Chinese Journal of Electronics
Vol.23, No.1, Jan. 2014

Temporal Difference Learning with Piecewise Linear Basis[*]

CHEN Xingguo, GAO Yang and FAN Shunguo

(*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China*)

Abstract — Temporal difference (TD) learning family tries to learn a least-squares solution of an approximate Linear value function (LVF) to deal with large scale and/or continuous reinforcement learning problems. However, due to the represented ability of the features in LVF, the predictive error of the learned LVF is bounded by the residual between the optimal value function and the projected optimal value function. In this paper, Temporal difference learning with Piecewise linear basis (PLB-TD) is proposed to further decrease the error bounds. In PLB-TD, there are two steps: (1) build the piecewise linear basis for problems with different dimensions; (2) learn the parameters via some famous members from the TD learning family (linear TD, GTD, GTD2 or TDC), which complexity is $O(n)$. The error bounds are proved to decrease to zero when the size of the piecewise basis goes into infinite. The empirical results demonstrate the effectiveness of the proposed algorithm.

Key words — Linear function approximation, Reinforcement learning, Piecewise linear basis, TD learning family.

I. Introduction

Reinforcement learning aims to maximize a long term cumulative reward by learning an optimal policy during interacting with an environment. Function approximation techniques such as linear function approximation[1], regression tree methods[2], neural networks[3], Bayesian methods[4], and kernel methods[5] are usually combined with reinforcement learning algorithms to deal with the curse of dimensionality in the large scale and/or continuous problems. Among them, linear function approximation is particularly attractive due to its simplicity to implement and analyze.

The TD learning family mainly includes linear TD and TD(λ)[6], least-squares TD(λ) algorithm (LSTD(λ))[7], Incremental least-squares TD learning (iLSTD)[8], Gradient temporal-difference (GTD)[9], 2nd generation of GTD (GTD2) and TD with gradient correction (TDC)[10], *etc.* All of them are trying to get a least squares solution $\boldsymbol{\theta}^*$ that is bounded as:

$$||\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\theta}^* - \boldsymbol{V}^*||_{\boldsymbol{D}} \leq \frac{1-\lambda\gamma}{1-\gamma}||\boldsymbol{\Pi}\boldsymbol{V}^* - \boldsymbol{V}^*||_{\boldsymbol{D}} \quad (1)$$

where $\boldsymbol{\Phi}$ is a feature matrix of all the states, $\boldsymbol{\theta}$ is the parameter vector to be learned in a linear system, $\boldsymbol{V}^*$ is the optimal value function unknown to learning algorithms, $\boldsymbol{D}$ is a diagonal matrix, $\boldsymbol{\Pi} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{D}$ is a projection operator, λ is the eligibility trace and γ is a discount factor[11]. Several methods have been developed to decrease the bound as in the righthand of Eq.(1). The most common way focuses on the study of the projection $\boldsymbol{\Pi}$: the projection operator itself and feature selection. The studies are summarized as follows: (1) Projection operator: Different solutions such as TD, LSTD, are theoretically analyzed and are proved that each solution is optimal with respect to a specific objective function (a specific projection operator) in 2002[12]; As an extension, a unified oblique projection view is proposed to compare the method of computing the temporal difference fix point with the method of minimizing the Bellman residual in 2011[13]; LSTD with random projections (LSTD-RP) is proposed to deal with the reinforcement learning problems with high dimensional space when the number of features is bigger than the number of samples in 2010[14]; New temporal difference methods for general projected equations are proposed to deal with near singularity in the associated matrix inversion[15]. (2) Feature selection: A Bellman-error-based approach is proposed to iteratively generate basis functions for linear value function approximation[16]. Regularization methods try to select appropriate features, which in some sense modifies the projection operator $\boldsymbol{\Pi}$ in order to decrease the bound[17−21].

In 2007, Piecewise linear function approximation (PLFA) divides the state (or state action) space into several partitions. The goal is to learn an estimate θ_i for each partition[22]. However, there is no theoretical analysis about error bounds.

The error bound Eq.(1) can be very large if the feature function ϕ is not well defined. In this paper, we aim to decrease the error bound further. Piecewise linear basis combined with the least squares method is proposed in 2009 to deal with the approximation problem of lognormal sum distribution[23]. Inspired by this, we bring piecewise linear basis to reinforce-

*Manuscript Received Dec. 2012; Accepted Jan. 2013. This work is supported in part by the National Science Foundation of China (No.61035003, No.61175042, No.60721002), the National 973 Program of China (No.2009CB320702), the 973 Program of Jiangsu, China (No.BK2011005), and Program for New Century Excellent Talents in University (No.NCET-10-0476).

ment learning with function approximation, extend piecewise linear basis from 1-dimension to multi-dimension, and propose a projection method to deal with RL problems with non-deterministic dimensionality. Suppose the true value function is smooth[+], the optimal value function with piecewise linear basis can exactly approximate the true value function as the number of the piecewise linear basis goes to infinity.

