TBQ(σ): Improving Efficiency of Trace Utilization for Off-Policy Reinforcement Learning

Longxiang Shi
College of Computer Science and Technology, Zhejiang University
Hangzhou, Zhejiang Province, China
shilongxiang@zju.edu.cn

Shijian Li*
College of Computer Science and Technology, Zhejiang University
Hangzhou, Zhejiang Province, China
shijianli@zju.edu.cn

Longbing Cao
Advanced Analytics Institute
University of Technology Sydney
Sydney, NSW, Australia
longbing.cao@uts.edu.au

Long Yang
College of Computer Science and Technology, Zhejiang University
Hangzhou, Zhejiang Province, China
yanglong@zju.edu.cn

Gang Pan
College of Computer Science and Technology, Zhejiang University
Hangzhou, Zhejiang Province, China
gpan@zju.edu.cn

ABSTRACT

Off-policy reinforcement learning with eligibility traces is challenging because of the discrepancy between target policy and behavior policy. One common approach is to measure the difference between two policies in a probabilistic way, such as importance sampling and tree-backup. However, existing off-policy learning methods based on probabilistic policy measurement are inefficient when utilizing traces under a greedy target policy, which is ineffective for control problems. The traces are cut immediately when a non-greedy action is taken, which may lose the advantage of eligibility traces and slow down the learning process. Alternatively, some non-probabilistic measurement methods such as General Q(λ) and Naive Q(λ) never cut traces, but face convergence problems in practice. To address the above issues, this paper introduces a new method named TBQ(σ), which effectively unifies the tree-backup algorithm and Naive Q(λ). By introducing a new parameter σ to illustrate the *degree* of utilizing traces, TBQ(σ) creates an effective integration of TB(λ) and Naive Q(λ) and continuous role shift between them. The contraction property of TB(σ) is theoretically analyzed for both policy evaluation and control settings. We also derive the online version of TBQ(σ) and give the convergence proof. We empirically show that, for $\epsilon \in (0, 1]$ in ϵ-greedy policies, there exists some degree of utilizing traces for $\lambda \in [0, 1]$, which can improve the efficiency in trace utilization for off-policy reinforcement learning, to both accelerate the learning process and improve the performance.

KEYWORDS

Reinforcement learning; Eligibility traces; Deep learning

ACM Reference Format:
Longxiang Shi, Shijian Li, Longbing Cao, Long Yang, and Gang Pan. 2019. TBQ(σ): Improving Efficiency of Trace Utilization for Off-Policy Reinforcement Learning. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 8 pages.

*Corresponding author.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

1 INTRODUCTION

As a basic mechanism in reinforcement learning (RL), eligibility traces [18] unify and generalize temporal-difference (TD) and Monte Carlo methods [20]. As a temporary record of an event (e.g., taking an action or visiting a state) in RL, eligibility traces mark the memory parameters associated with the event as eligible for undergoing changes [19]. The eligible traces are then used to assign credit to the current TD-error which leads the learning of policies. With traces, credit is passed through multiple preceding states and therefore learning is often significantly faster [16].

With the on-policy TD learning with traces (e.g., TD(λ), Sarsa(λ)), the assignment of credit to previous states decays exponentially according to the parameter $\lambda \in [0, 1]$. If $\lambda = 0$, the traces are set to zero immediately and the on-policy TD learning algorithm with traces is equal to one-step TD learning. If $\lambda = 1$, the traces fade away slowly and no bootstrapping is made, and thus producing the Monte Carlo algorithm with online update [17]. Moreover, the intermediate value of λ makes the learning algorithm to perform better than the method at either extreme.

In the off-policy case, when the samples generated from a behavior policy is used to learn a different target policy, the usual approach is to measure the difference of the two policies in a probabilistic way. For example, Per-Decision Importance Sampling [5] weights returns based on the mismatch between target and behavior probabilities of the related actions. Alternatively, Tree-backup (TB) algorithm [5] combines the value estimates for the actions along the traces according to their probabilities of target policy. More recently, Retrace(λ) [13] combines Naive Q(λ) with importance sampling, and offers a safe (whatever the behavior policy is) and efficient (can learn from full returns) way for off-policy reinforcement learning. However, existing off-policy learning methods based on state-action probability are inefficient when utilizing the traces for off-policy learning, especially when the target policy is deterministic, which is quite obvious in control problems. If the target policy is deterministic, the probability of target policy is zero when an exploratory action is taken. In this setting, importance sampling always involves a large variance since the importance ratio may be greater than 1 and is rarely used in practice. Retrace(λ) and TB(λ) is identical to Watkins' Q(λ) [23] and the traces are cut when an

exploratory action is taken. This may cause to lose the advantage of eligibility traces and slow down the learning process [19]. Peng's Q(λ) [14] tried to solve this problem, but fails to converge to the optimal value.

On the other hand, some existing methods do not depend on target policy probabilities and can learn from full returns without cutting traces under the greedy target policy. Unfortunately, some of them may face limitations in convergence. For instance, Naive Q(λ) [19] never cuts traces thus provides a way to use full returns when performing off-policy RL with eligibility traces, which can sometimes achieve a better performance over Watkins' Q(λ) [9]. A more recent work by [6] shows that Naive Q(λ) for control can converge to the optimal value under some conditions. An open question is: how about the intermediate condition between target policy probabilities-based and non-target policy probabilities-based methods?

To address the above question, in this paper we propose a TBQ(σ) algorithm, which unifies TB(λ) (cutting traces immediately) and Naive Q(λ) (never cutting traces). By introducing a new parameter σ to illustrate the *degree* of utilizing traces, TBQ(σ) creates a continuous integration and role shift between TB(λ) and Naive Q(λ). If $\sigma = 1$ then TBQ(σ) is converted to the Naive Q(λ) that never cuts traces; and if $\sigma = 0$ then TBQ(σ) is transformed to the Watkins' Q(λ). We then theoretically analyze the contraction property of TB(σ) for both policy evaluation and control settings. We also derive the online version of TBQ(σ) and give the convergence proof. Compared to TB(λ), TBQ(σ) is efficient in trace utilization with the greedy target policy. Compared to Naive Q(λ), TBQ(σ) can achieve convergence by adjusting a suitable σ. We empirically show that, for $\epsilon \in (0, 1]$ in ϵ-greedy policies, there exists some degree of utilizing traces for $\lambda \in [0, 1]$, which can improve the efficiency in trace utilization, therefore accelerating the learning process and improving the performance as well.

