The Thirteenth International Symposium on Artificial Life and Robotics 2008(AROB 13th '08),
B-Con Plaza, Beppu, Oita, Japan, January 31-February 2, 2008

An effective reinforcement learning with automatic construction of basis functions and sequential approximation

Nobuhito Nanjoh, Takeshi Mori
Guraduate School of Information Science
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, 630-0192, Japan

Shin Ishii
Guraduate School of Informatics
Kyoto University
Gokasho, Uji, Kyoto, 611-0011, Japan

Abstract

In reinforcement learning, a large state space makes the value function estimation infeasible, and hence it is often approximated as a linear combination of basis functions whose linear coefficients constitute the parameter. However, the basis function construction needs some prior knowledge and is often troublesome. To overcome this difficulty, Keller et al. proposed an automatic basis function construction technique [2], but it may cause serious computational cost. We propose a novel approach to this context, where the computational cost can be drastically reduced from that of the existing method, due to our sequential approximation.

1 Introduction

Reinforcement learning (RL) is a machine learning method through interactions with environments to maximize the long-term reward accumulation (return). In usual RL schemes, it is important to estimate the value function which predicts the return starting form each state. In many realistic RL problems, however, a large state space makes the value function estimation in its original function space infeasible, and hence, the value function approximation using a parametric linear model comes to be important; the value function is represented as a linear combination of basis functions whose linear coefficients constitute the parameter. While the parameter is obtained by the conventional RL methods based on sample data, the basis functions must be determined by a designer at hand in advance, but the determination is very difficult due to the small amount of available prior knowledge. A poor setting of basis functions may increase the potential error of approximating the value function, and a large approximation error makes the policy's control fail.

One possible idea to deal with this problem is to construct the basis functions automatically based on the available data in an online manner. Keller et al. proposed an automatic basis function construction technique [2], in which appropriate basis functions were generated based on temporal difference (TD) error. However, this approach had some difficulties. First, the basis functions were generated in each time step, and then, all the basis functions, whose number could be huge, were used to approximate the value function by means of the least-squares optimization. This may cause serious computational cost, as the number of bases increases. Second, the value function approximation can be unstable due to the singularity in the least squares solution, especially as the number of highly correlated basis functions increases.

To overcome these problems, in this study, we propose a novel approach to the automatic construction of basis functions, where the number of basis functions and the computational cost can be drastically reduced from those of the existing method. According to our method, the basis functions, are constructed in to approximate the TD error at each time step instead of the value function, and the learned parameters are fixed in the subsequent optimizations. In other words, our method decomposes whole the approximation problem in the Markov decision processes (MDPs) into small approximation problems which are easily to be solved with a small number of basis functions, so that the parameter is optimized within each sub-problem. This method has some theoretical foundations, given by Bertsekas and Castanon [3], which showed that the sequential calculation of analytic approximation of the TD error with appropriate basis functions can geometrically reduce the approximation error along the whole time-series. Computer simulation shows our method could drastically reduce not only the number of basis functions but also the computational time in comparison to the Keller et al.'s method [2].

2 MDPs and the value function approximation

We consider finite MDPs, which is composed of the discrete time t, a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a set of transition probabilities $\mathcal{P}$, and a scalar reward $r \in \mathcal{R}$. At time t, the agent selects an action $a_t \in \mathcal{A}$ according to a

stationary policy π at a state $s_t \in \mathcal{S}$, and then it moves to the next state $s_{t+1} \in \mathcal{S}$ and simultaneously recieves the reward r_{t+1}. The objective is to find a policy that maximizes the value function:

$$V\ (s) = E\left[\sum_{t=0}^{\infty}\ {}^{t}r_{t+1}|s_0 = s\right] \tag{1}$$

where $\in [0,1)$ is a discount factor.