The rest of the paper is organized as follows. Related work and our motivation is given in Section II. In Section III, 1-dimension piecewise linear basis is introduced, and its error bound is theoretically analyzed. In Section IV, the extension from 1-dimension to deterministic multi-dimension and non-deterministic dimension piecewise linear basis is studied. Then, the PLB-TD algorithm is proposed. Section V presents the experimental settings and results, and some analysis is given. Finally, the conclusion is given in Section VI.

II. Related Work

We start with a short introduction to RL, tabular TD(λ), linear TD learning family and our motivation.

1. Markov decision process

Reinforcement learning (RL) in sequential decision making can typically be modeled as a Markov decision process (MDP)[1]. An MDP is a tuple $\langle S, A, R, P\rangle$, where S is the state space of the environment; A is the action space of the agent; $R : S\times A\times S \rightarrow R$ is a reward function; $P : S\times A\times S \rightarrow [0,1]$ is a state transition function.

A policy is a mapping $\pi : S\times A \rightarrow [0,1]$. The goal of the agent is to find an optimal policy π^* to maximize the expectation of a discounted accumulative reward in a long period R: $\pi^* = \arg\max_\pi R_\pi = \arg\max_\pi E_\pi[\sum_{t=0}^{\infty}\gamma^t r_t]$, where γ is a discount factor; r_t is the reward at time-step t, and $E_\pi[\cdot]$ is the expectation with respect to the policy π.

In this paper, we focus on the state value function for a stationary policy π, which is defined as: $V^\pi(s) = E_\pi[\sum_{t=0}^{\infty}\gamma^t r_t|s_0 = s]$. The optimal value function is defined as: $V^*(s) = E_{\pi^*}[\sum_{t=0}^{\infty}\gamma^t r_t|s_0 = s]$. According to the Bellman optimality equation,

$$\begin{aligned}V^*(s) &= \max_a E[r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a]\\ &= \max_a \sum_{s'} P(s,a,s')[R(s,a,s') + \gamma V^*(s')]\end{aligned}$$

2. The linear TD learning family

The tabular TD learning algorithms cannot work for large scale or continuous RL problems due to the curse of dimensionality[1]. Thus, function approximators are commonly used to solve such problems. In this paper, we focus on the linear function approximation. The linear TD learning family has many members that have been summaried in the introduction. In the following, linear TD learning and gradient temporal difference learning are introduced because they are the most famous algorithms, and their complexity is $O(n)$ that enables online learning.

(1) Linear TD(λ) learning

Linear TD(λ) learning is combined with a linear function approximator, where the value function is represented as: $\tilde{V}(s) = \boldsymbol{\theta}\boldsymbol{\phi}^{\mathrm{T}}(s) = \sum_{i=1}^{M}\theta_i\phi_i(s)$, where $\boldsymbol{\phi}(s) = [\phi_1(s), \phi_2(s), \ldots, \phi_M(s)]$ is a vector of feature functions; M is the number of the feature function and $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_M]$ is a weight vector. The update rules for the linear TD(λ) algorithm at each iteration t become:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha_t\delta_t\boldsymbol{e}_t \tag{2}$$

where the temporal difference error is evaluated as:

$$\delta_t \leftarrow r_t + \gamma\boldsymbol{\theta}\boldsymbol{\Phi}^{\mathrm{T}}(s_{t+1}) - \boldsymbol{\theta}\boldsymbol{\Phi}^{\mathrm{T}}(s_t) \tag{3}$$

and the eligibility traces are updated according to:

$$\boldsymbol{e}_{t+1} \leftarrow \gamma\lambda\boldsymbol{e}_t + \boldsymbol{\Phi}(s_{t+1}) \tag{4}$$

The convergence of the tabular and linear TD learning algorithms has been well studied in Ref.[23].

(2) Gradient temporal difference learning

Gradient TD (GTD)[9], 2nd generation of GTD (GTD2)[10] and linear TD with gradient correction (TDC)[10] aim to minimize the Mean square projected Bellman Error (MSPBE).

$$MSPBE(\boldsymbol{\theta}) = ||\tilde{\boldsymbol{V}}_{\boldsymbol{\theta}} - \boldsymbol{\Pi}T\tilde{\boldsymbol{V}}_{\boldsymbol{\theta}}||_{\boldsymbol{D}^2} \tag{5}$$

where T is the Bellman operator ($\tilde{\boldsymbol{V}} = R + \gamma P\tilde{\boldsymbol{V}} = T\tilde{\boldsymbol{V}}$) and $\boldsymbol{\Pi}$ is the projection operator ($\boldsymbol{\Pi} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{D}$), where $\boldsymbol{\Phi} = [\boldsymbol{\phi}(s_1), \boldsymbol{\phi}(s_2), \cdots, \boldsymbol{\phi}(s_N)]^{\mathrm{T}}$. The update rules for GTD are as follows:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})(\boldsymbol{\phi}_t^{\mathrm{T}}\boldsymbol{w}_t) \tag{6}$$

with

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha_t'(\delta_t\boldsymbol{\phi}_t - \boldsymbol{w}_t) \tag{7}$$