2 PRELIMINARIES AND PROBLEM SETTINGS

Here, we introduce some basic concepts, our target problems, notations, and related work.

2.1 Preliminaries and Problem Settings

A reinforcement learning problem can be formulated as a Markovian Decision Process (MDP) (S, A, γ, P, r), where S is a finite state space, A is the action space, $\gamma \in [0, 1]$ is the discount factor and P is the mapping of transition function for each state-action pair $(s, a) \in (S, A)$ to a distribution over S. A policy π is a probability distribution over the set $(S \times A)$.

The state-action value Q is a mapping on $S \times A$ to $\mathbb{R}$, which indicates the expected discounted future reward when taking action a at state s under policy π:

$$Q(s, a) := \mathbb{E}_\pi(r_1 + \gamma r_2 + \ldots + \gamma^{T-1} r_T | s_0 = s, a_0 = a) \tag{1}$$

where T is the time of termination. For each policy π, we define the operator P^π [6]:

$$(P^\pi Q)(s, a) := \sum_{s' \in S} \sum_{a' \in A} P(s'|s, a)\pi(a'|s')Q(s', a')$$

For an arbitrary policy π we use Q^π to describe the unique Q-function corresponding to π:

$$Q^\pi := \sum_{t \geq 0} \gamma^t (P^\pi)^t r$$

The Bellman operator $\mathcal{T}^\pi$ for a policy π is defined as:

$$\mathcal{T}^\pi Q := r + \gamma P^\pi Q \tag{2}$$

Obviously, $\mathcal{T}^\pi$ has a unique fixed point Q^π:

$$\mathcal{T}^\pi Q^\pi = Q^\pi = (I - \gamma P^\pi)^{-1} r \tag{3}$$

The Bellman optimality operator $\mathcal{T}$ introduces a maximization over a set of policies and is defined as:

$$\mathcal{T} Q := r + \gamma \max_\pi P^\pi Q \tag{4}$$

Its unique fixed point is $Q^* := \sup_\pi Q^\pi$.

The Bellman equation can also be extended using the exponentially weighted sum of n-step returns [18]:

$$\begin{aligned} \mathcal{T}_\lambda^\pi &:= (1 - \lambda) \sum_{n \geq 0} \lambda^n [(\mathcal{T}^\pi)^n Q] \\ &= Q + (I - \lambda \gamma P^\pi)^{-1} (\mathcal{T}^\pi Q - Q) \end{aligned} \tag{5}$$

In this λ-return version of Bellman equation, the fixed point of $\mathcal{T}_\lambda^\pi$ is also Q^π. By varying the parameter λ from 0 to 1, $\mathcal{T}_\lambda^\pi$ provides a continuous connection and role shift between one-step TD learning and Monte Carlo methods.

In this paper, we consider two types of RL problems, and mainly focus on action-value case under the off-policy setting. That is, in a *policy evaluation* problem, we wish to estimate Q^π of a fixed policy π under the samples drawn from a different behavior policy μ; in a *control* problem, we seek to approximate Q^* based on the iteration of Q-values. We specially focus on the learning scenario that the target policy is greedy, which is obvious in the control setting. Our main challenge is to improve the efficiency of trace utilization as well as ensure learning convergence during the off-policy learning process.

2.2 Related Work

Based on the usage of target policy probability when calculating the λ-return, existing works can be divided into 2 categories:

2.2.1 Target policy probability-based methods. The n-step methods face challenges when involving off-policy, which has triggered to produce many methods to solve those challenges. The most common approach is to measure the two policies in a probabilistic sense[11]. Based on the work in [13], several off-policy return-based methods based on target policy probability: importance sampling (IS), tree-backup and Retrace(λ) can be expressed in a unified operator $\mathcal{R}$ as follows:

$$\begin{aligned} \mathcal{R}Q(s, a) &:= Q(s, a) + \mathbb{E}_\mu[\sum_{t \geq 0} \gamma^t (\prod_{i=1}^{t} c_i)\delta_t] \\ \delta_t &= r_t + \gamma \mathbb{E}_\pi Q(s_{t+1}, \cdot) - Q(s_t, a_t) \end{aligned} \tag{6}$$

Importance sampling: $c_i = \frac{\pi(a_i|s_i)}{\mu(a_i|s_i)}$. The IS methods correct the difference between target policy and behavior policy by their division of probabilities [20]. For example, Per-Decision Importance Sampling (PDIS) [5] incorporates eligibility traces with importance

sampling. Since the estimation value contains a cumulative production of importance rations (c_s) which may exceeds 1, IS methods suffer from large variance and are seldom used in practice. In addition, weighted importance sampling [15] can reduce the variance of IS, but leads to a biased estimation.

Tree-backup: $c_i = \lambda\pi(a_i|s_i)$. The TB($\lambda$) algorithm [5] provides an alternative way for off-policy learning without IS. In control problems, if the target policy is greedy, then TB(λ) produces Watkins' Q(λ) [22]. In this case, TB(λ) is not efficient as it cuts traces when encountered an exploratory action and is not able to learn from the full returns.

Retrace(λ): $c_i = \lambda \min(1, \frac{\pi(a_i|s_i)}{\mu(a_i|s_i)})$, was proposed in [13]. Comparing to IS methods, this method truncates the importance ration by 1 to reduce the variance in IS. It is proved to convergence under any behavior policy and can learn from full returns when the behavior and target policies are near. However, in the control case when the target policy is greedy, Retrace(λ) is identical to TB(λ) and is not efficient in utilizing traces.

