A Unified Approach for Multi-step Temporal-Difference Learning with Eligibility Traces in Reinforcement Learning

Long Yang, Minhao Shi, Qian Zheng, Wenjia Meng, and Gang Pan*
College of Computer Science and Technology, Zhejiang University, China
\{yanglong, minhaowill, qianzheng, mengwenjia, gpan\}@zju.edu.cn

Abstract

Recently, a new multi-step temporal learning algorithm \( Q(\sigma) \) unifies \( n \)-step Tree-Backup (when \( \sigma = 0 \)) and \( n \)-step Sarsa (when \( \sigma = 1 \)) by introducing a sampling parameter \( \sigma \). However, similar to other multi-step temporal-difference learning algorithms, \( Q(\sigma) \) needs much memory consumption and computation time. Eligibility trace is an important mechanism to transform the off-line updates into efficient on-line ones which consume less memory and computation time. In this paper, we combine the original \( Q(\sigma) \) with eligibility traces and propose a new algorithm, called \( Q^\lambda(\sigma, \lambda) \), where \( \lambda \) is trace-decay parameter. This new algorithm unifies Sarsa(\( \lambda \)) (when \( \sigma = 1 \)) and \( Q^\lambda(\lambda) \) (when \( \sigma = 0 \)). Furthermore, we give an upper error bound of \( Q^\lambda(\sigma, \lambda) \) policy evaluation algorithm. We prove that \( Q^\lambda(\sigma, \lambda) \) control algorithm converges to the optimal value function exponentially. We also empirically compare it with conventional temporal-difference learning methods. Results show that, with an intermediate value of \( \sigma \), \( Q^\lambda(\sigma, \lambda) \) creates a mixture of the existing algorithms which learn the optimal value significantly faster than the extreme end (\( \sigma = 0 \), or \( 1 \)).

1 Introduction

In reinforcement learning (RL), experiences are sequences of states, actions and rewards which are generated by the agent interacts with environment. The agent’s goal is learning from experiences and seeking an optimal policy from the delayed reward decision system. There are two fundamental mechanisms of learning have been studied in RL, one is temporal-difference (TD) learning method which is a combination of Monte Carlo method and dynamic programming [Sutton, 1988]. The other one is eligibility trace [Sutton, 1984; Watkins, 1989] which is a short-term memory process as a function of states. TD(\( \lambda \)) combines TD learning with eligibility trace and unifies one-step learning (\( \lambda = 0 \)) and Monte Carlo methods (\( \lambda = 1 \)) through the trace-decay parameter \( \lambda \) [Sutton, 1988]. Results show that the unified algorithm TD(\( \lambda \)) performs better at an intermediate value of \( \lambda \) rather than either extreme case.

Recently, Multi-step \( Q(\sigma) \) [Sutton and Barto, 2017] unifies \( n \)-step Sarsa (\( \sigma = 1 \), full-sampling) and \( n \)-step Tree-backup (\( \sigma = 0 \), pure-expectation). For some intermediate value \( \sigma(0 < \sigma < 1) \), \( Q(\sigma) \) creates a mixture of full-sampling and pure-expectation approach, can perform better than the extreme case \( \sigma = 0 \) or 1 [De Asis et al., 2018].

The results in [De Asis et al., 2018] implies a fundamental trade-off problem in reinforcement learning: should one estimates the value function by adopting pure-expectation (\( \sigma = 0 \)) algorithm or full-sampling (\( \sigma = 1 \)) algorithm? Although pure-expectation approach has lower variance, it needs more complexity and larger calculation [Van Seijen et al., 2009]. On the other hand, full-sampling algorithm needs smaller calculation time, however, it may have a worse asymptotic performance [De Asis et al., 2018]. Multi-step \( Q(\sigma) \) [Sutton and Barto, 2017] firstly attempts to combine pure-expectation with full-sample algorithms, however, multi-step temporal-difference learning is too expensive in complexity of time and space during its training.

Eligibility trace is an effective way to address the above complexity problem. In this paper, we try to combine the \( Q(\sigma) \) algorithm with eligibility trace, and create a new algorithm, called \( Q^\lambda(\sigma, \lambda) \). The proposed \( Q^\lambda(\sigma, \lambda) \) unifies the Sarsa(\( \lambda \)) algorithm [Rummery and Niranjan, 1994] and \( Q^\lambda(\lambda) \) algorithm [Harutyunyan, 2016]. When \( \sigma \) varies from 0 to 1, \( Q^\lambda(\sigma, \lambda) \) changes continuously from Sarsa(\( \lambda \)) (\( \sigma = 1 \) in \( Q^\lambda(\sigma, \lambda) \)) to \( Q^\lambda(\lambda) \) (\( \sigma = 0 \) in \( Q^\lambda(\sigma, \lambda) \)). We also focus on the trade-off between pure-expectation and full-sample in control task, our experiments show that an intermediate value \( \sigma \) achieves a better performance than either extreme case.

