Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Efficient Average Reward Reinforcement Learning Using Constant Shifting Values

Shangdong Yang[†] and **Yang Gao**[†] and **Bo An**[§] and **Hao Wang**[†] and **Xingguo Chen**[‡]
[†]State Key Laboratory for Novel Software Technology, Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing University, Nanjing 210023, China
[§]School of Computer Engineering, Nanyang Technological University, 639798 Singapore
[‡]School of Computer Science and Technology, School of Software, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
[†]yangshangdong007@gmail.com, [†]{gaoy, wanghao}@nju.edu.cn, [§]boan@ntu.edu.sg, [‡]chenxg@njupt.edu.cn

Abstract

There are two classes of average reward reinforcement learning (RL) algorithms: model-based ones that explicitly maintain MDP models and model-free ones that do not learn such models. Though model-free algorithms are known to be more efficient, they often cannot converge to optimal policies due to the perturbation of parameters. In this paper, a novel model-free algorithm is proposed, which makes use of constant shifting values (CSVs) estimated from prior knowledge. To encourage exploration during the learning process, the algorithm constantly subtracts the CSV from the rewards. A terminating condition is proposed to handle the unboundedness of Q-values caused by such substraction. The convergence of the proposed algorithm is proved under very mild assumptions. Furthermore, linear function approximation is investigated to generalize our method to handle large-scale tasks. Extensive experiments on representative MDPs and the popular game Tetris show that the proposed algorithms significantly outperform the state-of-the-art ones.

Introduction

Reinforcement learning (RL) is an effective learning technique for solving sequential decision-making problems. An RL agent tries to maximize its cumulative reward by interacting with the *environment*, which is usually modeled as a *Markov decision process* (MDP) (Kaelbling, Littman, and Moore 1996). Average reward MDPs are natural models of many non-terminating tasks, such as the call admission control and routing problem (Marbach, Mihatsch, and Tsitsiklis 2000) and the automatic guided vehicle routing problem (Ghavamzadeh and Mahadevan 2007). Average reward RL has received much attention in the recent years (Ortner 2013; Mahadevan 2014; Nguyen et al. 2014).

There are mainly two classes of average reward RL algorithms (Tadepalli 2010). The first class is model-based algorithms such as UCRL (Ortner and Auer 2007), UCRL2 (Jaksch, Ortner, and Auer 2010), and PSRL (Osband, Russo, and van Roy 2013). They achieve efficient exploration by estimating the environment model of an MDP. It often takes a very long time to obtain an accurate model though, especially when the state-action space is large. The second class is model-free algorithms which do not explicitly maintain MDP models, including R-learning (Schwartz 1993; Singh 1994), SMART (Das et al. 1999), RVI Q-learning (Abounadi, Bertsekas, and Borkar 2001; Li 2012), and GR-learning (Gosavi 2004), etc. They all use an adaptive *shifting value* to approximate the optimal average reward to avoid the unboundedness of Q-values. Their learning processes are however typically sensitive to input parameters and may often result in long execution time or even non-convergence. For example, R-learning can be easily perturbed by the learning rates (Mahadevan 1996) and choosing a proper reference state for RVI Q-learning can be difficult when the state space is large (Abounadi, Bertsekas, and Borkar 2001).

To overcome the instability and improve scalability of average reward RL, this paper proposes a novel model-free stochastic approximation algorithm which takes advantage of *constant* shifting values (CSVs) when such values can be inferred from prior knowledge. A main feature of our method (named CSV-LEARNING), is that it encourages a more precise and accurate exploration until a stable policy is reached. As will be discussed later, a CSV usually leads to unboundedness of Q-values during the learning process. Nonetheless, we have derived a terminating condition to secure the *convergence of policy*, and proved that such a convergence could be towards the optimal policy if CSVs are wisely chosen (i.e., with good prior knowledge).

Linear function approximation is also considered in this paper to handle large scale tasks. We compare our method with other existing algorithms using representative MDPs and the popular game Tetris. Experiment results show that CSV-LEARNING converges to the optimal policy significantly faster than existing algorithms.

Preliminaries and Related Work

In this section, we first introduce the average reward MDP, and then briefly present some related work in this area.

An average reward MDP is a tuple $\langle S, A, R, P\rangle$, where S

is the state space, A the action space, $R : S \times A \to \mathcal{R}$ the reward function, and $P : S \times A \times S \to [0, 1]$ the transition function. The average reward under a policy $\pi : S \to A$ is

$$\rho^\pi \stackrel{\text{def}}{=} \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} r(s_k, \pi(s_k)).$$

The goal of a learner is to find a policy π^* to maximize ρ^π.

The *Q-value* of a state-action pair (s, a) under a given policy π is defined as

$$Q^\pi(s, a) \stackrel{\text{def}}{=} \sum_{k=0}^{\infty} E\left[r(s_k, \pi(s_k)) - \rho^\pi \,|s_0 = s, a_0 = a\right].$$

ρ^π can be considered as a shifting value subtracted from the immediate reward r, such that $Q^\pi(s, a)$ is a *bounded* value.