GTD2 updates the same $\boldsymbol{\theta}$ as GTD, but with a different $\boldsymbol{w}$ update:

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \alpha_t'(\delta_t - \boldsymbol{\phi}_t^{\mathrm{T}}\boldsymbol{w}_t)\boldsymbol{\phi}_t \tag{8}$$

In TDC, $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha_t\delta_t\boldsymbol{\phi}_t - \alpha_t\gamma\boldsymbol{\phi}_{t+1}(\boldsymbol{\phi}_t^{\mathrm{T}}\boldsymbol{w}_t)$ with $\boldsymbol{w}$ updated as in GTD2.

3. Motivation

The above algorithms try to find a best parameter $\boldsymbol{\theta}$ in the sense of least squares. The error of the optimal $\boldsymbol{\theta}$ is bounded as in Eq.(1). Since the approximated solution ($\boldsymbol{\theta}$) is on the predefined feature space, if the feature function $\boldsymbol{\phi}$ is not well defined, then, this solution may differ a lot from the (unknown) optimal solution. In this paper, our motivation is to decrease the error bound further *via* piecewise linear basis functions.

III. Piecewise Linear Value Function

Based on the above motivation, 1-dimension piecewise linear basis is introduced and the error bound is analyzed as follows.

1. 1-dimension piecewise linear basis

Let the interval $[a,b]$ be divided into n equal sub-intervals $I_i = [x_{i-1}, x_i]$ of length $l = (b-a)/n$, $i = 1, 2, \cdots, n$. Obviously, $x_i = a + i\times l$ for $i = 0, 1, \cdots, n$. Let a Piecewise linear basis (PLB) function $\boldsymbol{\phi}$ denotes $n+1$ dimensional vector space of the interval $[a,b]$. $\boldsymbol{\phi} = [\phi_0, \phi_1, \cdots, \phi_n]$, where

$$\phi_i(x) = f\left(\frac{x - x_i}{l}\right), \quad i = 0, 1, \cdots, n \tag{9}$$

[+]This assumption is usually true in practise.

and

$$f(x) = \begin{cases} 1+x, & \text{if } x \in [-1,0] \\ 1-x, & \text{if } x \in [0,1] \\ 0, & \text{otherwise} \end{cases}$$

2. Piecewise linear value function

Piecewise linear value function (PLVF) is a linear value function with piecewise linear basis, *e.g.*, for state s:

$$V(s) = \boldsymbol{\theta}\boldsymbol{\phi}^{\mathrm{T}}(s) = \sum\nolimits_{i=0}^{n} \theta_i \phi_i(s) \tag{10}$$

where the PLB $\boldsymbol{\phi}(s) = [\phi_0(s), \phi_1(s), \phi_2(s), \cdots, \phi_n(s)]$ is defined as Eq.(9), $(n+1)$ is the size of the piecewise linear basis and $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \cdots, \theta_n]$ is the parameter vector. It is easy to get that $V(x_i) = \theta_i$ for all i.

Theorem 1 Suppose the true value function $\boldsymbol{V}$ is smooth, then the piecewise linear value function $\tilde{\boldsymbol{V}}$ learned from the TD learning family will converge to the true value function $\boldsymbol{V}$ as the number of piecewise linear basis goes to infinity.

Proof It is enough to show that $\lim_{n\to\infty} ||\boldsymbol{V} - \tilde{\boldsymbol{V}}^*||_2 = \lim_{n\to\infty} \min_{\tilde{V}\in\phi} ||\boldsymbol{V} - \tilde{\boldsymbol{V}}||_2 = 0$.

Since $\boldsymbol{V}$ is a smooth function of x over $[a,b]$, it is uniformly continuous. Thus, for any given precision parameter $\epsilon > 0$, there exist $\delta > 0$ such that $|V(p) - V(q)| < \epsilon/\sqrt{b-a}$ whenever $|p-q| < \delta$ for all $p, q \in [a,b]$. Divide $[a,b]$ into n equal subintervals with $n > (b-a)/\delta$.