One of the approaches of searching for the optimal policy is policy iteration [1]. The policy iteration is constructed of two steps, i.e., the policy evaluation step and the policy improvement step. In the policy evaluation step, the value function V for the current policy π is calculated or approximated. In the policy improvement step, the policy π is improved by using the learned value function. In this article, we focus on the policy evaluation step. The Bellman equation under π is defined as

$$V\ (s) = E\ [r_{t+1} + \ V\ (s_{t+1})|s_t = s]\,. \tag{2}$$

For the sake of later convenience, we introduce the matrix notations; the value function vector $\boldsymbol{V}\ \in \mathcal{R}^{|\mathcal{S}|}$ whose i-th element is the value at the i-th state in $\mathcal{S}$, the state transition matrix $\boldsymbol{P} \in \mathcal{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ whose i,j-th element is the state transition probability under policy π from the i-th state to the j-th state in $\mathcal{S}$, and the reward vector $\boldsymbol{r} \in \mathcal{R}^{|\mathcal{S}|}$ whose i-th element is the expected reward with respect to the next state conditioned on the i-th state and policy π. Then, the Bellman equation Eq.(2) is represented as $\boldsymbol{V}\ = \boldsymbol{r} + \ \boldsymbol{PV}\ $.

In the dynamic programming context, the policy evaluation in average operates successive approximations of the value function $\boldsymbol{V}$ using Bellman operator $T(\cdot)$:

$$\boldsymbol{V}^{k+1} = T(\boldsymbol{V}^k) \equiv \boldsymbol{r} + \ \boldsymbol{PV}^k, \tag{3}$$

where k denotes an iteration step and $\boldsymbol{V}^k$ converges to $\mathbf{V}$ in the limit of $k \to \infty$ [1]. In usual RL settings, the vector $\mathbf{r}$ and the matrix $\boldsymbol{P}$ are unknown, so the value function is estimated by using sample trajectory, $\langle s_0, r_1, s_1, r_2, \ldots, s_t, r_{t+1}, \ldots\rangle$.

In realistic RL problems, however, the state space is large, and the value function estimation is very difficult. Then the value function is often approximated by using a paramatric linear model, that is, represented as a linear combination of basis function whose linear coefficients constitute the parameter:

$$V(s) \approx \sum_{m=1}^{M} \phi_m(s)'\theta_m = \ \ (s)'\boldsymbol{\theta}, \tag{4}$$

where (') is the transpose, and the basis function vector (s) and the parameter $\boldsymbol{\theta}$ are both M-dimensional vectors. Note that the designer of the system must prepare the basis functions at hand. The parameter $\boldsymbol{\theta}$ is tuned in the policy evaluation step by using the sample trajectory. One of the learning methods is least-squares TD learning [4], which is the closed-form estimation method for the linearly-approximated value function.

3 Previous method for automatic basis function construction and aggregation theory

In usual, the basis function construction requires designer's trial and error. The poor setting of basis functions may increase the approximation error of the value function, and a large approximation error makes the policy's update fail. One possible idea to deal with this problem is to construct the basis functions automatically based on the available data in an online manner.

Keller et al. [2] proposed an automatic basis function construction technique, in which appropriate basis functions were generated based on the aggregation theory, proposed formerly by Bertsekas and Castanon [3]. According to this theory, automatically produced aggregated states are used to speed up the value iteration in dynamic programming, even if the original operations of the Bellman operater (Eq.(3)) are very slow. In section 5, we show that our proposal method is also based on this theory, actually a more appropriate application of this theory rather than Keller et al.'s. In this theory, the Bellman operator (Eq.(3)) is replaced by aggregation iterations of the form:

$$\boldsymbol{V}^{k+1} = \boldsymbol{V}^k + \ \ \boldsymbol{y}, \tag{5}$$

where $\in \{0,1\}^{|\mathcal{S}|\times M}$ is such an aggregation matrix that $_{ij} = 1$ if state i is assigned to group j or 0 otherwise. The grouping of the states is determined by the ranking of the Bellman residual: $T(\boldsymbol{V}^k) - \boldsymbol{V}^k$. $\boldsymbol{y}$ is the parameter vector corresponding to , which is derived by analytical calculation. $\boldsymbol{y}$ is the orthogonal projected vector of the Bellman residual onto the subspace spanned by sparse matrix . By using the Bellman operator T,