Starting from an initial policy π_0, a learning process coutinuously improves the policy over time, thus generates a sequence of policies $\{\pi_t\}$, $t = 0, 1, 2, \cdots$. Such a sequence of policies $\{\pi_t\}$ is said to be *convergent* if after some time T the policy becomes *stable*, i.e., $\pi_t = \pi_T$ for all $t > T$. We may use Q_t as a shorthand for Q^{π_t} when discussing a learning process.

Average reward MDPs can be solved by model-free algorithms, e.g., R-learning (Schwartz 1993; Singh 1994), SMART (Das et al. 1999), RVI Q-learning (Abounadi, Bertsekas, and Borkar 2001), and recently developed model-based regret minimization (RM) algorithms, e.g., UCRL (Ortner and Auer 2007), UCRL2 (Jaksch, Ortner, and Auer 2010), and PSRL (Osband, Russo, and van Roy 2013).

R-learning (Schwartz 1993; Singh 1994) uses a stochastic estimattion of the shifting value ρ^π; $Q(s, a)$ and ρ are updated alternately:

$$Q(s,a) \leftarrow (1-\alpha)\cdot Q(s,a) + \alpha\cdot\left(r(s,\pi(s)) - \rho + \max_{a'\in A} Q(s',a')\right),$$

$$\rho \leftarrow \rho + \beta\cdot\left(r(s,\pi(s)) - \rho + \max_{a'\in A} Q(s',a') - Q(s,a)\right),$$

where s' is the next state after executing $\pi(s)$ in state s; α and β are learning rates.

RVI Q-learning (Abounadi, Bertsekas, and Borkar 2001) uses the Q-values of a *reference state* as the shifting value:

$$Q(s,a) \leftarrow (1-\alpha)\cdot Q(s,a) + \alpha\cdot\left(r(s,\pi(s)) - f(Q) + \max_{a'\in A} Q(s',a')\right),$$

where $f(Q) : \mathcal{R}^{|S|\times|A|} \to \mathcal{R}$ is a function of the Q-values of some reference state s_0; for example, $f(Q) = Q(s_0, a_0)$ or $f(Q) = \max_a Q(s_0, a)$.

SMART (Das et al. 1999) estimates the shifting value by averaging over the immediate rewards:

$$\rho \leftarrow (1-\beta)\cdot\rho + \beta\cdot r(s,\pi(s)),$$

and the update of $Q(s, a)$ is the same as in **R-learning**.

GR-learning (Gosavi 2004) udpates the shifting value as

$$\rho \leftarrow (1-\beta)\cdot\rho + \beta\cdot\frac{\rho\cdot K + r(s,\pi(s))}{K+1},$$

where K is the number of steps the learner has learned.

RM algorithms keep being optimistic about poorly understood states and actions (i.e., states and actions with high *uncertainty*) in order to encourage exploration. They estimate the MDP using some sampling techniques. Specifically, UCRL (Ortner and Auer 2007) and UCRL2 (Jaksch, Ortner, and Auer 2010) use two *upper confidence* (UC) values about the reward function and transition function; in each episode a policy is generated based on the MDP model and UCs (Jaksch, Ortner, and Auer 2010; Ortner and Auer 2007). PSRL (Osband, Russo, and van Roy 2013) uses posterior sampling: in each episode, it samples an MDP from the posterior distribution and then generates a policy.

In many real world problems, the optimal (or a near-optimal) average reward may be estimated based on our knowledge on the learning task. The estimated average reward can then be used as a (constant) shifting value to speed up learning. Consider the video game *Tetris* as an example. The goal of a Tetris player is to survive as long as possible by clearing tetrominoes. Suppose that 1 line in Tetris contains 10 blocks, and the player gets 1 point for clearing every 1 line. Under a good (i.e., ideally everlasting) policy, every time a tetromino (consisting of 4 blocks) drops, 4 blocks should be cleared on average, hence the average point per step is $\hat{\rho} = \frac{4}{10} = 0.4$, which could be a good shifting value.

Table 1 summaries existing algorithms. Convergence for UCRL2 and PSRL are regret bounds, in which T is the time step, D the diameter of the MDP, and τ the length of a learning episode. In general, model-free methods are *scalable* in terms that they can easily incorporate with function approximation techniques, whereas for model-based methods it is not that straightforward to do so.

Our Methods

In this section, we describe our CSV-LEARNING method, in which the key issue is to derive the terminating condition by using two novel concepts, the *action-value increase* $\Delta Q(s, a)$ (Definition 1) and the *state-value difference* $d(s', s)$ (Definition 2). We also extend the basic CSV-LEARNING with linear function approximation to handle large scale tasks.

Tabular CSV-Learning with prior knowledge $\hat{\rho}$

We first make two mild assumptions on average reward MDPs and the learner.

Assumption 1. *The MDP is irreducible, aperiodic, and ergodic; that is, under any stationary policy, the generated Markov chain is communicating and has a recurrent state.*

Note that this assumption is widely adopted in RL literature (see, e.g., (Abounadi, Bertsekas, and Borkar 2001)).