Our contributions are summarised as follows: We define a mixed-sampling operator through which we derive the proposed \( Q^\lambda(\sigma, \lambda) \) policy evaluation algorithm and control algorithm. The new algorithm \( Q^\lambda(\sigma, \lambda) \) unifies Sarsa(\( \lambda \)) and \( Q^\lambda(\lambda) \). For \( Q^\lambda(\sigma, \lambda) \) policy evaluation algorithm, we give its upper error bound. For the control problem, we prove that both of the off-line and on-line \( Q^\lambda(\sigma, \lambda) \) algorithm converge to the optimal value function.

2 Framework and Notation

The standard episodic reinforcement learning framework [Sutton and Barto, 2017] is often formalized as Markov deci-
sion processes (MDPs). Such framework considers 5-tuples form \( M = (S, A, P, R, \gamma) \), where \( S \) indicates the set of all states, \( A \) indicates the set of all actions, \( P^a_s \) is the element of \( P \) and it indicates a state-transition probability from state \( s \) to state \( s' \) under taking action \( a \), \( a \in A \), \( s, s' \in S \); \( R^a_s \) is the element of \( R \) and it indicates the expected reward for a transition, \( \gamma \in [0, 1] \) is the discount factor. A policy \( \pi \) is a probability distribution on \( S \times A \) and stationary policy is a policy that does not change over time. In this paper, we denote \( \{(S_t, A_t, R_t)\}_{t\geq 0} \) as a trajectory of the state-reward sequence in one episode.

State-action value \( q \) maps on \( S \times A \) to \( \mathbb{R} \), for a given policy \( \pi \), has a corresponding state-action value:

\[
q^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s, A_0 = a \right].
\]

Optimal state-action value is defined as:

\[
q^*(s, a) = \max_{\pi} q^\pi(s, a).
\]

Bellman operator \( T^\pi \):

\[
T^\pi q = R^\pi + \gamma P^\pi q,
\]

Bellman optimal operator \( T^* \):

\[
T^* q = \max_{\pi} (R^\pi + \gamma P^\pi q),
\]

where \( \mathcal{P} \in \mathbb{R}^{|S| \times |S|} \), \( \mathcal{R} \in \mathbb{R}^{|S| \times |A|} \), and the corresponding entry is:

\[
\mathcal{P}_{s's} = \sum_{a \in A} \mathcal{P}(s, a) P^a_s, \quad \mathcal{R}(s, a) = \mathcal{P}(s, a) \sum_{s' \in S} P^a_s R^a_{s'}. \]

Value function \( q^\pi \) and \( q^* \) satisfy the following Bellman equation and optimal Bellman equation correspondingly:

\[
T^\pi q^\pi = q^\pi, \quad T^* q^* = q^*.
\]

Both \( T^\pi \) and \( T^* \) are \( \gamma \)-contraction operator with sup-norm, that is to say, \( \| T Q_1 - T Q_2 \|_{\infty} \leq \gamma \| Q_1 - Q_2 \|_{\infty} \) for any \( Q_1, Q_2 \in \mathbb{R}^{|S| \times |A|} \), where \( T = T^\pi \) or \( T^* \). Based on the fact that the fixed point of a contraction operator is unique, then value iteration converges: \( (T^\pi)^n Q \to q^\pi \), \( (T^*)^n Q \to q^* \), as \( n \to \infty \), for any initial \( Q \).

### 2.1 One-step TD Learning Algorithms

Unfortunately, both the system (1) and (2) can not be solved directly because of the fact that the \( \mathcal{P} \) and \( \mathcal{R} \) in the environment are usually unknown or the dimension of \( S \) is huge in practice. An unavailable model in reinforcement learning is called model-free.

TD learning algorithm [Sutton, 1984; Sutton, 1988] is one of the most significant algorithms in model free reinforcement learning, the idea of bootstrapping is critical to TD learning: the evaluation of the value function are used as targets during the learning process.

In reinforcement learning, target policy \( \mu \) is the policy being learned about, and the policy \( \mu \) used to generate trajectory \( \{(S_t, A_t, R_t)\}_{t\geq 0} \) is called the behavior policy. If \( \pi = \mu \), the learning is called on-policy learning, otherwise it is off-policy learning.

**Sarsa:** For a given sample transition \((S, A, R, S', A')\), Sarsa [Rummery and Niranjan, 1994] is a on-policy learning algorithm and it updates \( Q \) value as follows:

\[
Q_{k+1}(S, A) = Q_k(S, A) + \alpha_k \delta^S_k,
\]

\[
\delta^S_k = R + \gamma Q_k(S', A') - Q_k(S, A),
\]

where \( \delta^S_k \) is the \( k \)-th TD error, \( \alpha_k \) is stepsize.

