Least Square Policy Iteration in Reinforcement Learning

Haifei Zhang [1,2]
[1]Dept. of Mechanical and Electrical Engineering
Jiangsu College of Engineering and Technology
Nantong, China
[2]College of Computer and Information
Hohai University
Nanjing, China
e-mail: zhanghaifei@hhu.edu.cn

Bin Zhao [1]
[1]Dept. of Mechanical and Electrical Engineering
Jiangsu College of Engineering and Technology
Nantong, China
e-mail: zhaobin@jcet.edu.cn

Hailong Deng [1]
[1]Dept. of Mechanical and Electrical Engineering
Jiangsu College of Engineering and Technology
Nantong, China
e-mail: dhl@jcet.edu.cn

Ying Hong [3]
[3]Center of International Education and Exchange
Jiangsu College of Engineering and Technology
Nantong, China
e-mail: hongying@jcet.edu.cn

Abstract—Policy iteration is the core procedure for solving problems of reinforcement learning method. Policy iteration evaluates polices by evaluating value functions of these polices and then new improvement polices will be figured out by these value functions. Value functions and polices in classic policy iteration are tabular and accurate. However, these are not suitable for problems in extensive and continuous, i.e. action space reinforcement learning. Therefore, approximate policy iteration is often used to solving the problems. It constructs approximate value function for present policy and becomes an important part of approximate policy iteration. Policy is expressed by instantly calculating policy action from approximate function rather than explicit expression. Least square reinforcement method is sample-effective in solving parameters approximating the value function, the larger the sample size, the faster the speed of approaching solution. This paper will discuss the online least square policy iteration algorithms in reinforcement learning.

Keywords- Policy iteration; Least Square; Reinforcement learning; Sample-effective; Policy improvement

I. INTRODUCTION

Policy iteration is the core procedure for solving problems of reinforcement learning method. Policy iteration evaluates polices by evaluating value functions of these polices and then new improvement polices will be figured out by these value functions. Value functions and polices in classic policy iteration are tabular and accurate. However, these are not suitable for problems in extensive and continuous, i.e. action space reinforcement learning. Therefore, approximate policy iteration is often used to solving the problems.

The linear parameterization is the most effective arithmetic for approximate policy iteration expressing value function. It acquires parametric linear system of equations by Bellman equation linear which satisfies value function. In order to acquire parameter close to value function, equations set is solved by least square sample in the way of once or multiple iteration.

II. MAIN PRINCIPLES AND CLASSIFICATION

The problems in large and continuous state-action spaces, value functions cannot be precisely expressed, only approximately. The solution to Bellman equation cannot be expressed by chosen approximator, but solved approximately instead. The two classifications of least square method for policy evaluation are distinguished by the approach to approximate solution to Bellman equation, as is respectively shown in Fig.1 Projected policy evaluation and Bellman residual minimization (BRM). According to the times of iteration, projected policy evaluation can be classified into Least-squares temporal difference (LSTD [1] [2]) and Least-squares policy evaluation (LSPE [3]).

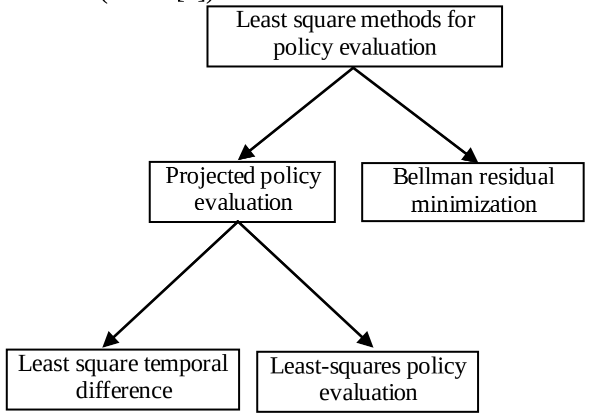
Least square methods for policy evaluation
Projected policy evaluation
Bellman residual minimization
Least square temporal difference
Least-squares policy evaluation