Assumption 2. *The learner has a prior estimation $\hat{\rho}$ on the optimal average reward ρ^*.*

This assumption is reasonable in many applications such as Tetris (see our analysis of Tetris in the last section)

Table 1: Summary of Average Reward RL Algorithms

	How to estimate shifting values	How to generate policies	Exploration strategy	Convergence	Scalability
R-learning	adaptively	using Q-table	ϵ-greedy, etc.	no guarantee	good
SMART	estimate directly	using Q-table	ϵ-greedy, etc.	no guarantee	good
RVI Q-learning	reference state	using Q-table	ϵ-greedy, etc.	limit convergence	good
GR-learning	estimate directly	using Q-table	ϵ-greedy, etc.	limit convergence	good
UCRL	not required	using estimated R, T	optimism to uncertainty	logarithmic regret	poor
UCRL2	not required	using estimated R, T	optimism to uncertainty	$\tilde{O}(DS\sqrt{AT})$	poor
PSRL	not required	using estimated R, T	optimism to uncertainty	$\tilde{O}(\tau S\sqrt{AT})$	poor

CSV-LEARNING uses the prior knowledge $\hat{\rho}$ as the (constant) shifting value in the update

$$Q_{t+1}(s,a) \leftarrow (1-\alpha)\cdot Q_t(s,a) + \alpha\cdot\left(r(s,a)-\hat{\rho}+\max_{a'} Q_t(s',a')\right). \quad (1)$$

If $\hat{\rho}=\rho^*$, $\{Q_t\}$ is guaranteed to converge to Q^* (Abounadi, Bertsekas, and Borkar 2001). However, the Q-values might be unbounded when $\hat{\rho}<\rho^*$. Having a goal of finding the optimal *policy*, we argue that it is acceptable to have unbounded Q-values as long as the policy converges. To see this, we define the *action-value increase* $\Delta Q_t(s,a)$ and the *state-value difference* $d_t(s',s)$ as follows.

Definition 1 (Action-value increase)**.** *The action-value increase $\Delta Q_t(s,a)$ of state s and action a is the increase of action-value $Q(s,a)$ from time $t-1$ to time t (with the change of policy from π_{t-1} to π_t),*

$$\Delta Q_t(s,a) \stackrel{def}{=} Q_t(s,a) - Q_{t-1}(s,a). \quad (2)$$

Let $\mathcal{M}_t$ be the generated Markov chain under policy π_t, and let Q-values of $\mathcal{M}_t$ be $\mathcal{Q}_t$.

Definition 2 (State-value difference)**.** *The state-value difference of two adjacent states s and s' in $\mathcal{M}_t$ is the difference between their maximum Q-values under a policy π_t.*

$$d_t(s',s) \stackrel{def}{=} \max_{a'} \mathcal{Q}_t(s',a') - \max_{a} \mathcal{Q}_t(s,a). \quad (3)$$

Note that following a policy π, states s and s' are said to be *adjacent* if $\exists a, P(s,a,s')>0$.

We have the following relationship between ρ_t (i.e., ρ^{π_t}), $d_t(\cdot,\cdot)$, and $\hat{\rho}$.

Lemma 1. *In $\mathcal{M}_t$, $d_t(s',s)=\rho_t - r(s,\pi_t(s))$.*

Proof. Let $W=\max_{a'}\mathcal{Q}_t(s',a')$. Then,

$$\begin{aligned} d_t(s',s) &= W - \max_a \sum_{k=0}^{\infty} E\left[r(s_k,\pi_t(s_k)) - \rho_t | s,a\right] \\ &= W - (r(s,\pi_t(s)) - \rho_t + W). \end{aligned}$$

Therefore, $d_t(s',s)=\rho_t - r(s,\pi_t(s))$. □

Lemma 2. *If π_t is stable, then $\rho_t \geq \hat{\rho}$.*

Proof. Based on Eq. 1, we have

$$\begin{aligned} \Delta\mathcal{Q}_t(s,\pi_t(s)) &= \alpha\cdot\left(r\left(s,\pi_t(s)\right) - \hat{\rho} + d_t(s',s)\right) \\ &= \alpha\cdot(\rho_t - \hat{\rho}). \end{aligned} \quad (4)$$

π_t being stable requires that $\Delta\mathcal{Q}_t(s,\pi_t(s))$ being nonnegative for all state s in $\mathcal{M}_t$. This is because, if otherwise $\Delta\mathcal{Q}_t(s,\pi_t(s))$ were negative, then the value of action $a=\pi_t(s)$ in state s would be ever decreasing and as a consequence at some point a could be suboptimal, violating the precondition that π_t is stable. Hence, $\rho_t \geq \hat{\rho}$. □

We now derive the terminating condition for the CSV-LEARNING algorithm.