2.2.2 Non-target policy probability-based methods. In addition, there are also some methods that does not depend on target policy probability, and can make full use of the traces:

General Q(λ): General Q(λ) [21][7] generalizes the on-policy Sarsa(λ) using the following update equation:

$$\begin{aligned} Q(s_t,a_t) \leftarrow & Q(s_t,a_t) + \alpha[\sum_{i\geq t}^{T}(\lambda\gamma)^{i-t}\delta_t + \mathbb{E}_\pi Q(s_{t+1},\cdot) \\ & -Q(s_t,a_t)] \\ \delta_t = & r_t + \gamma\mathbb{E}_\pi Q(s_{t+1},\cdot) - \mathbb{E}_\pi Q(s_t,a_t) \end{aligned}$$

In control case, when target policy is greedy, General Q(λ) is identical to Peng's Q(λ) [14]. It does not cut traces so much as Watkins' Q(λ). However, When learning is off-policy, General Q(λ) lead to a biased estimation and does not converge to Q^π.

Q(λ) with off policy corrections [6]: it is an off-policy correction method based on a Q-baseline. Their proposed operator $\mathcal{R}_\lambda^{\pi,\mu}$ is the same as $\mathcal{R}$ if $c_i = \lambda$ in (6). Their algorithms, named $Q^\pi(\lambda)$ and $Q^*(\lambda)$ for policy evaluation and control, respectively. If the distance $d = \max_s \|\pi(\cdot|s) - \mu(\cdot|s)\|$ between target policy π and behavior policy μ is small, i.e., $d < \frac{1-\gamma}{\gamma}$, $Q^\pi(\lambda)$ converges to its fixed point Q^π. In control scenarios, $Q^*(\lambda)$ is equal to Naive $Q(\lambda)$ [19] and is guaranteed to converge to Q^* under $\lambda < \frac{1-\gamma}{2\gamma}$. Besides, they also empirically show that in fact there exists some trade-off between d and λ beyond the convergence guarantee, which can make the learning faster and better. In addition, $Q^\pi(\sigma,\lambda)$ is proposed in [24] to combine Sarsa(λ) and $Q^\pi(\lambda)$, and inherit the similar properties with $Q^\pi(\lambda)$.

In conclusion, existing off-policy learning methods based on target policy probability are inefficient when utilizing eligibility traces, especially when target policy is greedy. In this scenario, The traces are cut immediately when encountered an exploratory action and thus may lose the advantage of eligibility traces and slow down the learning process. In addition, existing non-target policy probability based methods can make full use of the traces, but may face limitations in convergence. In this paper, we try to solve this dilemma by create a hybridization of those two different methods.

3 TBQ(σ): DEGREE OF TRACES UTILIZATION

In the RL literature, unifying different algorithmic ideas to leverage the pros and cons in each idea and to produce better algorithms has been a pragmatic approach [4]. This also applies to several policy learning methods, e.g., TD(λ) to unify TD-learning and Monte Carlo methods, Q(σ) [4] to fuse multi-step tree-backup and Sarsa, and Q(σ,λ) [24] to integrate $Q^\pi(\sigma)$ and Sarsa(λ). Such hybridization is useful for balancing the capabilities of different trace-cutting methods discussed above. Accordingly, in this paper, we introduce a new parameter σ into trace-cutting to enable the degree of utilizing traces. The proposed method, TBQ(σ), unifies TB(λ) (cutting traces immediately) and Naive Q(λ) (never cutting traces).

We first give the definition of operator that used for the update equation of TBQ(σ):

Definition 3.1. The proposed operator $\mathcal{R}_\sigma$ is a map on $\mathbb{R}^{|S|\times|A|}$ to $\mathbb{R}^{|S|\times|A|}$, $\forall s\in S, a\in A, \sigma \in [0,1]$:

$$\begin{aligned} \mathcal{R}_\sigma :& \mathbb{R}^{|S|\times|A|} \leftarrow \mathbb{R}^{|S|\times|A|} \\ & Q(s,a) := Q(s,a) + \mathbb{E}_\mu[\sum_{t\geq 0}\gamma^t(\prod_{i=1}^{t} c_i)\delta_t] \end{aligned} \tag{7}$$

where

$$\begin{aligned} c_i &= \lambda[\sigma + (1-\sigma)\pi(a_i|s_i)] \\ \delta_t &= r_t + \gamma\mathbb{E}_\pi Q(s_{t+1},\cdot) - Q(s_t,a_t) \end{aligned}$$

TBQ(σ) linearly combines TB(λ) and Naive Q(λ) by using the degree parameter σ. When $\sigma = 0$ then TBQ(σ) is converted to TB(λ), and $\sigma = 1$ TBQ(σ) is transformed to Naive Q(λ). By exploratory adjusting the parameter σ from 0 to 1 we can produce a continuous integration and role shift between cutting the traces immediately and never cutting traces. We then analyze the contraction property of $\mathcal{R}_\sigma$ in policy evaluation. We here use $\|\cdot\|$ to represent the supremum norm.

Theorem 3.2. *The proposed operator $\mathcal{R}_\sigma$ has a unique fixed point Q^π. If the behavior policy and target policy are near, i.e.,* $d = \max_x \|\pi(\cdot|x) - \mu(\cdot|x)\| < (1-\gamma)[\frac{1}{\gamma\lambda} + 1 - \sigma]$, *then* $\|\mathcal{R}_\sigma Q - Q^\pi\| = O(\eta^k)$.