Since the parameter vector $\boldsymbol{\theta}$ of the piecewise linear value function is learned from the TD learning family, then the optimal vector $\boldsymbol{\theta}$ is $\theta_i = V(x_i)$, for any $i \le n$. That is $\tilde{V}(x) = \sum\nolimits_{i=0}^{n} V(x_i)\phi_i(x)$. Then, since $\sum\nolimits_{i=0}^{n} \phi_i(x) \equiv 1$,

$$\begin{aligned} |V(x) - \tilde{V}(x)| &= \left| \sum_{i=0}^{n} V(x)\phi_i(x) - \sum_{i=0}^{n} V(x_i)\phi_i(x) \right| \\ &\le \sum\nolimits_{i=0}^{n} |V(x) - V(x_i)|\phi_i(x) \\ &< \frac{\epsilon}{\sqrt{b-a}} \sum\nolimits_{i=0}^{n} \phi_i(x) \\ &= \frac{\epsilon}{\sqrt{b-a}} \end{aligned}$$

uniformly on $[a,b]$. It follows that

$$\begin{aligned} ||\boldsymbol{V} - \tilde{\boldsymbol{V}}||_2 &= \left[\int_a^b (V(x) - \tilde{V}(x))^2 dx \right]^{\frac{1}{2}} \\ &\le \max_{x\in[a,b]} |V(x) - \tilde{V}(x)| \sqrt{b-a} \\ &< \frac{\epsilon}{\sqrt{b-a}} \sqrt{b-a} \\ &= \epsilon \end{aligned}$$

which implies that $\min_{\tilde{V}\in\phi} ||\boldsymbol{V} - \tilde{\boldsymbol{V}}||_2 \le ||\boldsymbol{V} - \tilde{\boldsymbol{V}}||_2 < \epsilon$. Since ϵ is arbitrary, the theorem is proved.

IV. Temporal Difference Learning Family with Piecewise Linear Basis

In the above theorem, we suppose that the true value function is smooth over the interval $[a,b]$. In practise, states are not naturally arranged in 1-dimension, but represented in multi-dimension, even in non-deterministic multi-dimension.

There are two main categories of methods: (1) to extend PLB from 1-dimension to multi-dimension for deterministic dimensionality; (2) to project the states from multi-dimension into 1-dimension for non-deterministic dimensionality.

1. Extension from 1-dimension to multi-dimension

Let state s be represented in a K-dimension vector $(x_1, x_2, \cdots, x_K)$, where K is the dimensionality of the multi-dimension. For each i-dimension, $x_i \in [a_i, b_i]$ is divided into n_i equal sub-intervals $I_{ij} = [x_{i,j-1}, x_{i,j}]$ of length $l_i = (b_i - a_i)/n_i, j = 1, 2, \cdots, n_i$. Obviously, $x_{i,j} = a_i + j \times l_i$, for $j = 0, 1, \cdots, n_i$. Then, the number of the piecewise linear basis for the state space in K-dimension is

$$N = \prod\nolimits_{i=1}^{K} (n_i + 1) \tag{11}$$

and the state space is divided into $\prod\nolimits_{i=1}^{K} n_i$ K-dimension subspaces. Let PLB $\boldsymbol{\phi} = [\phi_0, \phi_1, \cdots, \phi_{N-1}]$ denotes N dimensional vector space of the states, where each basis ϕ_m $(0 \le m \le N-1)$ is corresponding to a vertex of the subspace $p = (x_{1,p_1}, x_{2,p_2}, \cdots, x_{K,p_K})$ and $m = \sum\nolimits_{i=1}^{K} p_i \prod\nolimits_{j=i}^{K} n_j$.

For a given state $s = (x_1, x_2, \cdots, x_K)$, find intervals I_{ij} for each dimension i, where $x_i \in I_{ij}$, that is $x_{i,j-1} \le x_i \le x_{i,j}$. There are 2^K vertex for all these intervals. Suppose each vertex denotes as $V_q = (v_{q,1}, v_{q,2}, \cdots, v_{q,K})$ where $1 \le q \le 2^K$. Compute each distance D_q between state s and vertex V_q, $D_q = \sqrt{\sum\nolimits_{i=1}^{K} (x_i - v_{q,i})^2}$. If there exits a distance $D_p = 0$, then PLB corresponding to vertex V_p is 1 and others are 0. Otherwise, PLB $\boldsymbol{\phi}$ corresponding to the vertex V_q is

$$\phi_q = \frac{1/D_q}{\sum\nolimits_{i=1}^{2^K} \frac{1}{D_i}} \tag{12}$$

The other $(N - 2^K)$ values in PLB $\boldsymbol{\phi}$ are 0.

From Eq.(12), we can see that the size of PLB increases exponentially with the linear increase of the states dimension. It is easy to implement for low-dimension, where the dimensionality of multi-dimension K is small. Thus, this extension for high-dimension is not recommended for practise. One possible way for high-dimension is feature selection or feature extraction before the extension. And another possible way is to do the state projection from multi-dimension to 1-dimension.

Note that the sum of all features defined as Eq.(12) for each state is 1. Then, the extension of multi-dimension PLB holds the properties of 1-dimension PLB. That is, the error bound of the TD learning family with multi-dimension PLB is the same as in Theorem 1.

2. States projection into 1-dimension for non-deterministic dimensionality

When the dimensionality is not deterministic, the extension from 1-dimension to multi-dimension becomes impossible. Then, a possible way is to do projection. Firstly, an optimal way to construct piecewise linear basis is shown as follows.