**Expected-Sarsa:** Expected-Sarsa [Van Seijen et al., 2009] uses expectation of all the next state-action value pairs according to the target policy \( \pi \) to estimate \( Q \) value as follows:

\[
Q_{k+1}(S, A) = Q_k(S, A) + \alpha_k \delta^{ES}_k,
\]

\[
\delta^{ES}_k = R + \gamma \mathbb{E}_{\pi}[Q_k(S', \cdot)] - Q_k(S, A),
\]

where \( \delta^{ES}_k \) is the \( k \)-th expected TD error. Expected-Sarsa is an off-policy learning algorithm if \( \mu \neq \pi \), for example, when \( \pi \) is greedy with respect to \( Q \) then Expected-Sarsa is restricted to Q-Learning [Watkins, 1989]. If the trajectory is generated by \( \pi \), Expected-Sarsa is a on-policy algorithm [Van Seijen et al., 2009].

The above two algorithms are guaranteed convergence under some conditions [Singh et al., 2000; Van Seijen et al., 2009].

\( \mathbb{Q}(\sigma) \): One-step \( \mathbb{Q}(\sigma) \) [Sutton and Barto, 2017; De Asis et al., 2018] is a weighted average between the Sarsa update and Expected Sarsa update through a sampling parameter \( \sigma \):

\[
Q_{k+1}(S, A) = Q_k(S, A) + \alpha_k \delta^S_k \sigma,
\]

\[
\delta^S_k = R + \gamma Q_k(S', A') - Q_k(S, A),
\]

\[
\delta^{ES}_k = R + \gamma \mathbb{E}_{\pi}[Q_k(S', \cdot)] - Q_k(S, A),
\]

\[
(1 - \sigma) \delta^{ES}_k,
\]

where \( \sigma \in [0, 1] \) is degree of sampling. When \( \sigma = 1 \), \( Q(\sigma) \) denotes a full-sampling, and \( \sigma = 0 \), \( Q(\sigma) \) denotes a pure-expectation with no sampling. \( \delta^S_k \) and \( \delta^{ES}_k \) are defined in (4) and (5).

### 2.2 \( \lambda \)-Return Algorithm

One-step TD learning algorithm can be generalized to multi-step bootstrapping learning method. The \( \lambda \)-return algorithm [Watkins, 1989] is a particular way to mix many multi-step TD learning algorithms through weighting \( \nu \)-step returns proportionally to \( \lambda^{\nu-1} \).

\( \lambda \)-operator\(^1 \) \( T^\lambda_\pi \) is a flexible way to express \( \lambda \)-return algorithm, for a given trajectory \( \{(S_t, A_t, R_t)\}_{t\geq 0} \) that is generated by policy \( \pi \),

\[
(T^\lambda_\pi q)(s, a) = \left\{ (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n (T^\pi)^{n+1} q \right\}(s, a)
\]

\[
= \sum_{n=0}^{\infty} (1 - \lambda) \lambda^n \mathbb{E}_\pi[G_n | S_0 = s, A_0 = a]
\]

\[
= q(s, a) + \sum_{n=0}^{\infty} (\lambda)^n \mathbb{E}_\pi[\delta_n],
\]

\(^1\)The notation is coincident with [Bertsekas et al., 2012].
where \( G_n = \sum_{t=0}^{\infty} \gamma^t R_t + \gamma^n Q(S_{n+1}, A_{n+1}) \) is \( n \)-step returns from initial state-action pair \((S_0, A_0)\), the term \( \sum_{n=0}^{\infty} (1 - \lambda)^n G_n \) is called \( \lambda \)-returns, and \( \delta_n = R_n + \gamma Q(S_{n+1}, A_{n+1}) - Q(S_n, A_n) \).

Based on the fact that \( q^\pi \) is fixed point of \( T^\pi \), thus \( q^\pi \) remains the fixed point of \( T^\lambda_{\pi} \). When \( \lambda = 0 \), \( T^\lambda_{\pi} \) is equal to the usual Bellman operator \( T^\pi \). When \( \lambda = 1 \), the evaluation iteration \( Q_{k+1} = T^\lambda_{\pi} Q_k \) becomes Monte Carlo method. It is well-known that \( \lambda \) trades off the bias of the bootstraping with an approximate \( q^\pi \), with the variance of sampling multi-step returns estimation [Kearns and Singh, 2000]. In practice, a high and intermediate value of \( \lambda \) could be typically better [Singh and Dayan, 1998; Sutton, 1996].

### 3 Mixed-sampling Operator

In this section, we present the mixed-sampling operator \( T^\pi_{\sigma, \mu} \), which is one of our key contribution and is flexible to analysis our new algorithm later. By introducing a sampling parameter \( \sigma \in [0, 1] \), the mixed-sampling operator varies continuously from pure-expectation method to full-sampling method. In this section, we analysis the contraction of \( T^\pi_{\sigma, \mu} \) firstly. Then we introduce the \( \lambda \)-return vision of mixed-sampling operator \( T^\lambda_{\sigma, \mu} \). Finally, we give a upper bound error of the corresponding policy evaluation algorithm.