Proof. Unfolding the operator:

$$\mathcal{R}_\sigma Q - Q^\pi = \sigma(\mathcal{R}_\lambda^{\pi,\mu} Q - Q^\pi) + (1-\sigma)(\mathcal{R}Q - Q^\pi)$$

Taking the supremum norm:

$$\begin{aligned} \|\mathcal{R}_\sigma Q - Q^\pi\| &= \|\sigma(\mathcal{R}_\lambda^{\pi,\mu} Q - Q^\pi) + (1-\sigma)(\mathcal{R}Q - Q^\pi)\| \\ &\leq \sigma\|\mathcal{R}_\lambda^{\pi,\mu} Q - Q^\pi\| + (1-\sigma)\|\mathcal{R}Q - Q^\pi\| \end{aligned}$$

Per Lemma 1 in [6] we have:

$$\|\mathcal{R}_\lambda^{\pi,\mu} Q - Q^\pi\| \leq \frac{\gamma(1-\lambda+\lambda d)}{1-\lambda\gamma}\|Q - Q^\pi\|$$

where d is the distance between π and μ:

$$\max_x \|\pi(\cdot|x) - \mu(\cdot|x)\| \leq d$$

Per Theorem 1 in [13] we have:

$$\|\mathcal{R}Q - Q^\pi\| \le \gamma\|Q - Q^\pi\|$$

Adding the above two items we have:

$$\|\mathcal{R}_\sigma Q - Q^\pi\| \le \eta\|Q - Q^\pi\|$$

where $\eta = \frac{\gamma - \lambda\gamma^2 + \lambda\sigma\gamma^2 + \sigma\gamma\lambda d - \sigma\gamma\lambda}{1-\gamma\lambda}$.
Further, for $d < (1-\gamma)[\frac{1}{\gamma\lambda} + 1 - \sigma]$, $\eta < 1$, we have

$$\|\mathcal{R}_\sigma Q - Q^\pi\| = O(\eta^k)$$

□

Theorem 3.2 indicates that, for any $\lambda \in [0,1]$, if the distance between two policies are near with regard to σ, then Q_k converges to Q^π. Comparing to $Q^\pi(\lambda)$ [6], our algorithm derives a wider convergence range w.r.t σ. We provide a hybridization of utilizing traces based on TB(λ) and Naive Q(λ). In practice, the convergence condition can be satisfied by adjusting the parameter σ under different situations.

4 TBQ(σ) FOR CONTROL

In control problems, we want to estimate Q^* by iteratively applying policy evaluation and policy improvement processes, which is referred to *generalized policy iteration* (GPI) [19]. Denoting (Q_k, π_k) as the Q-value and the corresponding target policy in the iteration process under the arbitrary behavior policy μ_k at step k, then π_{k+1} can be retrieved by our operator $\mathcal{R}_\sigma^{\pi,\mu}$ by using the following steps:

- Policy evaluation step:

$$Q_{k+1} = \mathcal{R}_\sigma^{\pi_k,\mu_k} Q_k$$

- Policy improvement step:

$$\pi_{k+1} = greedy(Q_{k+1})$$

We here use the notion $greedy(Q_k)$ to represent π_k, which is greedy with respect to Q_k. Based on GPI, the TBQ(σ) algorithm for control problems is depicted in Algorithm 1 with an online forward view, i.e., TBQ$^F(\sigma)$. Note that $\mathbb{I}\{(s_t, a_t) = (s,a)\}$ is the indicator function.

To analyze the convergence of Algorithm 1, we first consider off-line version of the TBQ(σ) algorithm. The following lemma states that, if σ satisfies some condition with regard to λ, then the off-line version of TBQ(σ) is guaranteed to converge.

Lemma 4.1. *Considering the sequence $\{(Q_k, \pi_k)\}_{k\ge0}$ generated by the operator $\mathcal{R}_\sigma$ under a greedy target policy π_k and an arbitrary behavior policy μ, we have:*

$$\|Q_{k+1} - Q^*\| \le \eta\|Q_k - Q^*\|$$

where $\eta = \frac{\sigma\gamma + \sigma\lambda\gamma}{1-\lambda\gamma} + (1-\sigma)\gamma$.

Specifically, if $\lambda \le \frac{1-\gamma}{\sigma\gamma + \sigma\gamma^2 + \gamma - \gamma^2}$, then the sequence $\{Q_k\}_{k\ge1}$ converges to Q^* exponentially fast.

Proof. Unfolding the operator:

$$\begin{aligned}\|Q_{k+1} - Q^*\| &= \|\mathcal{R}_\sigma Q_k - Q^*\| \\ &\le \sigma\|\mathcal{R}_\lambda^{\pi,\mu} Q_k - Q^*\| + (1-\sigma)\|\mathcal{R}Q_k - Q^*\|\end{aligned}$$

based on [6] and [13], we have:

$$\|\mathcal{R}_\lambda^{\pi,\mu} Q_k - Q^*\| \le \frac{\gamma + \lambda\gamma}{1-\lambda\gamma}\|Q_k - Q^*\|$$

$$\|\mathcal{R}Q_k - Q^*\| \le \gamma\|Q_k - Q^*\|$$

As a consequence, we deduce the result:

$$\begin{aligned}\|Q_{k+1} - Q^*\| &= \|\mathcal{R}_\sigma Q_k - Q^*\| \\ &\le \sigma\|\mathcal{R}_\lambda^{\pi,\mu} Q_k - Q^*\| + (1-\sigma)\|\mathcal{R}Q_k - Q^*\| \\ &\le \sigma\frac{\gamma + \lambda\gamma}{1-\lambda\gamma}\|Q_k - Q^*\| + (1-\sigma)\gamma\|Q_k - Q^*\| \\ &= [\frac{\sigma\gamma + \sigma\gamma\lambda}{1-\lambda\gamma} + (1-\sigma)\gamma]\|Q_k - Q^*\|\end{aligned}$$

□

Lemma 4.1 states that, for any d, if $\lambda \le \frac{1-\gamma}{\sigma\gamma + \sigma\gamma^2 + \gamma - \gamma^2}$ then the off-line control algorithm is guaranteed to converge. However, similar to $Q^*(\lambda)$ [6], in practice, there exist some trade-offs between λ and σ under different d values, which goes beyond the convergence guarantee. By introducing a new parameter σ, we can alleviate $\lambda - d$ relationship through adjusting a suitable σ. The traces can also be utilized when an exploratory action is taken. In addition, comparing to Naive Q(λ), we derive a wider convergence range by tuning σ. Although we have not give a detail theoretical analyze of $\lambda - d$ relationship under different σ, in the experiment part we will show that for any $\lambda \in [0,1]$ and $\epsilon \in [0,1]$ in $\epsilon - greedy$ policies, there exist some degree of utilizing traces σ, which can accelerate the learning process and yield a better performance through utilizing the full returns as well.