(1) An optimal projection

Suppose $V^\pi(s)$ is the optimal value function for policy π for some MDP. Then, a projection $\boldsymbol{P}$ from states $\boldsymbol{S}$ to the interval $[V^\pi_{\min}(s), V^\pi_{\max}(s)]$, where $V^\pi_{\min}(s) = \min_s V^\pi(s)$ and

$V^\pi_{\max}(s) = \max_s V^\pi(s)$, is defined as $P(s) = V^\pi(s)$. It is easy to see that this projection is optimal*.

Certainly, it is not necessary to learn a PLVF if we can get an optimal value function. Moreover, there is usually not such a given optimal value function in practise.

(2) An approximate projection

In this paper, we find by empirical experiments that a projection using a sub-optimal value function can usually accelerate the convergence of learning. The reason can be explained as follows: the projection using a sub-optimal value function may have captured the structure of the optimal value function. The result is shown in the experiments.

3. The LPB-TD learning

The LPB-TD learning refers to the Temporal difference learning family with Linear piecewise basis. In LPB-TD, there are two steps: ① Build the linear piecewise basis ϕ; ② Learn the parameters *via* the linear TD learning family (linear TD, GTD, GTD2, or TDC), see Fig.1.

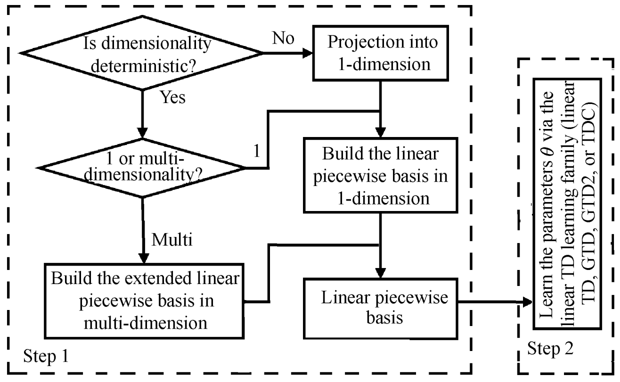
Is dimensionality deterministic?
No
Projection into 1-dimension
Yes
1 or multi-dimensionality?
1
Build the linear piecewise basis in 1-dimension
Multi
Build the extended linear piecewise basis in multi-dimension
Linear piecewise basis
Learn the parameters θ via the linear TD learning family (linear TD, GTD, GTD2, or TDC)
Step 1
Step 2

Fig. 1. Two steps of the LPB-TD learning

Suppose a state in reinforcement learning problem is represented as a multi-dimension (noted as K-dimension) vector. In step ①, (1) If K is not deterministic, then the projection method is used to project the states into 1-dimension and the value function with 1-dimension piecewise linear basis is built; (2) If K is deterministic, and $K = 1$, then it is easy to build directly the value function with 1-dimension piecewise linear basis; (3) If K is deterministic, and $K > 1$, then the value function with multi-dimension piecewise linear basis (extension from 1-dimension) is built.

V. Empirical Results

In this paper, three experiments are used to demonstrate the effectiveness of the proposed PLB-TD algorithm: (1) Boyan-chain (1-dimension), a standard episodic task for comparing the TD methods[7,8,10]; (2) Mountain car (2-dimension), a classic RL problem with a continuous state space[25]; (3) Maze (non-deterministic dimensionality), a classic discrete RL problem for testing RL algorithms[1].

1. Boyan-chain

Fig.2 shows the Boyan-chain problem in the general form. In order to demonstrate the effectiveness of piecewise linear basis, results were conducted with two different problem sizes: 14 (original problem) and 102 (extended problem) states.

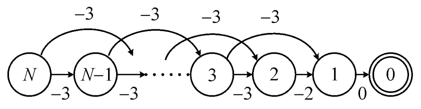

Fig. 2. The general Boyan chain problem

For all states, $s > 2$, the transition probability from s to $(s-1)$ or $(s-2)$ is 0.5, and reward of all transitions are -3, except from state 2 to 1 (when reward is -2) and transitions to 0 (when reward is 0). The optimal state value function is $V(s) = -2s$. The performance of all algorithms are compared in term of RMS error of values over all states: $||\boldsymbol{V} - \tilde{\boldsymbol{V}}||_2$.