#### 3.1 Contraction of Mixed-sampling Operator

**Definition 1.** Mixed-sampling operator \( T^\pi_{\sigma, \mu} \) is a map on \( \mathbb{R}^{S \times A} \to \mathbb{R}^{S \times A} \), \( \forall S \in S, A \in A, \sigma \in [0, 1] \):

\[
T^\pi_{\sigma, \mu} : \mathbb{R}^{S \times A} \to \mathbb{R}^{S \times A}
q(s, a) \mapsto q(s, a) + \mathbb{E}_a \sum_{t=0}^{\infty} [\gamma^t \delta_t^{\pi, \sigma}]
\tag{7}
\]

where

\[
\delta_t^{\pi, \sigma} = \sigma \left( R_t + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right) + (1 - \sigma) \left( R_t + \gamma \mathbb{E}_a Q(S_{t+1}, \cdot) - Q(S_t, A_t) \right)
\]

\[
\mathbb{E}_a Q(S_{t+1}, \cdot) = \sum_{a \in A} \pi(S_{t+1}, a) Q(S_{t+1}, a).
\]

The parameter \( \sigma \) is also degree of sampling that is introduced by the \( Q(\sigma) \) algorithm [De Asis et al., 2018]. In one of extreme end (\( \sigma = 0 \), pure-expectation), \( T^\pi_{\sigma=0} \) deduces the \( n \)-step returns \( G_n^\pi \) in \( Q^\pi(\lambda) \) [Harutyunyan, 2016], where \( G_n^\pi = \sum_{k=n}^{\infty} \lambda^{k-n} \gamma^k E_q + \gamma^{n+1} \mathbb{E}_a Q(S_{n+1}, \cdot) \), \( \delta^E_k \) is the \( k \)-th expected TD error. Multi-step Sarsa [Sutton and Barto, 2017] is in another extreme end (\( \sigma = 1 \), full-sampling). Every intermediate value \( \sigma \) creates a mixed method varies continuously from pure-expectation to full-sampling which is why we call \( T^\pi_{\sigma, \mu} \) mixed-sample operator.

#### 3.2 Upper Error Bound of Policy Evaluation

In this section we discuss the ability of policy evaluation iteration \( Q_{k+1} = T^\pi_{\sigma, \mu} Q_k \) in Theorem 1. Our results show that when \( \mu \) and \( \pi \) are sufficiently close, the ability of the policy evaluation iteration increases gradually as the \( \sigma \) decreases from 1 to 0.

**Lemma 1.** If a positive sequence \( \{a_k\}_{k=1}^\infty \) satisfies \( a_{k+1} \leq \alpha a_k + \beta \), then for any \( |a| < 1 \), we have

\[
a_k - \frac{\beta}{1 - \alpha} \leq a_1 - \frac{\beta}{1 - \alpha}.
\]

Furthermore, for any \( \epsilon > 0, \exists K, s.t. \forall k > K \) has the following estimation

\[
|a_k| \leq \frac{\beta}{1 - \alpha} + \epsilon.
\]

**Theorem 2** (Upper error bound of policy evaluation). Consider the policy evaluation algorithm \( Q_{k+1} = T^\pi_{\sigma, \mu} Q_k \), if the behavior policy \( \mu \) is \( \epsilon \)-away from the target policy \( \pi \), in the sense that \( \max_{s \in S} ||\pi(s, a) - \mu(s, a)||_1 \leq \epsilon, \epsilon < \frac{1 - \gamma}{1 + 2\lambda} \), and \( \gamma(1 + 2\lambda) < 1 \), then \( \exists K, s.t. \forall k > K \), the policy evaluation sequence \( \{Q_k\}_{k=0}^\infty \) satisfies

\[
||Q_{k+1} - q^\pi|| \leq \epsilon \left[ \frac{M + \gamma C_\pi}{1 - \gamma(1 + 2\lambda)} + 1 \right],
\]

where for a given policy \( \pi \), \( M, C_\pi \) is determined by the learning system.
Proof. Firstly, we provide an equation which could be used later:

\[
q + B[T^\mu q - q] - q^n
\]

In this section, we present the off-line version of the policy evaluation iteration is dependent on the gap \(\gamma\) between \(\pi\) and \(\pi^\mu\):

\[
\begin{align*}
Q_{k+1} &= \pi^\mu Q_k + (1-\sigma)T^\mu_{\pi^\mu} Q_k.
\end{align*}
\]

Note \(q^n\) is fixed point of \(T^\mu_{\pi^\mu}\) [Harutyunyan, 2016, then we merely consider next estimator:

\[
\begin{align*}
Q_{k+1} - q^n &= \sigma Q_k + B[T^\mu Q_k - Q_k] - q^n
\end{align*}
\]

\[
\begin{align*}
&\leq \sigma \left[ (M + \gamma ||q^n||) + \gamma (1 + \lambda) \right]
\end{align*}
\]

Lemma 1:

\[
\begin{align*}
\sigma \epsilon \left[ \frac{M + \gamma C_\pi}{1 - \gamma (1 + 2 \lambda)} \right] + \epsilon M_s,
\end{align*}
\]

where \(M_s = |A| \max_a |R^*_a|\) is determined by the reinforcement learning system and is independent of \(\pi, \mu, M = \max_s \mu S M_s\). For the given policy \(\pi, ||q^n||\) is a constant that is determined by learning system, we denote it \(C_\pi\).