Algorithm 1 TBQ$^F(\sigma)$: The online forward view version of TBQ(σ) algorithm

Input: discounting factor γ, degree of utilizing traces σ, bootstrapping parameter λ, and stepsize α_k
Initialization: $Q_0(s,a)$ arbitrary
for Episode k from 1 to n **do**
  $Q_{k+1}(s,a) \leftarrow Q_k(s,a)\ \forall(s,a)$
  $e(s,a) \leftarrow 0\ \forall(s,a)$
  Sample a trajectory $s_0, a_0, r_0, ..., x_{T_k}$ from μ_k
  for Sample t from 0 to $T_k - 1$ **do**
    $\delta_t^{\pi_k} \leftarrow r_t + \gamma \max_{a'} Q_{k+1}(s_{t+1}, a') - Q_{k+1}(s_t, a_t)$
    $c_t = \begin{cases} 1 & t = 0 \\ \sigma + (1-\sigma)\pi(a_t|s_t) & t \ne 0 \end{cases}$
    $e(s,a) \leftarrow \lambda\gamma c_t e(s,a) + \mathbb{I}\{(s_t,a_t) = (s,a)\}\ \forall(s,a)$
    $Q_{k+1} \leftarrow Q_{k+1} + \alpha\delta_t^{\pi_k} e(s,a)\ \forall(s,a)$
  end for
end for

4.1 Convergence Analysis of TBQ(σ) Algorithm

We now consider the convergence proof of TBQ(σ) described in Algorithm 1. First, we make some assumptions similar to [6] [13].

Assumption 1. *For bounded stepsize α_k: $\sum_{k\ge0} \alpha_k(s,a) = \infty$, $\sum_{k\ge0} \alpha_k(s,a) < \infty$.*

Assumption 2. *Minimum visit frequency: all (s,a) pairs are visited infinitely often:* $\sum_{t\geq 0} P\{(s_t,a_t)=(s,a)\} \geq D > 0$.

Assumption 3. *Finite sample trajectories:* $\mathbb{E}_{\mu_k}(T_k^2) < \infty$, T_k *denotes the length of sample trajectories.*

Under those assumptions, Algorithm 1 can converge to Q^* with probability 1 as stated below:

Theorem 4.2. *Considering the sequence of Q-functions* $\{(Q_k,\pi_k)\}_{k\geq 0}$ *generated from Algorithm 1, where* π_k *is the greedy policy with respect to* Q_k*, if* $\lambda \leq \frac{1-\gamma}{\sigma\gamma+\sigma\gamma^2+\gamma-\gamma^2}$*, then under Assumptions 1-3,* $Q_k \to Q^*$ *with probability 1.*

Proof. For reading convenience, we first define some notations: Let k denote the kth iteration, t denote the length of the trajectory, l denote the lth sample of current trajectory, then the accumulating trace [19] $z_{l,t}^k$ can be written as:

$$z_{l,t}^k = \sum_{j=k}^{t} \gamma^{t-j}\Big(\prod_{i=j+1}^{t} c_i\Big)\mathbb{I}\{(s_j,a_j)=(s_l,a_l)\} \tag{8}$$

We use $Q_k^o(s_l,a_l)$ to emphasize the online setting, then Equation (7) can be written as:

$$Q_{k+1}^o(s_l,a_l) \leftarrow Q_k^o(s_l,a_l) + \alpha_k(s_l,a_l)\sum_{t\geq l}\sigma_t^{\pi_k} z_{l,t}^k \tag{9}$$

$$\delta_t^{\pi_k} = r_t + \gamma\mathbb{E}_{\pi_k} Q_k^o(s_{t+1},\cdot) - Q_k^o(s_t,a_t) \tag{10}$$

Since $c_i = \lambda[\sigma + (1-\sigma)\pi(a_i|s_i)] \leq 1$, based on Assumption 3, we have:

$$\mathbb{E}[\sum_{t\geq l} z_{l,t}^k] < \mathbb{E}[T_k^2] < \infty \tag{11}$$

Therefore, the total update is bounded based on Equation (11). Further, we can rewrite the update Equation (9) as:

$$\begin{aligned}
Q_{k+1}^o(s_l,a_l) &\leftarrow (1-D_k\alpha_k)Q_k^o(s_l,a_l) + D_k\alpha_k(\mathcal{R}_\sigma^{\pi_k}Q_k^o(s_l,a_l) \\
&\quad + w_k(s_l,a_l) + v_k(s_l,a_l)) \\
w_k(s_l,a_l) &:= (D_k)^{-1}[\sum_{t\geq l}\delta_t^{\pi_k} z_{l,t}^k - \mathbb{E}_{\mu_k}(\sum_{t\geq l}\delta_t^{\pi_k} z_{l,t}^k)] \\
v_k(s_l,a_l) &:= (D_k\alpha_k)^{-1}(Q_{k+1}^o(s_l,a_l) - Q_{k+1}(s_l,a_l)) \\
D_k &:= D_k(s_l,a_l) = \sum_{t\geq l} P\{(s_t,a_t)=(s_l,a_l)\}
\end{aligned}$$

Based on Assumptions 1 and 2, the new stepsize $(D_k\alpha_k)$ satisfies Assumption (a) of Proposition 4.5 in [2]. Lemma 4.1 states that the operator $\mathcal{R}_\sigma$ is a contraction, which satisfies Assumption (c) of Proposition 4.5 in [2]. Based on Equation (7) and the bounded reward function, the variance noise term w_k is bounded, thus Assumption (b) of Proposition 4.5 in [2] is satisfied. The noise term v_k can also be shown to satisfy Assumption (d) of Proposition 4.5 in [2], based on Proposition 5.2 in [2]. Finally, we are able to apply Proposition 4.5 [2] to conclude that the sequence Q_k^o converges to Q^* with probability 1. □

4.2 Online Backward Version of TBQ(σ)

Since the online forward view algorithm described in Algorithm 1 needs extra memory to store the trajectories, we here also provide an online backward version of TBQ(σ): TBQ$^B(\sigma)$. Based on the equivalence between forward view and backward view of the eligibility traces [19], the online backward view version of TBQ(σ) can be implemented as in Algorithm 2. The online backward view version TBQ$^B(\sigma)$ provides a more concise and efficient form and it is more efficient in executing the TBQ(σ) algorithm.