Figure 1. Classifications of least square methods for policy iteration

In classical policy evaluation, Bellman equation Q^π for Q function of policy π_0 (formular) can be briefly expressed as:

$$Q^\pi = B_Q^\pi(Q^\pi) \quad (1)$$

B_Q^π is called as Bellman map and also called Back-up Operator. It is shown as follows:

$$\begin{aligned}[B_Q^\pi(Q^\pi)](s,a) &= \sum_{s'} P_{ss'}^a (R_{ss'}^a + \gamma Q(s', \pi(s'))) \\ &= E_{s' \sim T(s,a,\bullet)}(R(s,a,s') + \gamma Q(s', \pi(s')))\end{aligned} \quad (2)$$

Projected policy evaluation looks for a $\hat{Q}$ and approximately equals to the projection of its updated version $B_Q^\pi(\hat{Q})$ on the space of Q function:

$$\hat{Q} \approx \Pi(B_Q^\pi(\hat{Q})) \quad (3)$$

$\Pi : \mathcal{Q} \to \hat{\mathcal{Q}}$ denotes projections from the space $\mathcal{Q}$ of all Q functions onto the space of $\hat{\mathcal{Q}}$ of denotable Q-functions. The solution to the equation equals the minimum of the distance between original $\hat{Q}$ and $\Pi(B_Q^\pi(\hat{Q}))$, that is minimizing the Mean Squared Projected Bellman Error (MSPBE):

$$\min_{\hat{Q} \in \hat{\mathcal{Q}}} MSPBE(\hat{Q}) = \min_{\hat{Q} \in \hat{\mathcal{Q}}} \| \hat{Q} - \Pi(B_Q^\pi(\hat{Q})) \|_w^2 \quad (4)$$

$w : S \times A \to [0,1]$ is a given weighted function.

$\| \bullet \|$ denotes a universal norm (or measurement). The Eq.3 is called the Projected Bellman Equation, hence the name projected policy evaluation.

Two subclasses methods are included in projected policy evaluation. One is One-shot methods namely least square temporal difference aiming direct solution to projected Bellman equation (LSTD). The other is least square policy evaluation by iteration to solute (LSPE)

Compared with projected policy evaluation, methods for Bellman residual minimization (BRM) don't adopt projection, but try to minimize the Mean Squared Bellman Error (MSBE) in an approximate sense:

$$\min_{\hat{Q} \in \hat{\mathcal{Q}}} MSBE(\hat{Q}) = \min_{\hat{Q} \in \hat{\mathcal{Q}}} \| \hat{Q} - B_Q^\pi(\hat{Q}) \|_w^2 \quad (5)$$

Directly solve the Bellman equation:

$$\hat{Q} \approx B_Q^\pi(\hat{Q}) \quad (6)$$

$\hat{Q} - B_Q^\pi(\hat{Q})$ is called Bellman residual, hence the name Bellman residual minimization. This article focuses on projected policy evaluation, and its model-free implementations.Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your word processor, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 or Open Type fonts are preferred. Please embed symbol fonts, as well, for math, etc.

III. The Linear Case and Matrix of the Equations

To formally define parametric approximate linear and Euclid norm, the state and the actions spaces will be assumed finite, $S = \{s_1, s_2, ..., s_m\}$, $A = \{a_1, a_2, ..., a_n\}$. The practical algorithms below can also be applied to infinite and continuous state-action spaces.

The Q-function represented by a linearly parameter can have the following form:

$$\hat{Q}(s,a) = \sum_{l=1}^{d} \varphi_l(s,a)\theta_l = \phi^T(s,a)\theta \quad (7)$$

where $\theta \in R^d$ is the parameter vector and $\varphi(s,a)$ is the vector of basis function, also called feature function. In order to use matrix to calculate, basis function is often transfer into row vector $\phi = [\varphi_1(s,a), \varphi_1(s,a), ..., \varphi_d(s,a)]^T$.