$$T(\boldsymbol{V}^{k+1}) = T(\boldsymbol{V}^k) + \ \ \boldsymbol{P}\ \ \boldsymbol{y}, \tag{6}$$

is obtained, and by using the definition of orthogonal projection: $\Pi = \ \ (\ \ ')^{-1}\ \ '$, the following holds [3]:

$$T(\boldsymbol{V}^{k+1}) - \boldsymbol{V}^{k+1} = \boldsymbol{g}_1 + \boldsymbol{g}_2, \tag{7}$$

where

$$\begin{aligned} \boldsymbol{g}_1 &= (\boldsymbol{I} - \Pi)\left(T(\boldsymbol{V}^k) - \boldsymbol{V}^k\right), \\ \boldsymbol{g}_2 &= \ \ (\boldsymbol{I} - \Pi)\boldsymbol{P}\ \ \boldsymbol{y}. \end{aligned}$$

Based on the above ranking scheme of the Bellman residual $T(\boldsymbol{V}^k) - \boldsymbol{V}^k$, the maximum norm of the first approximation error $\boldsymbol{g}_1$ can be minimized under a restricted number

of groups M. Moreover, it is know experimentally that $\boldsymbol{g}_2$ often decreases as $\boldsymbol{g}_1$ decreases [3]. So we may focus only on the minimization of $\boldsymbol{g}_1$.

In the Keller et al.'s method, by applying this theory to an RL context, the value function is updated as

$$\begin{aligned} \boldsymbol{V}_{k+1} &= \Phi_k \boldsymbol{\theta}_k + \quad \boldsymbol{\theta}_{\text{new}} \\ &= [\Phi_k \quad] \begin{matrix} \boldsymbol{\theta}_k \\ \boldsymbol{\theta}_{\text{new}} \end{matrix} = \Phi_{k+1}\boldsymbol{\theta}_{k+1}, \end{aligned} \tag{8}$$

where $\Phi_k \in \{0,1\}^{|\mathcal{S}|\times kM}$ is the basis function matrix: $\Phi_k = [\ _k(1),\ _k(2),\ldots,\ _k(|\mathcal{S}|)]'$, $_k(s) \in \{0,1\}^{kM}$ is a kM-dimensional binary vector corresponding to a state s, and M is the number of basis functions produced in each step. In this method, the basis functions are constructed iteratively based on the TD error (the Bellman residual for a sampled state), and the parameters are evaluated by least-squares TD learning (LSTD) [4]. By using a sample trajectory, the parameter vector $\boldsymbol{\theta}_{k+1}$ is optimized so as to minimize the cost function:

$$\begin{aligned} J_{\text{LSTD}}(\boldsymbol{\theta}_{k+1}) = \ & \tfrac{1}{2t}\textstyle\sum_{i=0}^{t-1}(r_{i+1} - \ V^k(s_{i+1}) \\ & - \ _{k+1}(s_i)'\boldsymbol{\theta}_{k+1})^2, \end{aligned} \tag{9}$$

where t is the current time step. Then, in each iteration step k, kM-dimensional vector $\boldsymbol{\theta}_k$ is estimated by the least-squares optimization.

Because the number of the parameters increases by M after each iteration, the computation (in the order of $\mathcal{O}((kM)^3)$ due to the inverse calculation of the design matrix) can be intractable. Furthermore, the value function approximation can be unstable due to the singularity of the design matrix as the number of highly correlated basis functions increases.

4 Sequential approximation method for automatic basis function construction

To overcome these diffculties, we propose a novel approach for automatic basis function construction, where the computational cost is only of $\mathcal{O}(M^3)$, which is independent of k, and stable learning can be achieved by avoiding the singularity. While in the previous approach shown above, the value function is approximated by all parameters, we consider the approximation of the TD error instead of the value function with restricted parameters $\boldsymbol{\theta}_{\text{new}}$, and learned parameters $\boldsymbol{\theta}_k$ being fixed after each optimization and embedded in $\boldsymbol{V}^k$. To this end, we adopt least-squares policy evaluation learning (LSPE) [5] instead of LSTD. In this method, the TD error instead of the value function is approximated by least-squares optimization, so as to have a form:

$$\boldsymbol{V}_{k+1} = \boldsymbol{V}_k + \quad \boldsymbol{\theta}_{\text{new}}. \tag{10}$$