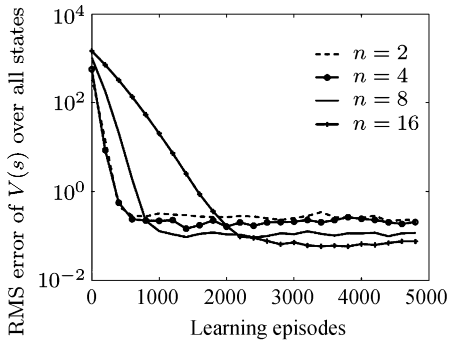

Fig. 3. The original problem with 14 states, TD(λ), where $\alpha = 0.01$, $\lambda = 0.4$, without decreasing α

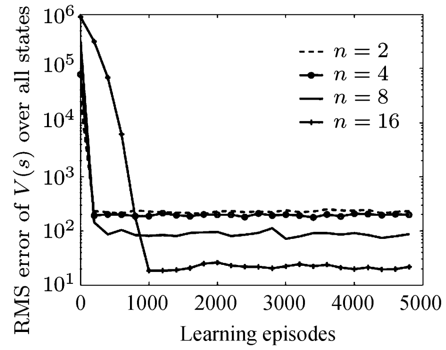

Fig. 4. The extended problem with 102 states, TD(λ), where $\alpha = 0.1$, $\lambda = 0.4$, without decreasing α

The experimental results are averaged by 30 runs with parameter vector θ of each algorithm initialized to zero. The step size α used in these experiments takes in two forms as follows: (1) keeps as a constant, $\alpha = 0.01$ or 0.1; (2) decrease as[8].

$$\alpha_t = \alpha_0 \frac{N_0 + 1}{N_0 + \text{Epsiode}\#} \tag{13}$$

Note that in Boyan-chain problem, the states are arranged in a natural interval $[1, 2, \cdots, 14]$, then there is no need to use a projection and the piecewise linear basis can be directly applied to the states. The selection of the parameters α (in the decreasing case), λ and N_0 was based on experimental finding the best in the set $\alpha_0 \in \{0.01, 0.1, 1.0\}$, $\lambda \in \{0, 0.4\}$ and $N_0 \in \{100, 1000, 10^6\}$. The number of piecewise linear basis are set

*It is easy to extend the state-action value based method. In the paper, we use state value based method for example.

as $n \in \{2,4,8,16\}$ for 14 Boyan-chain states and $n \in \{2,8,32,128\}$ for 102 Boyan-chain states. The result of the best member from the TD learning family of TD(λ), GTD, GTD2 and TDC is choosed to show in figures. The performance of PLB-TD with different parameters are shown in Fig.3 for the original problem with 14 states; and in Fig.4 for the extended problem with 102 states.

From these curves in both 14 and 102 states, we can find that with the increase of the PLB size, the RMS error of $V(s)$ over all states decrease significantly. It is then in an experimental way proved that Theorem 1 is true.

2. Mountain car

In the Mountain car simulation[1], the learning agent needs to drive an underpowered car up a steep mountain road, see Fig.5(a). The reward in this problem is -1 for all time steps until the car moves past its goal position at the top of the mountain, then the episode ends. There are three possible actions: full throttle forward (+1), full throttle reverse (-1), and zero throttle (0). The car moves based on simple physics. Its position x_t, and velocity $\dot{x}_t$ are updated by: $x_{t+1} = bound[x_t + \dot{x}_{t+1}]$, $\dot{x}_{t+1} = bound[\dot{x}_t + 0.001a_t - 0.0025\cos(3x_t)]$, where the bound operation enforces $-1.2 \leq x_{t+1} \leq 0.5$ and $-0.07 \leq \dot{x}_{t+1} \leq 0.07$. The car starts at the position $x_0 = -0.3$, and its goal position is $x_T = 0.5$.

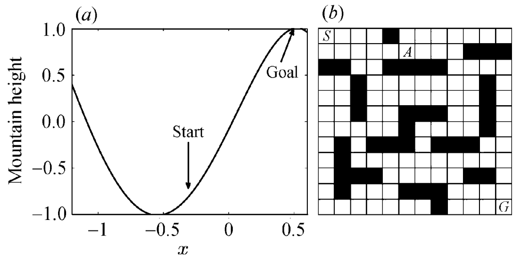

Fig. 5. An illustration for Mountain car and Maze. (a) Mountain car; (b) Maze

The experimental results of the curves are averaged by 10 runs. The parameter settings are summarized as follows: (1) Stepsize α varies from decreasing as Eq.(13) and not decreasing, the initial value varies from $\{0.1, 0.2, 0.5\}$; (2) $\lambda \in \{0.2, 0.4, 0.8\}$, $\beta = 0.1$.

It is difficult to compute the RMS error of $V(s)$ over all Mountain car states. Then, the learning performance is compared based on the ultimate "sum of rewards" of an episode. From Fig.6, we can find that (1) different members from TD learning family with different parameter settings have a different convergence rate; (2) in Fig.6(a), they converge at about or below than -82; in Fig.6(b), they converge at about -82, but below than -80; in Fig.6(c) and (d), they converge at about -80. (3) The trend is that with the increase of the pieces, all algorithms converge closer to the optimal policy.