Given a weight function $w : S \times A \to [0,1]$, Q-function is the root mean square of Q value of every state-action. The squared weighted Euclidean norm can be defined by

$$\| Q \|_w^2 = \sum_{\substack{i=1,......,m \\ j=1,......,n}} w(s_i, a_j) | Q(s_i, a_i) |^2 \quad (8)$$

In projected policy iteration, the best solution is the average value of quadratic sum of discrepancy between approximation $\hat{Q}$ and true valus Q, that is Mean Squared Error (MSE) between $\hat{Q}$ and Q.

$$\Pi^w(Q) = \arg\min_{\hat{Q}} \| \hat{Q} - Q \|_w^2 \quad (9)$$

By (7) and (8):

$$\begin{aligned}\Pi^w(Q) &= \arg\min_{\hat{Q}} \| \hat{Q} - Q \|_w^2 = \arg\min_{\hat{Q}} \| \hat{Q}(s,a) - Q(s,a) \|_w^2 \\ &= \arg\min_{\hat{Q}} \sum_{\substack{i=1,......,m \\ j=1,......,n}} w(s_i, a_j) | \hat{Q}(s_i, a_j) - Q(s_i, a_j) |^2\end{aligned} \quad (10)$$

A. Matrix Form of the Bellman Mapping

Since the state-action square is discrete, the Bellman mapping can be written as a sum:

$$[B_Q^{\pi}(Q)](s_i,a_j)=\sum_{s_{i'}}T(s_i,a_j,s_{i'})[R(s_i,a_j,s_{i'})+\gamma Q(s_{i'},\pi(s_{i'})]$$
$$=\sum_{i'=1}^{M}T(s_i,a_j,s_{i'})R(s_i,a_j,s_{i'})+\gamma\sum_{i'=1}^{M}T(s_i,a_j,s_{i'})Q(s_{i'},\pi(s_{i'})) \quad (11)$$

The two-sum expression is written in a matrix form:

$$B_Q^{\pi}(Q)=R+\gamma T^{\pi}Q \quad (12)$$

where $B_Q^{\pi}:R^{MN}\to R^{MN}$, $[i,j]$ denotes a numerical subscript of a matrix, $[i,j]=i+(j-1)M$. The rest vectors and matrices are defined as follows:

$Q\in R^{MN}$ *is* a vector representation of the Q-function Q, $Q_{[i,j]}=Q(s_i,a_j)$.

R *is* a vector representation of the expectation of the reward function R, $R_{[i,j]}=\sum_{i'}T(s_i,a_j,s_{i'})R(s_i,a_j,s_{i'})$.

T^{π} is a matrix representation of the transition function combined with the policy $T_{[i,j][i',j']}=T(s_i,a_j,s_{i'})\bullet\pi(s_{i'},a_{j'})$, and 0 otherwise. The universal Q-vector in (12) is replaced by approximate parameterized Q vector and the matrix defining basis function is as follows:

$$\phi_{[i,j],l}=\varphi_l(s_i,a_j),\phi\in R^{MN\times d} \quad (13)$$

According to (7), approximate Q vector can be written as $\hat{Q}=\phi\theta$, substituting (12):

$$B_Q^{\pi}(\phi\theta)=R+\gamma T^{\pi}\phi\theta \quad (14)$$

B. *Projected Policy Evaluation*

We are interested in estimating parameters θ that yield a value function $\hat{Q}_{\theta}$ as close as possible to the projected Bellman equation representable function $\prod(B_Q^{\pi}(\hat{Q}_{\theta}))$, this goal directly corresponds to minimizing the mean squared projected Bellman error (MSPBE).

$$MSPBE(\theta)=\|\hat{Q}_{\theta}-\prod(B_Q^{\pi}(\hat{Q}_{\theta}))\|_w^2 \quad (15)$$

Under proper condition on the basis functions and the weights w and according to the (3), the projected Bellman equation can be exactly solved in the linear case:

$$\hat{Q}_{\theta}=\prod^{w}(B_Q^{\pi}(\hat{Q}_{\theta})) \quad (16)$$

Then it turns into a minimum problem of (4). The Bellman equation in the matrix form can be expressed as follows:

$$\phi\theta=\prod^{w}(B_Q^{\pi}(\phi\theta))$$
$$=\prod^{w}(R+\gamma T^{\pi}\phi\theta) \quad (17)$$