Remark 1: The proof in Theorem 2 strictly dependent on the assumption that \(\epsilon\) is smaller but never to be zero, where the \(\epsilon\) is a bound of discrepancy between the behavior policy \(\mu\) and target policy \(\pi\). That is to say, the ability of the prediction of policy evaluation iteration is dependent on the gap between \(\mu\) and \(\pi\). A more profound discussion about this gap is related to \(\lambda\) trade off [Harutyunyan, 2016] which is beyond the scope in this paper.

### 4 \(Q^\pi(\sigma, \lambda)\) Control Algorithm

In this section, we present the off-line version of \(Q^\pi(\sigma, \lambda)\) which converges to optimal value function exponentially fast.

Considering the typical iteration \((Q_k, \pi_k)\), where \(\pi_k\) is calculated by the following two steps:

Step 1: policy evaluation

\[
Q_{k+1} = T^\pi_{\sigma, \lambda} Q_k,
\]

Step 2: policy improvement

\[
T^\pi_{\sigma, \lambda} Q_{k+1} = T^\pi Q_{k+1},
\]

that is \(\pi_{k+1}\) is greedy policy with respect to \(Q_{k+1}\). We call the approach introduced by above step 1 and step 2 \(Q^\pi(\sigma, \lambda)\) control algorithm.

In the following Theorem 3, we presents the convergence rate of \(Q^\pi(\sigma, \lambda)\) control algorithm.

Theorem 3 (Convergence of \(Q^\pi(\sigma, \lambda)\) Control Algorithm):

Considering the sequence \(\{(Q_k, \pi_k)\}_{k\geq 0}\) generated by the \(Q^\pi(\sigma, \lambda)\) control algorithm, for the given \(\lambda, \gamma \in (0, 1)\), then

\[
\|Q_k - q^*\|_\infty \leq \eta \|Q_k - q^*\|_\infty,
\]

where \(\eta = \frac{\gamma (1 + \lambda - 2 \lambda \gamma)}{1 - 2 \lambda \gamma}\).

Particularly, if \(\lambda < \frac{1 - \gamma}{2 \gamma}\), then \(0 < \eta < 1\) and sequence \(\{Q_k\}_{k\geq 1}\) converges to \(q^*\) exponentially fast:

\[
\|Q_k - q^*\|_\infty = O(\eta)^{k+1}.
\]

Proof. By the definition of \(T^\mu_{\sigma, \lambda}\),

\[
\begin{align*}
\|T^\mu_{\sigma, \lambda} q - T^\mu q\|_\infty &= \|\sigma T^\mu q + (1-\sigma)T^\mu_{\pi^\mu} q - T^\mu q\|_\infty
\end{align*}
\]

we have:

\[
\|Q_k - q^*\|_\infty \leq \|T^\mu_{\sigma, \lambda} (Q_k - q^*)\|_\infty + (1 - \sigma)\|T^\mu_{\pi^\mu} (Q_k - q^*)\|_\infty
\]

\[
\leq \left( \frac{\sigma (1 - \lambda \gamma)}{1 - \lambda \gamma} + (1 - \sigma) \frac{\gamma (1 + \lambda)}{1 - \lambda \gamma} \right) \|Q_k - q^*\|_\infty
\]

\[
= \frac{\gamma (1 + \lambda - 2 \lambda \gamma)}{1 - 2 \lambda \gamma} \|Q_k - q^*\|_\infty
\]

Algorithm 1: Online \(Q^\pi(\sigma, \lambda)\) algorithm

**Require**: Initialize \(Q_0(s, a)\) arbitrarily, \(\forall s \in S, \forall a \in A\)

**Require**: Initialize \(\mu_k\) to be the behavior policy

**Parameters**: step-size \(\alpha_t \in (0, 1]\)

**Repeat** (for each episode):

- **Solve** a state-action pair \((S_t, A_t)\) by \(\mu_k\)

- \(\delta_t^{\pi, \mu} = R_t + \gamma \{ (1 - \sigma) E_{\pi_k} [Q_{k+1}(S_{t+1}, \cdot)] + \sigma Q_{k+1}(S_{t+1}, A_{t+1}) \} - Q_{k+1}(S_t, A_t)\)

For \(\forall s \in S, \forall a \in A\):

- \(Z(s, a) = \gamma Z(s, a) + \{ (1 - \sigma) E_{\pi_k} [Q_{k+1}(S_{t+1}, \cdot)] + \sigma Q_{k+1}(S_{t+1}, A_{t+1}) \} - Q_{k+1}(S_t, A_t)\)

**End For**

**If** \(S_{t+1}\) is terminal:

**Break**

**End For**
5 On-line Implementation of $Q^\pi(\sigma, \lambda)$

We have discussed the contraction property of mixed-sampling operator $T_{\sigma, \lambda}^\mu$, and via it we have introduced the $Q^\pi(\sigma, \lambda)$ control algorithm. Both of the iterations in Theorem 2 and Theorem 3 are the version of offline. In this section, we give the on-line version of $Q^\pi(\sigma, \lambda)$ and discuss its convergence.