Theorem 1. *During* CSV-LEARNING*, if for all adjacent states s and s' in $\mathcal{M}_t$ there holds $d_t(s',s)+r(s,\pi_t(s)) \geq \hat{\rho}$, then π_t is stable. In addition, when π_t is stable, if $\hat{\rho}>\rho^{\pi}$ holds for all $\pi\neq\pi^*$, then $\pi_t=\pi^*$.*

Proof. Given the condition $d_t(s',s)+r(s,\pi_t(s)) \geq \hat{\rho}$ and Eq. 4, we have (at time t)

$$\Delta\mathcal{Q}_t(s,a) = \alpha\cdot(\rho_t-\hat{\rho}) \geq 0.$$

Then, for any state s and action a in $\mathcal{M}_t$,

$$\begin{aligned} \pi_{t+1}(s) &= \arg\max_{a'} \mathcal{Q}_{t+1}(s,a') \\ &= \arg\max_{a'} \left(\mathcal{Q}_t(s,a') + \alpha\cdot(\rho_t-\hat{\rho})\right) \\ &= \arg\max_{a'} \mathcal{Q}_t(s,a') \\ &= \pi_t(s), \end{aligned}$$

which means that π_t becomes stable after time t. Due to Lemma 2 and the fact $\hat{\rho} > \max_{\pi\neq\pi^*}\rho^{\pi}$ we know that $\rho_t \geq \max_{\pi\neq\pi^*}\rho^{\pi}$, implying $\pi_t=\pi^*$. □

Algorithm 1 presents CSV-LEARNING.

CSV-Learning with linear function approximation

For large scale problems, tabular form of Q-values might be inefficient. A remedy is to represent Q-values as the product of a group of *features* and their corresponding weights. This technique is known as *linear function approximation* (linear FA). Formally,

$$V_{\boldsymbol{\theta}} = \boldsymbol{\theta}\boldsymbol{\phi}^{\top} = \sum_{i=1}^{M}\theta_i\phi_i(s),$$

Algorithm 1: CSV-LEARNING

Input: an MDP $M = \langle S, A, R, P\rangle$, prior estimation $\hat{\rho}$
Output: a policy π

Initialize $Q(s,a)$ for all s, a arbitrarily, $t \leftarrow 0$
repeat
  $s \leftarrow$ current state
  $a \leftarrow \arg\max_{a'} Q(s,a')$
  Take action a, and observe r, s'
  update Q-value using Eq. 1
  generate $\mathcal{M}_t$ and calculate $d_t(s',s)$ using to Eq. 3
  $t \leftarrow t+1$
until $\forall s', s \in \mathcal{M}_t:\ d_t(s',s) + r(s,\pi_t(s)) \geq \hat{\rho}$;
return policy π derived from $Q(s,a)$.

where $\boldsymbol{\phi}(s) = [\phi_1(s), \phi_2(s), ..., \phi_M(s)]$ is M features of state s and $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_M]$ is the weight vector.

To the best of our knowledge, little attention is paid to average reward learning with linear FA. In this paper, we consider *mean square error* (MSE) as our objective function:

$$\text{MSE}(\boldsymbol{\theta}) = ||V_{\boldsymbol{\theta}} - V^{\pi}||^2_{\boldsymbol{D}},$$

where V^{π} is the value function under policy π and $\|\cdot\|_{\boldsymbol{D}}$ denotes the $\boldsymbol{D}$-norm w.r.t. a diagonal matrix $\boldsymbol{D}$. In our work, we use $TV_{\boldsymbol{\theta}} = R - \hat{\rho} + V_{\boldsymbol{\theta}}$ to estimate V^{π} since the latter is practically unavailable. We use *direct gradient* to find an optimal $\boldsymbol{\theta}$, i.e., to minimize MSE($\boldsymbol{\theta}$):

$$\Delta\theta_d = \alpha \sum_s \left(r(s,\pi(s)) - \hat{\rho} + V_{\boldsymbol{\theta}}(s') - V_{\boldsymbol{\theta}}(s)\right) \cdot \nabla V_{\boldsymbol{\theta}}(s),$$

where $\nabla V_{\boldsymbol{\theta}}(s) = \boldsymbol{\phi}(s)$. We use this policy control strategy together with the eligibility traces $\boldsymbol{e}$. Algorithm 2 describes our extension of CSV-LEARNING using linear FA.

Algorithm 2: CSV-LEARNING with linear FA

Input: an MDP $M = \langle S, A, R, P\rangle$, a group of features $\mathcal{F}$, prior estimation $\hat{\rho}$
Output: a policy π

Initialize $\boldsymbol{\theta}$ arbitrarily, $\boldsymbol{e} \leftarrow \mathbf{0}$, $\lambda \in (0,1)$
repeat
  $s \leftarrow$ current state
  Choose action a in s using a greedy policy
  Take action a, and observe r, s'
  for each $i \in \mathcal{F}$ **do** $e_i \leftarrow e_i + 1$
  $\delta \leftarrow r(s,\pi(s)) - \sum_{i\in\mathcal{F}_a} \theta_i$
  for each $i \in \mathcal{F}$ **do** $Q_a \leftarrow \sum_{i\in\mathcal{F}_a} \theta_i$
  $a \leftarrow \arg\max_{a'} Q_{a'}$
  $\delta \leftarrow \delta - \hat{\rho} + Q_a$; $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\boldsymbol{e}$; $\boldsymbol{e} \leftarrow \lambda\boldsymbol{e}$
  $t \leftarrow t+1$
until *the number of learning steps reaches the set value*;
return policy π derived from $\boldsymbol{\theta}$

Convergence Analysis

In this section, we convert our algorithm to ordinary differential equations (ODEs) and track their solutions. Then we show that the policy out of our algorithm is the same as the one out of the bounded optimal Q-values Q^*. Finally, we prove that the optimal policy learnt by CSV-LEARNING is asymptotically achievable.