Where the weighted least square projection $\prod^{w}$ is a close form and its matrix form can be expressed as follows:

$$\prod^{w}=\phi(\phi^T w\phi)^{-1}\phi^T w \quad (18)$$

The weight matrix w collects the weights of each state-action on its main diagonal.

$$w_{[i,j],[i,j]}=w(s_i,a_i),w\in R^{MN\times MN} \quad (19)$$

After substituting (18) into (17), multiplying $\phi^T w$ on both sides, we obtain the following by rearrangement:

$$\phi^T w\phi\theta=\gamma\phi^T wT^{\pi}\phi\theta+\phi^T wR$$

Introduce matrix $A,B\in R^{n\times n}$ and vector $b\in R^n$:

$$A=\phi^T w\phi, B=\phi^T wT^{\pi}\phi, b=\phi^T wR$$

Projected Bellman equation can be written as [4]:

$$A\theta=\gamma B\theta+b \quad (20)$$

Thus Bellman equation can be represented and solved by low dimensional matrix and vector.

The idealized LSTD-Q (Least Square Time Difference of Q function) belongs to the first ('one shot') subclass of methods for projected policy evaluation methods in Fig.1. It only solves the (20) to obtain the parameter vector θ and this parameter vector provides an approximate Q function $\hat{Q}^{\pi}(s,a)=\phi^T(s,a)\theta$ (Eq.7 is shown) of the considered policy π.

The idealized LSPE-Q (Least Square Policy Evaluation of Q function) is an iterative algorithm, belonging to the second subclass of projected policy evaluation methods in Fig.1. It still relies on (21), but needs to be updated gradually.

$$\theta_{l+1}=\theta_l+\alpha(\theta_{l+1}'-\theta_l)$$
$$where\quad A\theta_{l+1}'=\gamma B\theta_l+b \quad (21)$$

Starting from initial value θ_0, α is a positive step size parameter.

Consider a sample set $\{(s_i,a_j,s_{i'},r_i)\,|\,1\le i\le M,1\le j\le N,1\le i'\le M\}$, $s_{i'}$ is drawn from state transition function $T(s_i,a_j,\bullet)$, $r_i=R(s_i,a_j,s_{i'})$, the state action pairs (s_i,a_j) are drawn from the distribution given by the weight function w.

The matries A, B and the vector b in (20) can also be written in term of data sum [4]:

$$A=\sum_{i=1}^{M}\sum_{j=1}^{N}[\phi(s_i,a_j)w(s_i,a_j)\phi^T(s_i,a_j)],$$
$$B=\sum_{i=1}^{M}\sum_{j=1}^{N}[\phi(s_i,a_j)w(s_i,a_j)\sum_{i'=1}^{M}(T(s_i,a_j,s_{i'})\phi^T(s_{i'},\pi(s_{i'})))],$$
$$b=\sum_{i=1}^{M}\sum_{j=1}^{N}[\phi(s_i,a_j)w(s_i,a_j)\sum_{i'=1}^{M}(T(s_i,a_j,s_{i'})R(s_i,a_j,s_{i'}))] \quad (22)$$

Solving matrix form of projected Bellman equation can use sample data to estimate A, B and b. According to (22), the evaluation value of A, B and b can be solved(l_s is the serial number of present sample action-state pair):

$$\begin{aligned} &A_0=0, B_0=0, b_0=0,\\ &A_{l_s+1}=A_{l_s}+\phi(s_{l_s},a_{l_s})\phi^T(s_{l_s},a_{l_s}),\\ &B_{l_s+1}=B_{l_s}+\phi(s_{l_s},a_{l_s})\phi^T(s_{l_s'},\pi(s_{l_s'})),\\ &b_{l_s+1}=b_{l_s}+\phi(s_{l_s},a_{l_s})r_{l_s'} \end{aligned} \quad (23)$$