Algorithm 2 TBQ$^B(\sigma)$: On-line backward version of TBQ(σ) algorithm

Input: discounting factor γ, degree of cutting traces σ, bootstrapping parameter λ and stepsize α
Initialization: $Q(s,a)$ arbitrary
for k from 1 to n **do**
  Initialize s,a
  $e(s,a) = 0\ \forall(s,a)$
  repeat
    Take action a, observe state s' and receive reward r
    Choose a' from s' using ϵ-greedy policy μ based on $Q(s,a)$
    $a^* \leftarrow \arg\max_b Q(s',b)$
    $\delta \leftarrow r + \gamma \max_b Q(s',b) - Q(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    for all s,a **do**
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
      if $a^* = a'$ **then**
        $e(s,a) \leftarrow \gamma\lambda e(s,a)$
      else
        $e(s,a) \leftarrow \sigma\gamma\lambda e(s,a)$
      end if
    end for
    $s \leftarrow s'$,$a \leftarrow a'$
  until s is terminal
end for

5 EXPERIMENTS

In this section, we explore the $\lambda - \sigma$ trade-off in the control case w.r.t. several environments. We empirically find that, for $\lambda \in [0,1]$ and $\epsilon \in [0,1]$, there exists some degree of utilizing traces σ, which can improve the efficiency of trace utilization.

5.1 19-State Random Walk

The 19-state random walk problem is a one-dimensional MDP environment which is widely used in RL [20][4]. There are two terminal states at the two ends of the environment, transition to the left terminal receives a reward 0 and to the right terminal receives 1. The agent at each state has two actions: left and right. We here apply the online forward version TBQ$^F(\sigma)$ by using an $\epsilon - greedy$ policy as behavior policy and a greedy policy as target policy. For each episode, the maximum step is bounded as 100. We then measure the mean-squared-error (MSE) of the optimal Q-value Q^* between the estimated values and the analytically computed values after 10,000 episodes of offline running. We test 3 different ϵ values: 0.1, 0.5, 1.

The corresponding distance d between target policy and behavior policy is 0.05, 0.25, 0.5, respectively. For each ϵ, we test different λ values from 0 to 1 with stepsize 0.1. Also, for each λ, we also try different σ values from 0 to 1 with stepsize 0.1. The learning stepsize α is tuned to 0.3. All results are averaged across 10 independent runs with fixed random seed. We compare TBQ(σ) with TB(λ) and Naive Q(λ). For TBQ(σ), we also mark out the best performance of σ, with the results shown in Figure 1.

Figure 1(a) shows that $\epsilon = 0.1$ is too small for the agent to explore the whole environment. The agent can seldom reach the left terminal. In addition, since the exploratory action is also rarely taken, the MSEs of TB(λ) between different λ values vary a little. Naive Q(λ) never cuts traces and enjoys the convergence when $\lambda \leq 0.6$. When $\lambda > 0.6$, the Naive Q(λ) diverges. The MSEs of TBQ(σ) vary a little when Naive Q(λ) converged. When $\lambda > 0.6$, we can still tune σ to reach a lower MSE. The best σ of TBQ(σ) decreases as the increase of λ. When $\epsilon = 0.5$ (Figure 1(b)), we observe results similar to $\epsilon = 0.1$ when $\lambda \leq 0.6$. When Naive Q(λ) diverges, TBQ(σ) can also benefit from learning from the full returns by adjusting a suitable σ. The MSE can also be reduced as well. When $\epsilon = 1$, the behavior policy becomes completely random. The performance between TB(λ) and Naive Q(λ) is nearly the same when $\lambda \leq 0.7$. When $\lambda > 0.7$ we can also adjust a suitable σ to ensure the convergence of TBQ(σ).

In this experiment, we observe that when $\epsilon \in [0, 1]$, $\lambda \in [0, 1]$, we can adjust a suitable σ in order to learn from the full returns and avoid cutting traces too often as well. In practice, when λ is close to 0, σ can be set to 1 to make full use of the traces. When λ is close to 1, σ can be set to a small number near 0 to improve the efficiency of traces utilization.

5.2 10×10 Maze Environment

The Maze environment is a 2-dimensional navigation task[1]. The agent's goal is to find the shortest path from start to the goal. For each state, the agent has 4 actions: go up, go down, turn left or turn right. If the path is blocked, the agent will stay at the current location. The reward is 1 when the agent reaches the goal, while at any intermediate state the agent gets reward -0.0001. Each episode is terminated if the agent reaches the goal, or the step count exceeds 2,000. To ensure adequate exploration and speed up the training process as well, we here adopt an $\epsilon - greedy$ policy as behavior policy and linearly decay the parameter ϵ from 1 to 0.1 by 0.02. In this experiment, we use the on-line backward version of TBQ(σ). The learning rate α is tuned to 0.05. We here use 6 different σ factors of TBQ(σ): {0, 0.2, 0.4, 0.6, 0.8, 1}, and measure the average total steps of each episode. In addition, the results are averaged across 10 independent runs with fixed random seeds.

The result is illustrated in Figure 2. Since the shortest path of the maze is deterministic, TBQ(σ) gradually accelerates the learning process when σ varies from 0 to 0.8. However, Naive Q(λ) diverges and cannot find the shortest path. The convergence speed of TBQ(σ) reaches fastest at $\sigma = 0.8$. The result shows that, in practice, we can accelerate the learning process by adjusting a suitable parameter σ based on the TBQ(σ) algorithm.

[1]We here use this version of gym-Maze environment: https://github.com/MattChanTK/gym-maze.