First, note that Eq. 1 can be rewritten as

$$Q_t = Q_{t-1} + \alpha(T(Q_{t-1}) - \hat{\rho} - Q_{t-1} + M_t), \quad (5)$$

where TQ_t and M_t are defined by

$$TQ_t(s,a) \stackrel{\text{def}}{=} \sum_{s'} P(s,a,s') \left(r(s,a) + \max_{a'} Q_t(s',a')\right),$$

$$M_t(s,a) \stackrel{\text{def}}{=} r + \max_{a'} Q_t(s',a') - TQ_t(s,a).$$

In (Abounadi, Bertsekas, and Borkar 2001), the martingale difference sequence M_t is proved to be a noise term in Eq. 5. We define T' and $\hat{T}$ based on T:

$$T'(Q) \stackrel{\text{def}}{=} T(Q) - \rho^*,$$
$$\hat{T}(Q) \stackrel{\text{def}}{=} T(Q) - \hat{\rho}.$$

With operators T' and $\hat{T}$, and by eliminating the noise term M_t, Eq. 5 becomes ODEs

$$\dot{Q}(t) = T'(Q(t)) - Q(t), \quad (6)$$
$$\dot{Q}(t) = \hat{T}(Q(t)) - Q(t). \quad (7)$$

Let $y(t)$ and $x(t)$ be the solutions of Eq. 6 and Eq. 7, respectively. Then the convergence proof turns into analyzing the relationship between $y(t)$ and $x(t)$. Due to the monotonicity of T', Eq. 6 converges to an equilibrium (Abounadi, Bertsekas, and Borkar 2001), meaning that $y(t)$ asymptotically converges to the unique optimal Q^*.

In Eq. 5, the Q-value can increase to infinity even though the derived policy is stable and optimal. Thus, we focus on the following *critical Q-table* $\hat{Q}^*$.

Definition 3 (Critical Q-table)**.** *In* CSV-LEARNING*, the critial Q-table $\hat{Q}^*$ is defined as the first Q_t from which the derived policy is stable. In other words, let t be the number satisfying $\pi_{t-1} \neq \pi_t = \pi_{t+1} = \cdots$, then $\hat{Q}^* \stackrel{def}{=} Q_t$.*

To prove the convergence of CSV-LEARNING, it is then sufficient to show that $x(t)$ can reach $\hat{Q}^*$ asymptotically.

Lemma 3. *Let $x(0) = y(0)$, then $x(t) = y(t) + \mathfrak{r}(t)\mathfrak{e}$, where $\mathfrak{e}$ is a matrix of all 1's, and $\mathfrak{r}(t)$ satisfies the ODE $\dot{\mathfrak{r}}(t) = -\mathfrak{r}(t) + \mathfrak{c}(t) + \delta$. Here $\mathfrak{c}(t) : \mathcal{N} \to \mathcal{R}$ is a function of time t and $\delta = \rho^* - \hat{\rho} \geq 0$.*

Proof. According to the monotonicity of $T'(x)$, we have

$$T'(x + \mathfrak{c}(t)\mathfrak{e}) = T'(x) + \mathfrak{c}(t)\mathfrak{e}. \quad (8)$$

Lemma 3.3 in (Abounadi, Bertsekas, and Borkar 2001) says

$$\|x(t) - y(t)\|_u \leq \int_0^t e^{-(t-u)} \|x(u) - y(u)\|_u du.$$

By Gronwall's inequality, we get $\|x(t) - y(t)\|_u = 0$, meaning that, when $\rho^* - \hat{\rho} \leq \min_{\pi\neq\pi^*}(\rho^* - \rho^{\pi})$, the difference between the critical Q-table $\hat{Q}^*$ and Q^* is a multiple of the

constant matrix $\mathfrak{e}$. Thus, we have $x(t) = y(t) + \mathfrak{r}(t)\mathfrak{e}$, and, according to Eq. 8,

$$\begin{aligned}\dot{\mathfrak{r}}(t)\mathfrak{e} &= \dot{x}(t) - \dot{y}(t) \\ &= (T'(x(t)) - x(t) + \delta\mathfrak{e}) - (T'(y(t)) - y(t)) \\ &= (-\mathfrak{r}(t) + \mathfrak{c}(t) + \delta)\mathfrak{e}.\end{aligned}$$

That is, $\dot{\mathfrak{r}}(t) = -\mathfrak{r}(t) + \mathfrak{c}(t) + \delta$. □

Lemma 3 tells us that, with a proper $\hat{\rho}$, CSV-LEARNING can find an optimal policy and the terminating condition in Theorem 1 is satisfiable.