3. Maze

Consider a maze shown in Fig.5(b), where 'S' is the start position; 'G' is the goal position, and 'A' represents the position of the agent. The objective of the learning agent is to escape from the maze as soon as possible. In each position of Fig.5(b), there are four actions, up, down, left, and right, which takes the agent deterministically to the corresponding neighbor position, except when a movement is blocked by an obstacle or the maze edge. Reward is -1 on all transitions,

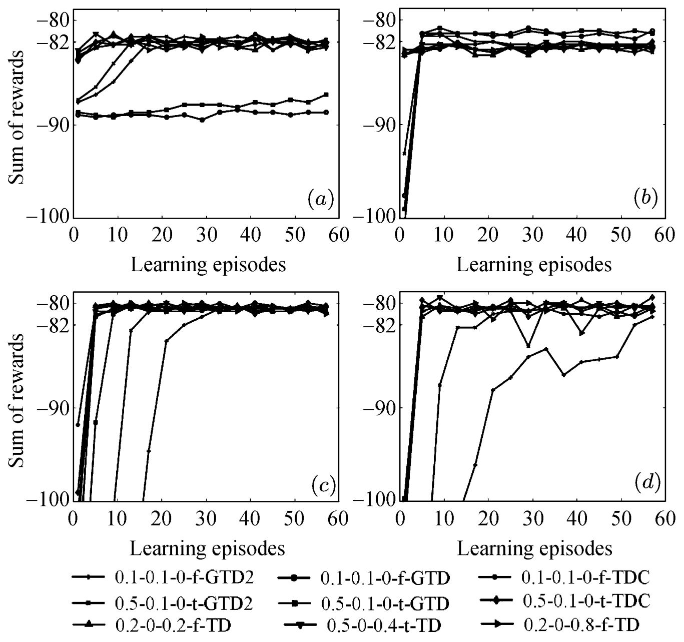

Fig. 6. Learning curves of various algorithms in Mountain car with different pieces. (a) Pieces: (1,4); (b) Pieces: (2,4); (c) Pieces: (3,4); (d) Pieces: (5,4)

except those into the goal state, which reward is 0.

Traditional tabular reinforcement learning algorithms have well performances in the Maze problem. However, reinforcement learning with linear function approximation is rarely applied to Maze. One possible reason may be that the dimensionality of maze is not deterministic. That is, there is not a well defined feature function that can be used to deal with all possible mazes. In our experiments, sub-optimal value function is used to project the states into an interval, and 1-dimension piecewise linear basis is then built. From the learning curves in Fig.7, we can find that with the increase of the pieces number, the ultimate RMS error decreases.

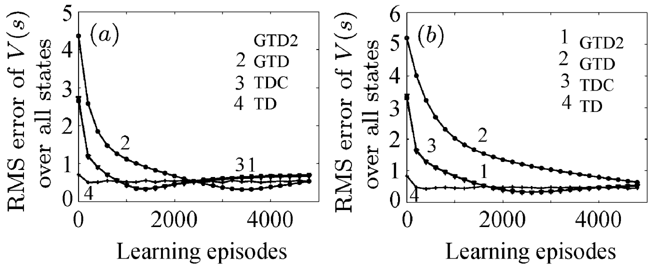

Fig. 7. Learning curves of various algorithms in Maze with different pieces. (a) Pieces: 5; (b) Pieces: 10

VI. Conclusion

In this paper, related works of reinforcement learning with function approximation to deal with large scale and/or continuous reinforcement learning problems are summarized. A new algorithm PLB-TD is proposed in order to decrease the error bounds further *via* piecewise linear basis. Multi-dimension piecewise linear basis is proposed to solve different reinforcement learning problems. The theoretical and empirical results show the effectiveness of the proposed method.

Future work includes (1) Projection method from multi-dimension into 1-dimension should be further studied. (2) In the extension from 1-dimension to multi-dimension, the distance is based on Euclidean distance. Other distance functions can be further studied. For example, metrics between two states in a MDP have been well studied in Ref.[26] for a discrete state space and in Ref.[27] for a continuous state space. Bisimulation metric is a metric generalization of bisimulation using the Kantorovich metric between probability distributions.

References

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA: MIT Press, 1998.

[2] S. Džeroski, L. De Raedt *et al.*, "Relational reinforcement learning", *Machine Learning*, Vol.43, No.1, pp.7–52, 2001.

[3] F. Wang, N. Jin, D. Liu and Q. Wei, "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ϵ-error bound", *IEEE Transactioin on Neural Network*, Vol.22, No.1, pp.1–13, 2011.

[4] Y. Cheng, X. Wang and Y. Zhang, "A bayesian reinforcement learning algorithm based on abstract states for elevator group scheduling systems", *Chinese Journal of Electronics*, Vol.19, No.3, pp.394–398, 2010.

[5] X. Xu, D. Hu and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning", *IEEE Transactioin on Neural Network*, Vol.18, No.7, pp.973–992, 2007.

[6] R. Sutton, "Learning to predict by the methods of temporal differences", *Machine Learning*, Vol.3, No.1, pp.9–44, 1988.

[7] J. Boyan, "Technical update: Least-squares temporal difference learning", *Machine Learning*, Vol.49, No.2, pp.233–246, 2002.