Use Eq.23 to solve n sample data and in LSTD-Q algorithm process search approximate parameter vector θ by (24):

$$\frac{1}{n}A_n\theta=\gamma\frac{1}{n}B_n\theta+\frac{1}{n}b_n \quad (24)$$

equivalently:

$$\frac{1}{n}(A_n-\gamma B_n)\theta=\frac{1}{n}b_n \quad (25)$$

The divisions by n on both sides of the formula, not mathematically necessary, but increase the numerical stability of the algorithm by preventing the coefficients from growing too large when more samples are processed. That means it is very possible the larger the number of samples, the larger the value of elements of A, B and b. Eventually it easily leads to the stability of algorithm convergence value.

A widely used policy iteration will be obtained by combining LSTD-Q policy evaluation algorithm with policy improvement. It is called as Least-squares policy iteration (LSPI).

Another LSTD-Q policy evaluation algorithm with eligibility traces can accelerate convergence, called LSTD(λ)-Q. The Approximate function parameter TD(λ) updating rule:

$$\begin{aligned}\theta_{l_s+1}=\theta_{l_s}+\alpha_{l_s}(r_{l_s}+\gamma\phi^T(s_{l_s+1},a_{l_s+1})\theta_{l_s+1}\\ -\phi^T(s_{l_s},a_{l_s})\theta_{l_s})\nabla_{\theta_{l_s}}Q(s_{l_s},a_{l_s})\end{aligned} \quad (26)$$

The eligibility traces vector is:

$$\vec{e}_{l_s}=\gamma\lambda\vec{e}_{l_s-1}+\nabla_{\theta_{l_s}}Q(s_{l_s},a_{l_s})=\gamma\lambda\vec{e}_{l_s-1}+\phi(s_{l_s},a_{l_s}) \quad (27)$$

The evaluation value of A, B and b can be solved:

$$\begin{aligned}&A_{l_s+1}-\gamma B_{l_s+1}=\vec{e}_{l_s+1}(\phi^T(s_{l_s},a_{l_s})-\gamma\phi^T(s_{l_s+1},a_{l_s+1})),\\ &b_{l_s+1}=\vec{e}_{l_s+1}r_{l_s'},\\ &\theta=(A_{l_s+1}-\gamma B_{l_s+1})^{-1}b_{l_s+1}\end{aligned} \quad (28)$$

LSPE-Q and LSTD-Q use the same evaluation A, B and b, but LSPE-Q is iteration updating parameter vector. LSPE-Q starts from random initial parameter vector and updates according to:

$$\begin{aligned}&\theta_{l_s+1}=\theta_{l_s}+\alpha(\theta_{l_s+1}'-\theta_{l_s})\\ &where\quad \frac{1}{l_s+1}\theta_{l_s+1}'A_{l_s+1}=\gamma\frac{1}{l_s+1}B_{l_s+1}\theta_{l_s}+\frac{1}{l_s+1}b_{l_s+1}\end{aligned} \quad (29)$$

This update is an approximate sample-based version compared with idealized version (21). Similarly to LSTD-Q, the division by l_s+1 is to increases the stability of the updates. At the beginning of learning process, A is invertible unless a few samples have been processed. The solution to the problem is to initialize A to a small multiple of the identity matrix.