5.1 On-line Learning

Off-line learning is too expensive due to the learning process must be carried out at the end of an episode, however, on-line learning updates value function with a lower computational cost, better performance. There is a simple interpretation of equivalence between off-line learning and on-line learning which means that, by the end of an episode, the total updates of the forward view (off-line learning) is equal to the total updates of the backward view (on-line learning) [Sutton and Barto, 1998]. By the view of equivalence, on-line learning can be seen as an implementation of off-line algorithm in an inexpensive manner. Another interpretation of online learning was provided by [Singh and Sutton, 1996], TD learning with accumulate trace comes to approximate every-visit Monte-Carlo method and TD learning with replace trace comes to approximate first-visit Monte-Carlo method.

The iterations in Theorem 2 and Theorem 3 are the version of expectations. In practice, we can only access to the trajectory $\{(S_t, A_t, R_t)\}_{t \geq 0}$. By statistical approaches, we can utilize the trajectory to estimate the value function. Algorithm 1 corresponds to online form of $Q^\pi(\sigma, \lambda)$.

5.2 On-line Learning Convergence Analysis

We make some common assumptions which are similar to [Bertsekas and Tsitsiklis, 1996; Harutyunyan, 2016].

Assumption 1. $\sum_{t \geq 0} P_t[(S_t, A_t) = (s, a)] > 0$, minimum visit frequency, which implies a complete exploration.

Assumption 2. For every historical chain $F_t$ in a MDPs, $P_t[N_t(s, a) \geq k | F_t] \leq \gamma^k$, where $\rho$ is a positive constants, $k$ is a positive integer and $N_t(s, a)$ denotes the times of the pair $(s, a)$ is visited in the $t$-th trajectory $F_t$.

For the convenience of expression, we give some notations first. Let $Q_{k,t}^\rho$ be the vector obtained after $t$ iterations in the $k$-th trajectory, and the superscript $\rho$ emphasizes online learning. We denote the $k$-th trajectory as $\{(S_t, A_t, R_t)\}_{t \geq 0}$ sampled by the policy $\mu_k$. Then the online update rules can be expressed as follows: $\forall (s, a) \in S \times A$:

$$Q_{k,0}^\rho(s, a) = Q_k^\rho(s, a)$$

$$\delta_{k,t}^\rho = R_t + \gamma \left\{ (1-\sigma)\mathbb{E}_\pi_k[Q_{k,t+1}(S_{t+1}, \cdot)] \right\} + \sigma Q_{k,t}(S_t, A_t) - Q_{k,t}^\rho(s, a)$$

$$Q_{k,t+1}^\rho(s, a) = Q_{k,t}^\rho(s, a) + \alpha_k(s, a)z_{k,t}^\rho(s, a)\delta_{k,t}$$

where $T_k$ is the length of the $k$-th trajectory.

Theorem 4. Based on the Assumption 1 and Assumption 2, step-size $\alpha_k$ satisfying, $\sum t \alpha_t = \infty, \sum t^2 \alpha_t^2 < \infty$, $\pi_k$ is greedy with respect to $Q_k^\rho$, then $Q_k^\rho \xrightarrow{w.p.1} q^*$ as $k \to \infty$, where w.p.1 is short for with probability one.

Proof. After some simple algebra:

$$Q_{k+1}(s, a) = Q_k^\rho(s, a) + \alpha_k(G_k^\rho(s, a) - \frac{N_k(s, a)}{\mathbb{E}[N_k(s, a)]}Q_k^\rho(s, a)),$$

$$G_k^\rho(s, a) = \frac{1}{\mathbb{E}[N_k(s, a)]} \sum_{t=0}^{T_k} z_{k,t}^\rho(s, a)\delta_{k,t}^\rho,$$

where $\alpha_k = \mathbb{E}[N_k(s, a)]q_k(s, a)$.

We rewrite the off-line update:

$$Q_{k+1}^f(s, a) = Q_k^f(s, a) + \alpha_k(G_k^f(s, a) - \frac{N_k(s, a)}{\mathbb{E}[N_k(s, a)]}Q_k^f(s, a)),$$

$$G_k^f = \frac{1}{\mathbb{E}[N_k(s, a)]} \sum_{t=0}^{T_k} Q_{k,t}^f(s, a),$$

where $Q_{k,t}^f(s, a)$ is the $\lambda$-returns at time $t$ when the pair $(s, a)$ is visited in the $k$-th trajectory, the superscript $f$ in $Q_{k+1}^f(s, a)$ emphasizes the forward (off-line) update. $N_k(s, a)$ denotes the times of the pair $(s, a)$ visited in the $k$-th trajectory. We define $Res_k(s, a)$ as the residual between on-line estimate $G_k^\rho$ and off-line estimate $G_k^f(s, a)$ in the $k$-th trajectory:

$$Res_k(s, a) = Q_k^\rho(s, a) - Q_k^f(s, a).$$

Set $\Delta_k(s, a) = Q_k^\rho(s, a) - q^*(s, a)$ which is the error between after $k$-episode the learned value function $Q_k^\rho$ and optimal value function $q^*$, then we consider the next random iterative process:

$$\Delta_{k+1}(s, a) = (1 - \alpha_k(s, a))\Delta_k(s, a) + \beta_k F_k(s, a),$$

where

$$\alpha_k(s, a) = \frac{N_k(s, a)q_k(s, a)}{\mathbb{E}_\mu_k[N_k(s, a)]}, \beta_k(s, a) = \alpha_k(s, a),$$

$$F_k(s, a) = G_k^\rho(s, a) - \frac{N_k(s, a)}{\mathbb{E}_\mu_k[N_k(s, a)]}q^*(s, a).$$

Step1: Upper bound on $Res_k(s, a)$:

$$\max_{(s, a)} \mathbb{E}_{\mu_k}[Res_k(s, a)] \leq C_k \max_{(s, a)} \left| Q_{k+1}^\rho(s, a) - q^* \right|, \quad (12)$$

where $C_k \xrightarrow{w.p.1} 0$.

$$Res_{k,t}(s, a) = \frac{1}{\mathbb{E}[N_k(s, a)]} \sum_{m=0}^{t} z_{k,m}^\rho(s, a)\delta_{k,m}^\rho - Q_{k,m}^\lambda(s, a).$$
where \(0 \leq t \leq T_k\). \(Res_{k,t}(s,a)\) is the difference between the total on-line updates of first \(t\) steps and the first \(t\) times off-line update in \(k\)-th trajectory. By induction on \(t\), we have:
\[
\|Res_{k,t+1}(s,a)\| \leq \alpha_M C(\Delta_{k,t} + \|Res_{k,t}(s,a)\|),
\]
where \(C\) is a consist and \(\Delta_{k,t} = \|Q^*_k(s,a) - q^*(s,a)\|\), \(\alpha_M = \max_{0 \leq t \leq T_k} \{\alpha_t\}\). Based on the condition of step-size in the Theorem 4, \(\alpha_M \xrightarrow{w,p} 0\), then we have (12).

**Step 2:** \(\max_{(s,a)} E_{\mu_k}[F_k(s,a)] \leq \gamma \max_{(s,a)} \|\Delta_k(s,a)\|\).

In fact:
\[
F_k(s,a) = G_k(s,a) + Res_k(s,a)
\]
\[
E_{\mu_k}[F_k(s,a)] = G_k(s,a) - \frac{N_k(s,a)}{E_{\mu_k}[N_k(s,a)]} q^*(s,a),
\]
\[
\sum_{i=0}^{N_k(s,a)} E_{\mu_k}[Q_k(s,a) - q^*] + Res_k(s,a).
\]

From the property of eligibility trace (more details refer to [Bertsekas et al., 2012]) and Assumption 2, we have:
\[
\left|E_{\mu_k}[Q_{k,t}(s,a) - q^*]\right| \\
\leq P[N_k(s,a) \geq t]\left|E_{\mu_k}[Q_k^t(s,a) - q^*]\right|
\]
\[
\leq \gamma \rho^t \max_{(s,a)} \|Q_k^t(s,a) - q^*\|.
\]

Then according to (11), for some \(t > 0\):
\[
E_{\mu_k}[F_k(s,a)] \\
\leq (\gamma \rho^t + \alpha_t) \max_{(s,a)} \|\Delta_k(s,a)\|
\leq \gamma \max_{(s,a)} \|\Delta_k(s,a)\|.
\]

**Step 3:** \(Q_k^t \xrightarrow{w,p} q^*\) as \(k \to \infty\).

Considering the iteration (11) and Theorem 1 in [Jaakkola et al., 1994], then we have \(Q_k^t \xrightarrow{w,p} q^*\).

\(\square\)

6 **Experiments**

6.1 **Experiment for Prediction Capability**

In this section, we test the prediction abilities of \(Q^\pi(s,\lambda)\) in 19-state random walk environment [De Asis et al., 2018]. The agent at each state has two action: \(left\) and \(right\), and taking each action with equal probability. We compare the root-mean-square (RMS) error as a function of episodes, \(\sigma\) varies dynamically \(\sigma\) from 0 to 1 with steps of 0.2. In this experiment, we set behavior policy \(\mu = 0.5 - D, 0.5 + D\), \(D \sim U[0.01, 0.01]\), where \(U\) is uniform distribution.

Results in Figure 1 show that the performance of \(Q^\pi(s,\lambda)\) increases gradually as the \(\sigma\) decreases from 1 to 0, which just verifies the upper bound error in Theorem 2.