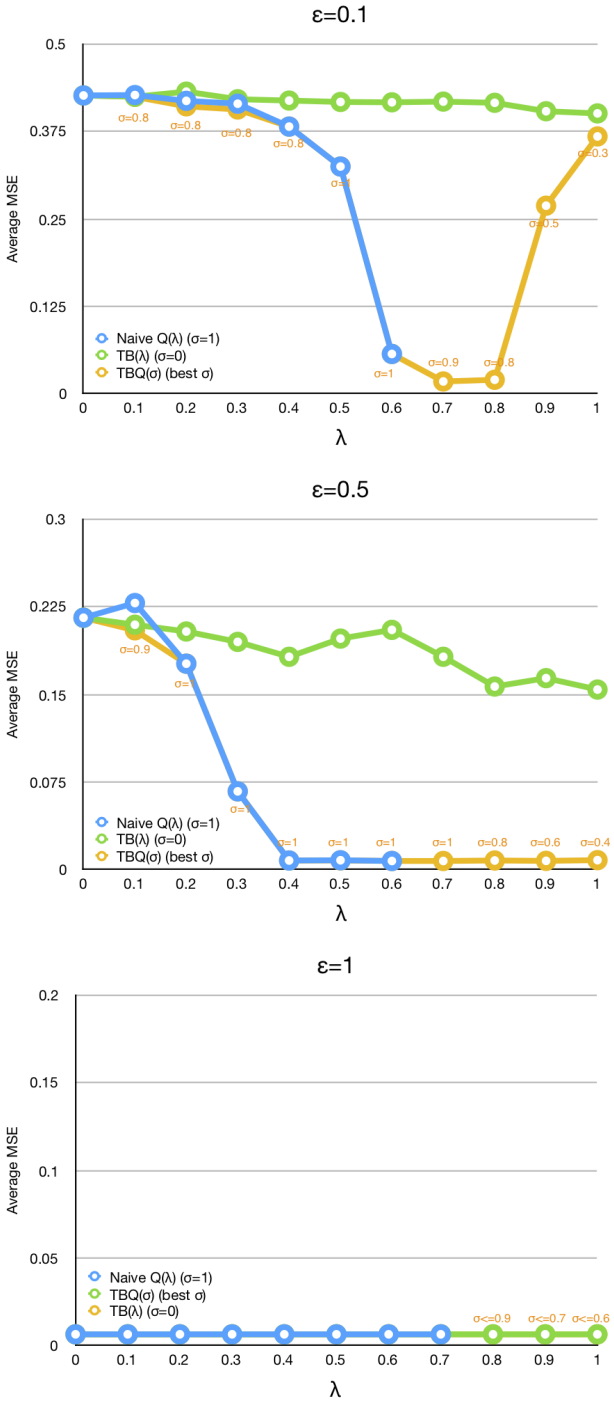

Figure 1: $\lambda - \sigma$ relationship under different ϵ values

5.3 TBQ(σ) with Function Approximator

We also evaluate TBQ(σ) algorithm using neural networks as function approximator. With the help of deep Q-neworks (DQN) [12],

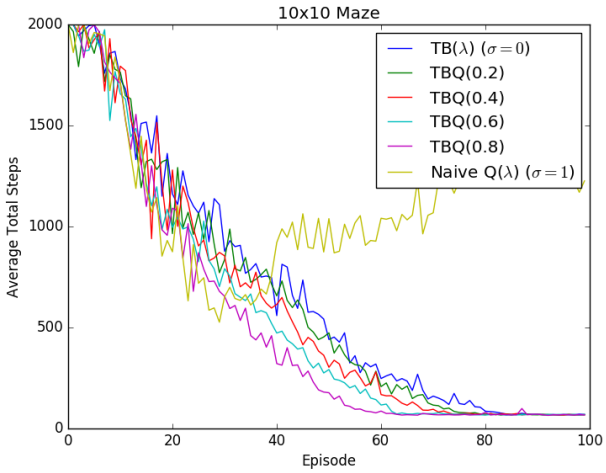

Figure 2: Averaged total steps of the Maze environment. TBQ(σ) gradually accelerates the learning process when σ varies from 0 to 0.8. However, Naive Q(λ) diverges and cannot find the shortest path.

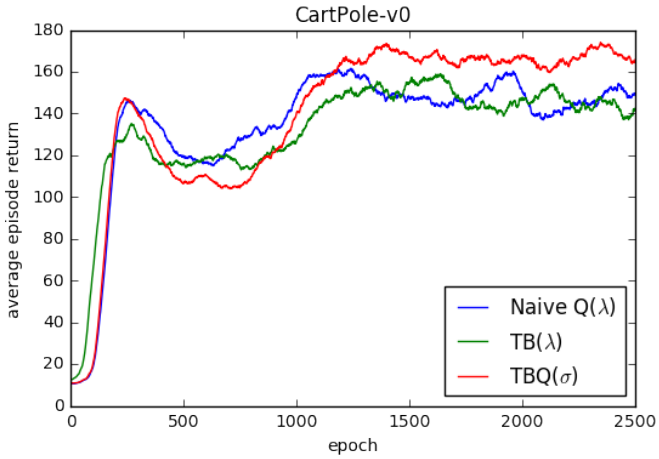

Figure 3: TBQ(σ) with function approximator in CartPole environment. The exploring parameter ϵ of ϵ- greedy policy decays from 1 to 0.1. To efficient utilize the traces under dynamic ϵ, σ is linearly decayed from 1 to 0.1 by step size 0.01. The result show that TBQ(σ) outperforms both TB(λ) and Naive Q(λ).

the offline version with a function approximator can be easily implemented. We here adopt online forward view for updating the parameters in the neural network. Unlike traditional DQN, we replay 4 consecutive sequences of samples with length of 8 for each update. We here evaluate TBQ(σ) on CartPole problem [1], and adopt the OpenAI Gym as the evaluation platform [2] [3]. In this setting, a pole is attached by an un-actuated joint to a cart, which can move along the track. The agent's goal is to prevent the pole from falling over with two actions controlling the cart: move left or right. Since the observation space is continuous, we adopt a two-layer neural network with 64 nodes in each layer to approximate the Q-value for the state action pairs. We use $\epsilon - greedy$ policy as behavior policy and exponentially decay the parameter ϵ from 1 to 0.1 by 0.995 to ensure adequate exploration. In addition, the target network parameters θ are updated using soft replacement [10] according to the evaluation network parameter θ': $\theta \leftarrow \tau\theta + (1-\tau)\theta'$.