IV. Online Least Square Policy Iteration

The application of least square methods to online learning is an important issue. Unlike in the offline case of the final performance, the performance will improve once every few transition samples in the online learning. Before a precise evaluation of the current policy is completed, policy iteration can take this requirement into account by performing policy improvement once every few transition samples. Such iteration method is called Partially Optimistic policy iteration [2] [5]. In the extreme, fully optimistic case, the policy is improved after every single transition. Optimistic policy updates were combined with LSTD-Q [6] [7], therefore obtaining optimistic LSPI, also with LSPE-Q [8] [9]. Li and others explored a non-optimistic, more computationally involved method to online iteration [10], in which LSPI is fully executed between consecutive sample-collection episodes.

```
Algorithm 1 Adopts LSTD-Q evaluation method to Online LSPI algorithm
1: input: initial policy $\pi_0$, basis function $\varphi$, policy
          improvement interval $L$, $exploration$, small
          constant $\delta_A > 0$
2: $k \leftarrow 0$
3、 $l_s \leftarrow 0, A_{l_s} \leftarrow \delta_A I, B_{l_s} \leftarrow 0, b_{l_s} \leftarrow 0$
4: observe initial state $s_{l_s}$
5: Repeat
6:    $a_{l_s} \leftarrow \pi_k(s_{l_s}) + exploration$
7:    apply $a_{l_s}$, observe nest state $s_{l_s+1}$ and reward $r_{l_s+1}$
```

```
8:  $A_{l_s+1} = A_{l_s} + \phi(s_{l_s}, a_{l_s})\phi^T(s_{l_s}, a_{l_s})$
9:  $B_{l_s+1} = B_{l_s} + \phi(s_{l_s}, a_{l_s})\phi^T(s_{l_s+1}, \pi(s_{l_s+1}))$
10: $b_{l_s+1} = b_{l_s} + \phi(s_{l_s}, a_{l_s})r_{l_s+1}$
11: IF $l_s + 1 = (k+1)L$ then
12: solve $\frac{1}{l_s}(A_{l_s} - \gamma B_{l_s})\theta_k = \frac{1}{l_s}b_{l_s}$
13:       $\pi_{k+1}(s) = \arg\max_{a\in A} \phi^T(s,a)\theta_k, \forall s$
15:       $k \leftarrow k+1$
16:  $l_s \leftarrow l_s + 1$
17: Until satisfy the final requirement set
```

Algorithm 1 presents the online LSPI based on LSTD-Q [7]. The same matrix and vector estimates are used as in offline LSTD-Q and LSPI, but there are great differences. First, online LSPI algorithm collects its own samples by using its current policy to interact with the system. It means that exploration must be added on top of the deterministic policy. Second, it is unnecessary for algorithm improvement on policy to wait for the estimate A、B and b to get close to approximation for current policy. In addition, these estimates continue to be updated without being reset after policy changes. So in fact they correspond to multiple polices. The underlying assumption here is that A、B and b are similar to subsequent policies. A more computationally costly alternative would be to store samples and rebuild the estimates from the beginning, but it may be unnecessary in practice.

The transition number L between consecutive policy improvement is an important parameter of algorithm. For example, when L=1 , online LSPI is fully optimistic. Generally speaking, L cannot be too large to avoid potentially bad policies from being used too long. Note that in offline case, improved policies should not have to be explicitly computed in online LSPI, but can be computed on demand.

Algorithm 2 adopts minimization iteration algorithm of LSPE-Q evaluation method.

```
Algorithm 2 Adopts LSPE-Q evaluation method to Online LSPI algorithm
1: input: initial policy $\pi_0$, basis function $\varphi$, policy
          improvement interval $L$, $exploration$, small
          constant $\delta_A > 0$
2: $k \leftarrow 0$
3、 $l_s \leftarrow 0, A_{l_s} \leftarrow \delta_A I, B_{l_s} \leftarrow 0, b_{l_s} \leftarrow 0$
4: observe initial state $s_{l_s}$
5: Repeat
6:   $a_{l_s} \leftarrow \pi_k(s_{l_s}) + exploration$
7:  apply $a_{l_s}$, observe nest state $s_{l_s+1}$ and reward $r_{l_s+1}$
8:   $A_{l_s+1} = A_{l_s} + \phi(s_{l_s}, a_{l_s})\phi^T(s_{l_s}, a_{l_s})$
9:   $B_{l_s+1} = B_{l_s} + \phi(s_{l_s}, a_{l_s})\phi^T(s_{l_s+1}, \pi(s_{l_s+1}))$
10:  $b_{l_s+1} = b_{l_s} + \phi(s_{l_s}, a_{l_s})r_{l_s+1}$
11:  $\theta_{l_s+1} = \theta_{l_s} + \alpha(\theta_{l_s+1}' - \theta_{l_s})$
       where $\frac{1}{l_s+1}\theta_{l_s+1}'A_{l_s+1} = \gamma\frac{1}{l_s+1}B_{l_s+1}\theta_{l_s} + \frac{1}{l_s+1}b_{l_s+1}$
12: IF $l_s + 1 = (k+1)L$ then
13:       $\pi_{k+1}(s) = \arg\max_{a\in A} \phi^T(s,a)\theta_{l_s+1}, \forall s$
15:       $k \leftarrow k+1$
16:  $l_s \leftarrow l_s + 1$
17: Until satisfy the final requirement set
```