Theorem 2. *The critical Q-table $\hat{Q}^*$ is asymptotically achievable when $\hat{\rho}$ satisfies $\max_{\pi\neq\pi^*}\rho^\pi < \hat{\rho} \leq \rho^*$. As time $t \to \infty$, the difference between $\hat{Q}^*$ and Q^* is $m(t)\cdot\mathfrak{e}$, where $m(t) : \mathcal{N} \to \mathcal{R}$ is a function of time t.*

Proof. Solving the ODE in Lemma 3, we have

$$\mathfrak{r}(t) = \int_0^t e^{-(t-u)}(\mathfrak{c}(t) + \delta)du = (\delta + \mathfrak{c}(t))(1 - e^{-t}),$$

thus $\mathfrak{r}(t) \to \mathfrak{c}(t) + \delta$ when $t \to \infty$. Assume w.o.l.g. that $\hat{Q}^* = Q(t_0)$. Then, $\hat{Q}^*(t_0) = Q^*(t_0) + (\mathfrak{c}(t_0) + \delta)\mathfrak{e}$. Note that $\dot{\mathfrak{c}}(t) = \Delta\mathcal{Q}_t$; hence,

$$\begin{aligned}\mathfrak{c}(t) &= \mathfrak{c}(t_0) + \int_{t_0}^t \Delta\mathcal{Q}_u du \\ &= \mathfrak{c}(t_0) + \alpha\int_{t_0}^t (r(s,\pi(s)) - \hat{\rho} + d_u(s',s))du.\end{aligned}$$

Since u is beyond the critical time t_0, we know that $d_u(s',s) = d^*(s',s)$, thus continuing the calculation,

$$\begin{aligned}\mathfrak{c}(t) &= \mathfrak{c}(t_0) + \alpha\int_{t_0}^t (r(s,\pi(s)) - \hat{\rho} + d^*(s',s))du \\ &= \mathfrak{c}(t_0) + \alpha\delta(t - t_0).\end{aligned}$$

Eventually, we conclude that for any $t > t_0$

$$\hat{Q}^*(t) = Q^*(t) + m(t)\cdot\mathfrak{e},$$

with $m(t) = \mathfrak{c}(t_0) + \delta + \alpha\delta(t - t_0)(1 - e^{-t})$.

$m(t)$ becomes a constant when $t \to \infty$, thus if $\hat{\rho}$ in CSV-learning satisfies $\max_{\pi\neq\pi^*}\rho^\pi < \hat{\rho} \leq \rho^*$ then $\pi_t = \pi^*$ for sufficiently large t. □

Experiments

In this section, the proposed CSV-LEARNING algorithms are evaluated in two average reward MDPs used by Schwartz and Osband (Schwartz 1993; Osband, Russo, and van Roy 2013) (Figure 1(a) & 1(b)), as well as Tetris (Figure 1(c)).

Experiments on two average reward MDPs

The 4-circle MDP (Figure 1(a)) contains 40 states and the action in each state is either staying or moving to the next state. The reward of 5, 20, 45, or 80 is given to the agent if it goes back to State 1 from State 5, 10, 15, or 20, respectively; the rewards are 0 in all the other cases. The optimal policy is to get the highest distant reward 80 with an average reward of 4. The RiverSwim MDP consists of 6 states of which the arrangement is shown in Figure 1(b). The agent begins at the leftmost state and has the choice to swim left or right. Swimming left (along the stream) is always successful while swimming right (against the stream) can fail with probability 0.1 (0.05 stay and 0.05 swim left). The optimal policy is to swim to the right and then stay there.

For comparison, we implemented well known algorithms including R-learning (Schwartz 1993; Singh 1994), SMART (Das et al. 1999), RVI Q-learning (Abounadi, Bertsekas, and Borkar 2001), GR-learning (Gosavi 2004), UCRL (Ortner and Auer 2007), UCRL2 (Jaksch, Ortner, and Auer 2010) and PSRL (Osband, Russo, and van Roy 2013). All the tested algorithms were run 100 times in each MDP and the results were averaged. In MDP 1(a), the learning rates α and β of all model-free algorithms were both 0.1 and exploration was executed by a fixed ϵ-greedy policy with $\epsilon = 0.1$. The reference of RVI Q-learning was State 1. In our method, the CSV was set to 4. In MDP 1(b), the following settings worked the best: learning rates $\alpha = \beta = 0.01$, and the reference of RVI Q-learning was the starting state. The CSV was set to 0.2 in our method. In both MDPs, we implemented UCRL2 with $\delta = 0.05$.

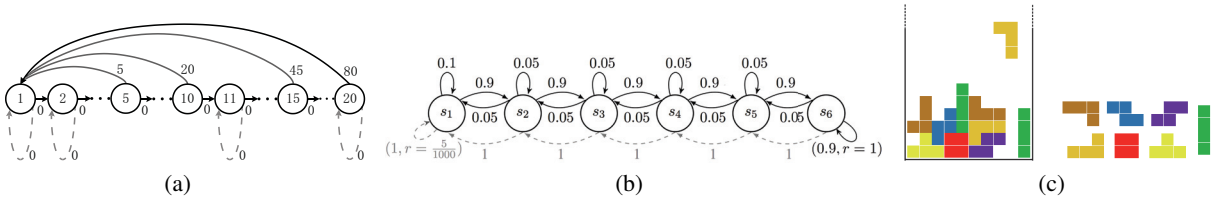
It can be found that in Figures 2(a) & 2(b), our method learned the optimal policy after about 10 thousand steps while other model-free algorithms required at least 60 thousand steps. PSRL and UCRL2 failed to converge to the optimal policy within 100 thousand steps. The reason could be that they need more steps to build sufficiently good MDP models. From Figures 3(a) & 3(b), we see that our methods beated PSRL by a particular of margins in its benchmark SwimRiver. In Figures 2(c) & 3(c) and Figures 2(d) & 3(d), the learning curves of average reward and total regret of different CSVs are presented to show that, with different CSVs, our proposed algorithm can converge. Especially, when the CSV was close to the optimal average reward, the algorithm converged to an optimal policy.