Parameter	Value
Discount factor	0.99
Initial exploration	1
Final exploration	0.1
Optimizer	Adam[8]
Initial learning rate	0.001
Replay memory size	20000
Replay start episode	100
λ	1
τ	0.001

Table 1: Learning parameters for the neural network

[2]http: gym.openai.com

In this setting, in the beginning of the learning process the distance between target policy and behavior policy reach the maximum. When ϵ fades to 0.1, the two policy then become close. Therefore, to ensure convergence we here adopt a dynamic σ linearly increase from 0.1 to 1 by stepsize 0.01. Other main learning parameters are listed in Table 1. the results are averaged across 5 independent runs with fixed random seeds. The result is showed in Figure 3. We also smooth the results with a right-centred moving average of 50 successive episodes. With a dynamic suitable σ, TBQ(σ) outperforms TB(λ) and Naive Q(λ) in the CartPole problem. The result indicates that in practice, we can improve the learning by adjusting a suitable parameter σ using TBQ(σ) algorithm.

6 DISCUSSION AND CONCLUSION

In this paper, we propose a new off-policy learning method called TBQ(σ) to define the degree of utilizing the off-policy traces. TBQ(σ) unifies TB(λ) and Naive Q(λ). Theoretical analysis shows the contraction property of TBQ(σ) in both policy evaluation and control. In addition, its convergence is proved for control setting. We also provide two versions of TBQ(σ) control algorithm: online forward version TBQ$^F(\sigma)$ and online backward version TBQ$^B(\sigma)$.

Comparing to TB(λ), the proposed algorithm improves the efficiency of trace utilization when target policy is greedy. Comparing to Naive Q(λ), our algorithm has relatively loose convergence requirement. Since the coefficient c in our algorithm is less than 1, the variance of our algorithm is bounded [13]. Although we are not able to give further theoretical analysis between bootstrapping parameter λ and degree of cutting traces σ on convergence, we empirically show that the existing off-policy learning algorithms with eligibility traces can be improved and accelerated by adjusting

a suitable trace-cutting degree parameter σ. The theoretical relationship between bootstrapping parameter λ and σ is remained for the future work.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is partly supported by National Key Research and Development Plan under Grant No. 2016YFB1001203, Zhejiang Provincial Natural Science Foundation of China (LR15F020001).

REFERENCES

[1] Andrew G Barto, Richard S Sutton, and Charles W Anderson. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics* 5 (1983), 834–846.

[2] Dimitry P. Bertsekas and John N. Tsitsiklis. 1996. *Neuro-Dynamic Programming.* Athena Scientific.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[4] Kristopher De Asis, J Fernando Hernandez-Garcia, G Zacharias Holland, and Richard S Sutton. 2017. Multi-step reinforcement learning: A unifying algorithm. *AAAI Conference on Artificial Intelligence* (2017).

[5] Satinder Singh Doina Precup, Richard S. Sutton. 2000. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning, 2000.* Morgan Kaufmann, 759–766.

[6] Anna Harutyunyan, Marc G Bellemare, Tom Stepleton, and Rémi Munos. 2016. Q(λ) with Off-Policy Corrections. In *International Conference on Algorithmic Learning Theory.* Springer, 305–320.

[7] H Hasselt. 2011. *Insights in reinforcement learning: formal analysis and empirical evaluation of temporal-difference learning algorithms.* Ph.D. Dissertation. Universiteit Utrecht.

[8] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[9] Jinsong Leng, Colin Fyfe, and Lakhmi C Jain. 2009. Experimental analysis on Sarsa (λ) and Q (λ) with different eligibility traces strategies. *Journal of Intelligent & Fuzzy Systems* 20, 1, 2 (2009), 73–82.

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. *International Conference on Learning Representations (ICLR)* (2016).

[11] Wenjia Meng, Qian Zheng, Long Yang, Pengfei Li, and Gang Pan. 2018. Qualitative Measurements of Policy Discrepancy for Return-based Deep Q-Network. *arXiv preprint arXiv:1806.06953* (2018).

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[13] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. 2016. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems.* 1054–1062.

[14] Jing Peng and Ronald J Williams. 1994. Incremental multi-step Q-learning. In *Machine Learning Proceedings 1994.* Elsevier, 226–232.

[15] Doina Precup. 2000. *Temporal abstraction in reinforcement learning.* Ph.D. Dissertation. University of Massachusetts Amherst.

[16] Satinder Singh and Peter Dayan. 1998. Analytical mean squared error curves for temporal difference learning. *Machine Learning* 32, 1 (1998), 5–40.

[17] Rich Sutton, Ashique Rupam Mahmood, Doina Precup, and Hado Hasselt. 2014. A new Q (lambda) with interim forward view and Monte Carlo equivalence. In *International Conference on Machine Learning.* 568–576.

[18] Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3, 1 (1988), 9–44.

[19] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction.* MIT press.

[20] Richard S Sutton and Andrew G Barto. 2011. *Reinforcement learning: An introduction.* Cambridge, MA: MIT Press.

[21] Harm Van Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering. 2009. A theoretical and empirical analysis of Expected Sarsa. In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on.* IEEE, 177–184.

[22] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[23] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards.* Ph.D. Dissertation. King's College, Cambridge.

[24] Long Yang, Minhao Shi, Qian Zheng, Wenjia Meng, and Gang Pan. 2018. A Unified Approach for Multi-step Temporal-Difference Learning with Eligibility Traces in Reinforcement Learning. *International Joint Conference on Artificial Intelligence (IJCAI)* (2018).