.08, 0.1\}^2$. The uniform random exploration noise is limited between $[-0.1, 0.1]$. The initial exploration value ε_0 is 1 and the decay factor ε_d is 0.999977. As the system is self-stable, policy equal to 0 is adopted as the initial admissible policy.

The results of the RLSPI algorithm is presented in Fig. 1, where (a) reveals 10 representative trials during implementation, respectively at 0^{th}, 10000^{th}, 20000^{th}, …,

90000^{th} step. It is obvious that our algorithm can learn to improve the policy to a good one which stabilizes the system very well. The performance of the finial learned policy using RLSPI is presented in (b) starting from $[-0.2, 0.2]^T$. And (c) reveals scores of policies at the end of different trials in RLSPI with respect to step index. The scores of policies increase as RLSPI running.

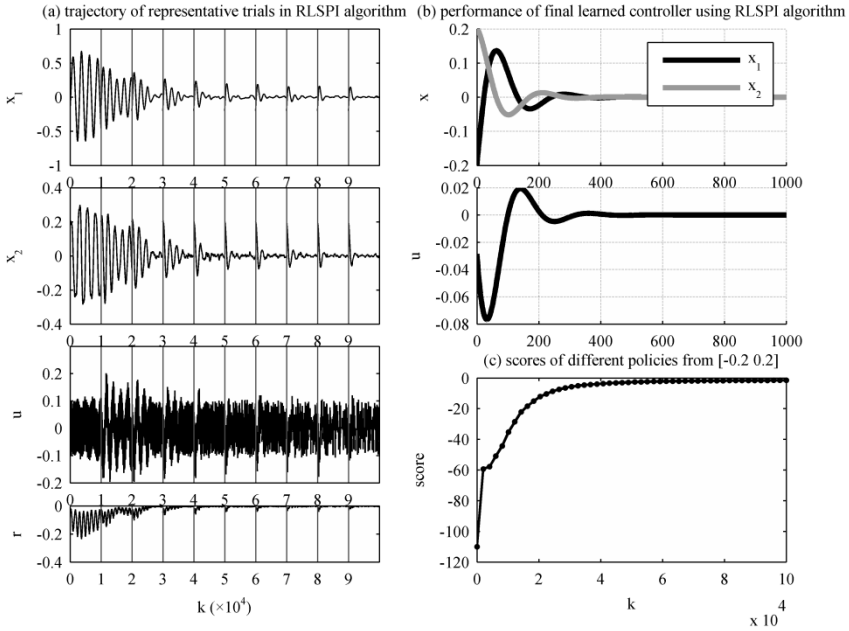

Fig. 1. Implementation of RLSPI algorithm for nonlinear discrete-time system. (a) Trajectory of representative trials in RLSPI algorithm. (b) Performance of finial learned policy using RLSPI algorithm. (c) Scores of different policies from $[-0.2, 0.2]^T$.

5 Conclusion

PI method for nonlinear discrete-time non-affine systems with continuous-state and continuous-action space is considered in this paper. Q function is introduced to approach undiscounted value function in PI. Relying on Q function, PI method does not need the information of system dynamics. Using linear parametrization, an online model-free RLSPI algorithm is proposed with RLS method and continuous policy approximation. The algorithm reveals efficiency on nonlinear discrete-time systems. Without the information of system dynamics and only relying on online data, RLSPI can learn optimal or near-optimal policies with finite steps.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
2. Wang, F., Zhang, H., Liu, D.: Adaptive Dynamic Programming: An Introduction. IEEE Comput. Intell. Mag. 4(2), 39–47 (2009)
3. Lewis, F.L., Vrabie, D.: Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control. IEEE Circuits Syst. Mag. 9(3), 32–50 (2009)
4. Howard, R.: Dynamic Programming and Markov Processes. MIT Press, Cambridge (1960)
5. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific, Belmont (1996)
6. Tsitsiklis, J.N., Van Roy, B.: Feature-Based Methods for Large Scale Dynamic Programming. Machine Learning 22, 59–94 (1996)
7. Tsitsiklis, J.N., Van Roy, B.: An Analysis of Temporal Difference Learning with Function Approximation. IEEE Trans. Automat. Contr. 42(5), 674–690 (1997)
8. Zhao, D.B., Bai, X.R., Wang, F.Y., Xu, J., Yu, W.S.: DHP Method for Ramp Metering of Freeway Traffic. IEEE Transactions on Intelligent Transportation Systems 12(4), 990–999 (2011)
9. Zhao, D.B., Hu, Z.H., Xia, Z.P., Alippi, C., Wang, D.: A Human-Like Full Range Adaptive Cruise Control Based on Supervised Adaptive Dynamic Programming. Neurocomputing (in press), http://dx.doi.org/10.1016/j.neucom.2012.09.034
10. Si, J., Wang, Y.T.: On-Line Learning Control by Association and Reinforcement. IEEE Trans. Neural Netw. 12(2), 264–276 (2001)
11. Busoniu, L., Ernst, D., De Schutter, B., Babuska, R.: Online Least-Squares Policy Iteration for Reinforcement Learning Control. In: Proc. 2010 American Control Conf. (ACC 2010), pp. 486–491 (2010)
12. Lagoudakis, M.G., Parr, R.: Least-Squares Policy Iteration. Journal of Machine Learning Research 4, 1107–1149 (2003)
13. Abu-Khalaf, M., Lewis, F.L.: Nearly Optimal Control Laws for Nonlinear Systems with Saturating Actuators Using a Neural Network HJB Approach. Automatica 41(5), 779–791 (2005)
14. Zhang, H., Luo, Y., Liu, D.: Neural-Network-Based Near-Optimal Control for a Class of Discrete-Time Affine Nonlinear Systems with Control Constraints. IEEE Trans. Neural Netw. 20(9), 1490–1503 (2009)