Note that only M-dimensional vector $\boldsymbol{\theta}_{\text{new}}$ is learned, and the others are fixed. The cost function is given by

$$\begin{aligned} J(\boldsymbol{\theta}_{\text{new}}) = \ & \tfrac{1}{2t}\textstyle\sum_{i=0}^{t-1}(r_{i+1} + \ V^k(s_{i+1}) - V^k(s_i) \\ & -\boldsymbol{\psi}(s_i)'\boldsymbol{\theta}_{\text{new}})^2, \end{aligned} \tag{11}$$

where $\boldsymbol{\psi}(s_i) \in \{0,1\}^M$ is a newly produced basis function and is the transpose of the row vector in , evaluated at the state s_i. TD error: $r_{i+1} + \ V^k(s_{i+1}) - V^k(s_i)$ is the target of $\boldsymbol{\psi}(s_i)'\boldsymbol{\theta}_{\text{new}}$ in this regression problem. In the minimization of this objective function with respect to $\boldsymbol{\theta}_{\text{new}}$, the least-squares solution is given as

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{B}^{-1}\boldsymbol{c}, \tag{12}$$

where

$$\begin{aligned} \boldsymbol{B} &= \frac{1}{t}\sum_{i=0}^{t-1}\boldsymbol{\psi}(s_i)\boldsymbol{\psi}(s_i)', \\ \boldsymbol{c} &= \frac{1}{t}\sum_{i=0}^{t-1}\boldsymbol{\psi}(s_i)\left(r_{i+1} + \ V^k(s_i+1) - V^k(s_i)\right)'. \end{aligned}$$

Then, the TD error can be successively approximated by sequentially constructed basis functions. Note that the newly produced basis functions are only used for each sequential approximation. Then, the computational cost is of $\mathcal{O}(M^3)$, which is much smaller than $\mathcal{O}((kM)^3)$ of the Keller et al.'s method. Moreover, it can be seen that our method is the straightfoward application of the Bertsekas and Castanon's aggregation theory shown in Eq.(5)-(7), since the TD error is orthogonally projected onto the space . The algorithm is depicted in Fig 1.

Figure 1: *single iteration step*

input
$h_t = \langle s_0, r_1, \ldots, s_t, r_t\rangle$: a sample trajectory until t
M : the number of basis functions
$\boldsymbol{V}_k$: estimated value function

For *i = 1:t-1*
$e_i \leftarrow r_{i+1} + \ V_k(s_{i+1}) - V_k(s_i)$
End
$\Psi \leftarrow$ *GenerateBasis*$(\{e\}, M)$
$\boldsymbol{\theta}_{\text{new}} \leftarrow$ *LSPE* $(h_t, \boldsymbol{V}_k, \Psi)$
$\boldsymbol{V}_{k+1} \leftarrow \boldsymbol{V}_k + \Psi\boldsymbol{\theta}_{\text{new}}$
return $\boldsymbol{V}_{k+1}$

The process of the adaptive basis function construction is described by $GenerateBasis$. This function generates basis functions according to the TD errors e. The parameter $\boldsymbol{\theta}_{\text{new}}$ is estimated by LSPE (Eq.(12)), and then the value function is updated. Since the previous parameters are fixed, serious instability due to the singularity can be avoided in our method.

5 Experimental results

To verify the performance of our method, we evaluate the maximum norm[1] between the true value function and the estimated value functions by several methods in simulation experiments. Since the maximum norm is associated with the worst case policy improvement, this measure is important for the convergence property of RL [1]. We compared our method with the following three methods:

(a). LSPE with fixed basis functions, which is a usual setting of conventional RL methods,

(b). LEPE with randomly generated basis functions,

(c). LSTD with adaptive construction of basis functions (Keller et al.'s method).