Table 2 shows the runtime and mean total regret results of the tested algorithms. We can see that, due to the process of building the MDP, model-based algorithms were time-consuming. Especially, in the 4-circle MDP with 20 states, our algorithm used one third of the running time of PSRL and got the best performance. Because there were just 6 states in RiverSwim, the runtime of each tested algorithm was almost the same. With regard to the mean total regret, our method outperformed PSRL by more than 10 percent.

Experiments on the video game Tetris

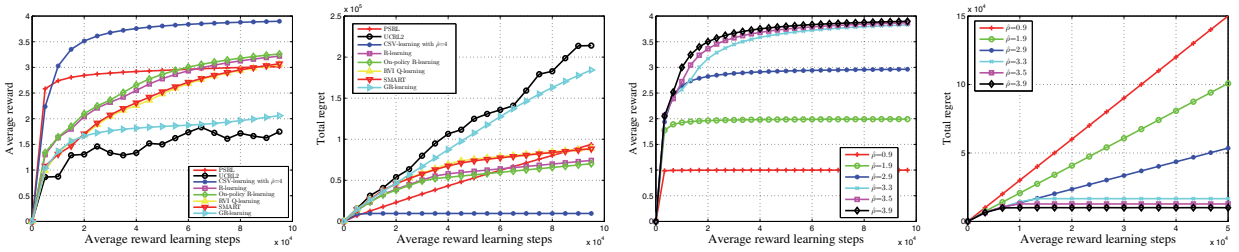
Tetris is a popular video game since 1984. A player pursuing a high score must clear the tetrominoes as many as possible. Although the rules of Tetris are simple, finding an optimal strategy for playing Tetris is NP-hard (Demaine, Hohenberger, and Liben-Nowell 2003). This makes Tetris an appealing benchmark problem for testing RL algorithms.

We used a 10×20 board for the game and selected 14 features (Table 4(a)) to represent the value of a state. The agent would get 1 point if it removed one line.



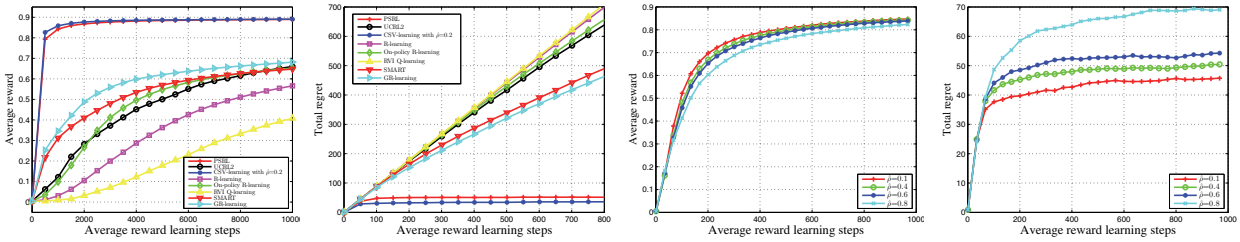
(a) (b) (c)

Figure 1: Two average reward MDPs: (a) A 4-circle MDP with different distant rewards; (b) *RiverSwim* - continuous and dotted arrows represent the MDP under the actions "right" and "left"; (c) The popular video game Tetris and the 7 tetrominoes.



(a) (b) (c) (d)

Figure 2: Results out of the 4-circle MDP: (a) Average reward learning curves; (b) Total regret curves; (c) Average reward learning curves under different CSVs; (d) Total regret curves under different CSVs.



(a) (b) (c) (d)

Figure 3: Results out of the RiverSwim MDP: (a) Average reward learning curves; (b) Total regret curves; (c) Average reward learning curves under different CSVs; (d) Total regret curves under different CSVs.

Table 2: Experiment Results of the Two MDPs.

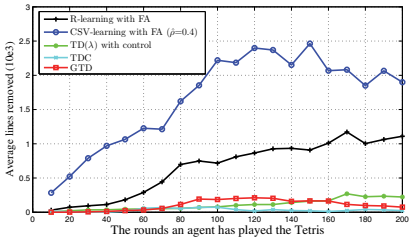
Algorithm	**4-circle MDP**		**RiverSwim**	
	Runtime (s)	**Mean total regret** ($\times 10^4$)	**Runtime** (s)	**Mean total regret** ($\times 10^3$)
R-learning	4.89	7.5721	2.57	7.2891
on-policy R-learning	8.80	7.1694	4.65	7.9683
SMART	4.79	8.9903	2.55	6.8463
RVI Q-learning	4.92	9.0489	2.58	9.1724
GR-learning	4.78	19.0780	2.51	6.2835
UCRL2	13.93	23.0385	**2.48**	3.7207
PSRL	13.31	9.8138	2.53	0.0530
CSV-learning	**4.74**	**0.9730**	2.50	**0.0477**

We tested three well-known FA algorithms: TD(λ) (Barto 1998), GTD (Sutton, Maei, and Szepesvári 2009), and TDC (Sutton et al. 2009). All the algorithms were run 5 times; in each run, every algorithm played 200 rounds of the game and the results were averaged. We used the learning rate $\alpha = 0.1$ for our method, $\alpha = 0.1$ and $\beta = 0.1$ for R-learning, $\alpha = 0.5$ and $\lambda = 0.1$ for TD(λ), $\alpha = 0.1$ and $\beta = 0.6$ for TDC, and $\alpha = 0.1$ and $\beta = 0.9$ for GTD. The parameters above all performed the best via tuning.