6.2 **Experiment for Control Capability**

We test the control capability of \(Q^\pi(s,\lambda)\) in the classical episodic task, mountain car [Sutton and Barto, 1998], and use the version 3 of Sutton’s tile coding software (n.d.) with 8 tilings.

In the right part of Figure 2, we collect the data by varying \(\sigma\) from 0 to 1 with steps of 0.02. Results show that \(Q^\pi(s,\lambda)\) significantly converges faster than \(Q(\sigma)\). In the left part of Figure 2, results show that the \(Q^\pi(s,\lambda)\) with \(\sigma\) in an intermediate value can outperform \(Q^\pi(\lambda)\) and Sarsa(\(\lambda\)) (\(\sigma = 1\)). \(\gamma = 0.99\), step-size \(\alpha = 0.3\).

Table 1 and Table 2 summarize the average return after 50 and 200 episodes. In order to gain more insight into the nature of the results, we run \(\sigma\) from 0 to 1 with steps of 0.02, we take the statistical method according to [De Asis et al., 2018], lower (LB) and upper (UB) 95% confidence interval bounds are provided to validate the results. Dynamic \(\sigma\) in the table mean that \(\sigma\) does not need to remain constant throughout every episode [De Asis et al., 2018]. In this experiment, \(\sigma\) is a function of time, a normal distribution, \(\sigma_t \sim N(0.5, 0.1^2)\). The average return after only 50 episodes could be interpreted as a measure of initial performance, whereas the average return after 200 episodes shows how well an algorithm is capable of learning [De Asis et al., 2018]. Results show that \(Q^\pi(s,\lambda)\) with an intermediate value had the best final performance.

![Figure 1: Root-mean-square (RMS) error of state values as a function of episodes in 19-state random walk, we consider both accumulating trace and replacing trace. The plot shows the prediction ability of \(Q(\sigma, \lambda)\) varying \(\sigma\), when \(\lambda\) is fixed to 0.8, \(\gamma = 1\).](image1)

![Figure 2: The plot shows the the average returns per episode. A right-centered moving average with a window of 20 successive episodes was employed in order to smooth the results. The right plot shows the result of \(Q^\pi(\sigma, \lambda)\) comparing with \(Q(\sigma)\). The left plot shows the result of \(Q^\pi(\sigma, \lambda)\), \(Q^\pi(\sigma)\) (\(\sigma = 0\)) and Sarsa(\(\lambda\)) (\(\sigma = 1\)). \(\gamma = 0.99\), step-size \(\alpha = 0.3\).](image2)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mean</th>
<th>UB</th>
<th>LB</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Q^\pi(\sigma = 0, \lambda)), (Q^\pi(\lambda))</td>
<td>-193.56</td>
<td>-179.33</td>
<td>-197.06</td>
</tr>
<tr>
<td>(Q^\pi(\sigma = 1, \lambda)), Sarsa((\lambda))</td>
<td>-196.84</td>
<td>-182.69</td>
<td>-200.42</td>
</tr>
<tr>
<td>(Q^\pi(\sigma = 0.5, \lambda))</td>
<td>-195.99</td>
<td>-181.60</td>
<td>-199.62</td>
</tr>
<tr>
<td>Dynamic (\sigma)</td>
<td>-193.01</td>
<td>-177.14</td>
<td>-195.01</td>
</tr>
</tbody>
</table>

**Table 1:** Average return per episode after 50 episodes.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mean</th>
<th>UB</th>
<th>LB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q^<em>(\sigma = 0, \lambda), Q^</em>(\lambda)$</td>
<td>-144.04</td>
<td>-138.49</td>
<td>-146.44</td>
</tr>
<tr>
<td>$Q^*(\sigma = 1, \lambda), Sarsa(\lambda)$</td>
<td>-145.72</td>
<td>-140.04</td>
<td>-148.21</td>
</tr>
<tr>
<td>$Q^*(\sigma = 0.5, \lambda)$</td>
<td>-143.71</td>
<td>-137.92</td>
<td>-146.12</td>
</tr>
<tr>
<td>Dynamic $\sigma$</td>
<td>-142.62</td>
<td>-137.24</td>
<td>-145.09</td>
</tr>
</tbody>
</table>

Table 2: Average return per episode after 200 episodes.

7 Conclusion

In this paper we presented a new method, called $Q^*(\sigma, \lambda)$, which unifies Sarsa($\lambda$) and $Q^*(\lambda)$. We solved a upper error bound of $Q^*(\sigma, \lambda)$ for the ability of policy evaluation. Furthermore, we discussed the convergence of $Q^*(\sigma, \lambda)$ control algorithm to $q^*$ under some conditions. The proposed approach was compared with one-step and multi-step TD learning methods and the results demonstrated its effectiveness.

Acknowledgements

This work was partly supported by National Key Research and Development Program of China (2017YFB1002503), Natural Science Foundation of China (No.61772460), and Zhejiang Provincial Natural Science Foundation of China (LR15F020001).

References