By comparing with (a), we can show the effectiveness of sequential construction of basis functions. By comparing with (b), we can show the effectiveness of the adaptive constructions based on TD error. By comparing with (c), we can show the effectiveness of our method compared with the Keller et al.'s method. We assume that the number of states is 100, the discount factor is 0.97, the number of basis functions (M) is 2, and the transitional matrix under the fixed policy π is as $\boldsymbol{P} = \boldsymbol{Z}\left(0.03 \times \boldsymbol{P}_{\text{rand}} + (1 - 0.03) \times \boldsymbol{I}\right)$, where the components of $\boldsymbol{P}_{\text{rand}}$ are drawn from the uniform distribution in $(0,1]^{|\mathcal{S}|\times|\mathcal{S}|}$, $\boldsymbol{I}$ is a unit matrix, and $\boldsymbol{Z}$ is a normalization term. Then the true value function is given by $\boldsymbol{V}_{\text{true}} = (\boldsymbol{I} - \boldsymbol{P})^{-1}\boldsymbol{r}$.Samples are drawn from the transition matrix $\boldsymbol{P}$ step by step starting from an initial state s_0. We generated new basis functions and the estimated value function every 10 sample generations. Fig 2 shows the performance comparison where the horizontal axis is the number of generated samples. The performances were averaged over 100 independent runs.

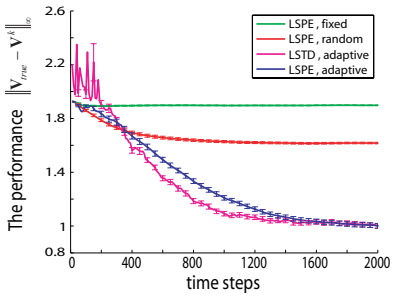

Figure 2: The performance comparisons

With setting (a), the performance saturated immediately due to the restricted expression of fixed basis functions. With setting (b), the performance gradually increased, actually much better than (a) every time step. The Keller et al.'s method (setting (c)) drastically improved the performance, but it showed substantial instability behavior due to simultaneous optimization of all the parameters. In the following learning phase, the design matrix was always singlar, so we had to use the singular value decomposition. Moreover, the computational cost of $\mathcal{O}((kM)^3)$ was a hazard for examination of longer time-steps behaviors. In our method, the performance was improved similar to (c), but did not show any instability in contrast to (c). Furthermore, since the computational cost is of only $\mathcal{O}(M^3)$, the learning can be continued for any period in on-line manner; this is an important feature for scaling-up the application.

[1]For an N-dimensional vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]$, the maximum norm is difined as $\boldsymbol{x}$ $\max$ x_1, x_2, $\ldots$, x_N.

6 Conclusion and future works

We proposed a novel scheme for automatic construction of basis functions, without huge computational cost. In this method, the value function was estimated by sequential approximation of the TD errors with adaptive basis functions. In our experiments, we showed that the performance of our method was much better than previous methods in spite of small computational cost.

Although our method was formalized by using aggregated discrete states, many realistic problems have continuous state space. We need some modifications of our method to be applicable to continuous-state space problems. One possibility is to extend the value function approximator to that with continuous-state basis functions such as the radial basis functions, but this extension hinder direct application of the aggregation theory, and the performance assureness can be abused. So theoretical foundations are desired in continuous-state settings. Furthermore, the combination of the automatic dimensionality reduction method, such as neighborhood component annalysis, explored by Keller et al. [2], could be interesting, but this orientation does not have sufficient theoretical backgrounds yet.

References

[1] Bertsekas, D. and Tsitseklis, J. (1996), Neuro-Dynamic Programming, ISBN 1-8865 29-10-8

[2] Keller, P., Mannor, S., and Precup, D. (2006), Automatic Basis Function Construction for Approximate Dynamic Programming and Reinforcement Learning. *International Conference on Machine Learning*

[3] Bertsekas, D. and Castanon, D. (1989), Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions of Automatic Control*, 34,589-598

[4] Bradtke, S. and Barto, A. (1996), Linear Least-Squares Algorithms for Temporal Difference Learning. *Machine Learning*, 22, 33-57

[5] Bertsekas, D. and Yu, H. (2006), Convergence results for some temporal diffrence methods based on least squares. (Technical Report LIDS-2697).