From Figure 4(b), we see that after 100 games the policy out of our method cleared near 2,500 lines on average which was twice as many as that of R-learning. The performance of TD(λ), TDC and GTD was poor as the policies learnt by them cleared less than 400 lines.

Feature	Weight	Feature	Weight
maximum well depth	1.614	number of wells	-0.742
holes	-1.675	blocks	20.362
connected holes	-1.776	weighted blocks	-3.549
altitude difference	-1.155	row transition	-22.000
height	-0.131	column transition	-14.033
highest hole	-1.160	potential	-0.946
blocks above the hole	-2.456	smoothness	3.805

(a) Features and corresponding weights learnt by our method



(b) The learning curves of 5 algorithms

Figure 4: Results of the experiments on Tetris

Conclusion

In this paper we propose CSV-LEARNING for solving average reward MDPs to make the learning process more stable and faster. In CSV-LEARNING, no extra exploration strategy is needed. We find that if the CSV lies below but close to the optimal average reward, the learned Q-values could be unbounded but the derived policy can converge to the optimum. We thus develop a mechanism to terminate the algorithm as soon as the policy is stable. We also prove the convergence of the proposed method. In addition, we conduct extensive experiments and demonstrate the efficiency of the proposed algorithm as compared with existing approaches.

Acknowledgements

The authors would like to acknowledge the support from National Natural Science Foundation of China (Grant No. 61432008, 61503178, 61403208, 61175042, 61321491) and Natural Science Foundation of Jiangsu Province, China (Grant No. BK20150587).

References

Abounadi, J.; Bertsekas, D.; and Borkar, V. S. 2001. Learning algorithms for Markov decisionprocesses with average cost. *SIAM Journal on Control and Optimization* 40(3):681–698.

Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press Cambridge.

Das, T. K.; Gosavi, A.; Mahadevan, S.; and Marchalleck, N. 1999. Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science* 45(4):560–574.

Demaine, E. D.; Hohenberger, S.; and Liben-Nowell, D. 2003. Tetris is hard, even to approximate. In *Annual International Conference on Computing and Combinatorics (COCOON)*. 351–363.

Ghavamzadeh, M., and Mahadevan, S. 2007. Hierarchical average reward reinforcement learning. *Journal of Machine Learning Research* 8:2629–2669.

Gosavi, A. 2004. Reinforcement learning for long-run average cost. *European Journal of Operational Research* 155(3):654–674.

Jaksch, T.; Ortner, R.; and Auer, P. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research* 11:1563–1600.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: a survey. *Journal of Artificial Intelligence ReMsearch* 237–285.

Li, Y. 2012. Reinforcement learning algorithms for semi-Markov decision processes with average reward. In *IEEE International Conference on Networking, Sensing and Control (ICNSC)*, 157–162.

Mahadevan, S. 1996. Average reward reinforcement learning: foundations, algorithms, and empirical results. *Machine Learning* 22(1-3):159–195.

Mahadevan, S. 2014. To discount or not to discount in reinforcement learning: a case study comparing R learning and Q learning. In *International Conference on Machine Learning (ICML)*, 164–172.

Marbach, P.; Mihatsch, O.; and Tsitsiklis, J. N. 2000. Call admission control and routing in integrated services networks using neuro-dynamic programming. *IEEE Journal on Selected Areas in Communications* 18(2):197–208.

Nguyen, D. T.; Yeoh, W.; Lau, H. C.; Zilberstein, S.; and Zhang, C. 2014. Decentralized multi-agent reinforcement learning in average-reward dynamic DCOPs. In *International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 1341–1342.

Ortner, P., and Auer, R. 2007. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, 49–56.

Ortner, R. 2013. Adaptive aggregation for reinforcement learning in average reward Markov decision processes. *Annals of Operations Research* 208(1):321–336.

Osband, I.; Russo, D.; and van Roy, B. 2013. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems (NIPS)*, 3003–3011.

Schwartz, A. 1993. A reinforcement learning method for maximizing undiscounted rewards. In *International Conference on Machine Learning (ICML)*, 298–305.

Singh, S. P. 1994. Reinforcement learning algorithms for average-payoff Markovian decision processes. In *AAAI Conference on Artificial Intelligence*, 700–705.

Sutton, R. S.; Maei, H. R.; Precup, D.; Bhatnagar, S.; Silver, D.; Szepesvári, C.; and Wiewiora, E. 2009. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning (ICML)*, 993–1000.

Sutton, R. S.; Maei, H. R.; and Szepesvári, C. 2009. A convergent $O(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 1609–1616.

Tadepalli, P. 2010. Average-reward reinforcement learning. In *Encyclopedia of Machine Learning*. 64–68.