Currently the most widely used and simplest exploration rule is $\varepsilon - greedy$. In every time step l_s, the greedy action $1-\varepsilon_{l_s}$ [11] in random exploratory action the of probability $\varepsilon_{l_s} \in [0,1]$ is adopted. As l_s increases, ε_{l_s} reduces gradually. Current approximate best policy will be more used in algorithm.

V. Conclusions

The most effective arithmetic for approximate policy iteration expressing value function is linear parameterization. It acquires parametric linear system of equations by Bellman equation linear which satisfies value function. In order to acquire parameter close to value function, equations set is solved by least square sample in the way of once or multiple iteration.

Because efficient numerical method can be used to solve these equation sets, least square reinforcement is effective calculation. In addition, a monolithic rapid convergence algorithm will be obtained by the common rapid convergence of policy iteration method. More importantly, least square reinforcement method is sample-effective. The larger the sample size, the faster the speed of approaching solution. It is a very important characteristic for reinforcement learning in realistic system because of tough acquirement of sample data.

Acknowledgment

This work is sponsored by the following grants: (i) Excellent Youth Scholar of “Qing Lan Project” of Jiangsu Province in China (2012), (ii) Domestic senior visiting scholar program of higher vocational education in Jiangsu province (No. 2014FX018), and (iii) a grant No. 61163057 from the National Natural Science Foundation of China.

References

[1] Bradtke, S.J., Barto, A.G.: Linear least-squares algorithms for temporal difference learning. Machine Learning, 1996, 22(1-3), p. 33–57

[2] Boyan, J.: Technical update: Least-squares temporal difference learning. Machine learning, 2002(49), p. 233-246

[3] Bertsekas, D.P., Ioffe, S.: Temporal differences-based policy iteration and applications in neuro-dynamic programming. Tech. Rep. LIDS-P-2349, Massachusetts Institute of Technology, Cambridge, US (1996), http://web.mit.edu/dimitrib/www/Tempdif.pdf

[4] Lagoudakis, M. G. and Parr, R. Least-squares policy iteration. Journal of Machine Learning Research, 2003(4), p. 1107-1149

[5] Sutton, R.S.. Learning to predict by the method of temporal differences. Machine Learning, 1988(3), p. 9-44

[6] Buşoniu, L., De Schutter, B., Babuška, R., Ernst, D.: Using prior knowledge to accelerate online least-squares policy iteration. In: 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR-2010), Cluj-Napoca, Romania (2010).

[7] Buşoniu, L., Ernst, D., De Schutter, B., Babuška, R.: Online least-squares policy iteration for reinforcement learning control. In: Proceedings 2010 American Control Conference (ACC-2010), Baltimore, US (2010), pp. 486–491

[8] Jung, T., Polani, D.: Kernelizing LSPE(λ). In: Proceedings 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL-2007), Honolulu, US (2007), pp. 338–345

[9] Jung, T., Polani, D.: Learning RoboCup-keepaway with kernels. In: Gaussian Processes in Practice, JMLR Workshop and Conference Proceedings, 2007, vol. 1, pp. 33–57

[10] Li, L., Littman, M.L., Mansley, C.R.. Online exploration in least-squares policy iteration. In: Proceedings 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2009), Budapest, Hungary, 2009, vol. 2, pp. 733–739

[11] R. S. Sutton, A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, 1998