[8] A. Geramifard, M. Bowling and R. Sutton, "Incremental least-squares temporal difference learning", in *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, Boston, Massachusetts, pp.356–361, 2006.

[9] R. Sutton *et al.,* "A convergent $O(n)$ algorithm for off-policy temporal difference learning with linear function approximation", in *Advances in Neural Information Processing Systems 21*, Vancouver, B.C., Canada, pp.1609–1616, 2008.

[10] R. Sutton, H. Maei *et al.*, "Fast gradient descent methods for temporal difference learning with linear function approximation", in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, pp.993–1000, 2009.

[11] J. Tsitsiklis and B. Van Roy, "An analysis of temporal difference learning with function approximation", *IEEE Transactions on Automatic Control*, Vol.42, No.5, pp.674–690, 1997.

[12] R. Schoknecht, "Optimality of reinforcement learning algorithms with linear function approximation", in *Advances in Neural Information Processing Systems 15*, Vancouver, B.C., Canada, pp.1555–1562, 2002.

[13] B. Scherrer, "Should one compute the temporal difference fix point or minimize the bellman residual? the unified oblique projection view", in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, pp.959–966, 2010.

[14] M. Ghavamzadeh, A. Lazaric *et al.*, "LSTD with random projections", in *Advances in Neural Information Processing Systems 23*, Lake Tahoe, Nevada, USA, pp.721–729, 2010.

[15] D. Bertsekas, "Temporal difference methods for general projected equations", *IEEE Transactions on Automatic Control*, Vol.56, No.9, pp.2128–2139, 2011.

[16] R. Parr, C. Painter-Wakefield, L. Li and M. Littman, "Analyzing feature generation for value function approximation", in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, Oregon: ACM, pp.737–744, 2007.

[17] M. Loth *et al.,* "Sparse temporal difference learning using LASSO", in *IEEE Symp. on Adaptive Dynamic Programming and Reinforcement Learning*, Honolulu, pp.352–359, 2007.

[18] J. Kolter and A. Ng, "Regularization and feature selection in least-squares temporal difference learning", in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, pp.521–528, 2009.

[19] M. Hoffman *et al.,* "Regularized least squares temporal difference learning with nested l2 and l1 penalization", in *Proceedings of European Workshop on Reinforcement Learning*, Athens, Greece, pp.102–114, 2011.

[20] M. Ghavamzadeh *et al.,* "Finite sample analysis of Lasso-TD", in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, Washington, USA, pp.1177–1184, 2011.

[21] M. Geist *et al.,* "A dantzig selector approach to temporal difference learning", in *Proceedings of the 29th International Conf. Machine Learning*, Edinburgh, Scotland, pp.1399–1406, 2012.

[22] C. Phua and R. Fitch, "Tracking value function dynamics to improve reinforcement learning with piecewise linear function approximation", in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, Oregon, USA, pp.751–758, 2007.

[23] L. Zhao and J. Ding, "Least squares approximation to lognormal sum distribution via piecewise linear functions", in *Proceedings of the 4th IEEE Conference on Industrial Electronics and Applications*, Xi' an, China, pp.1324–1329, 2009.

[24] D. Bertsekas, J. Tsitsiklis and A. Scientific, *Neuro-Dynamic Programming*, Athens, Greece: Athena Scientific Press, 1996.

[25] A. Dutech *et al.*, "Reinforcement learning benchmarks and bake-offs II", in *Workshop of 17th Advances in Neural Information Processing Systems*, Vancouver, B.C., Canada, 2005.

[26] N. Ferns, P. Castro, D. Precup and P. Panangaden, "Methods for computing state similarity in markov decision processes", in *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, Cambridge, MA, USA, pp.174–181, 2006.

[27] N. Ferns, P. Panangaden and D. Precup, "Bisimulation metrics for continuous markov decision processes", *SIAM Journal of Computing*, Vol.40, No.6, pp.1662–1714, 2011.

CHEN Xingguo was born in 1984. He received the B.S. degree from Nanjing University, Nanjing, China, in 2007, where he is currently working towards the Ph.D. degree in Computer Application Technology. His main research interests include game AI, transfer learning and reinforcement learning. (Email: chenxgspring@gmail.com)

GAO Yang (corresponding author) was born in 1972. He received the Ph.D. degree in computer software and theory from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2000. Currently, he is a professor at the Department of Computer Science and Technology, Nanjing University. His research interests include artificial intelligence and machine learning. He has had more than 50 papers published in top conferences and journals in and out of China. (Email: gaoy@nju.edu.cn)

FAN Shunguo was born in 1989. He received the B.S. degree from Faculty of Science, Jiangsu University, Zhenjiang, China, in 2011. Currently, he is working towards the M.S. degree in computer application technology from the Department of Computer Science and Technology, Nanjing University, Nanjing, China. His main research interests include transfer learning and reinforcement learning. (Email: fanshunguo@163.com)