Target Transfer Q-Learning and Its Convergence Analysis

Yue Wang[a], Yuting Liu[a,*], Wei Chen[b], Zhi-Ming Ma[c], Tie-Yan Liu[b]

[a] *School of Science, Beijing Jiaotong University*
[b] *Microsoft Research*
[c] *Academy of Mathematics and Systems Science, Chinese Academy of Sciences*

Abstract

Reinforcement Learning (RL) technologies are powerful to learn how to interact with environments and have been successfully applied to various important applications. Q-learning is one of the most popular methods in RL, which leverages the Bellman equation to update the Q-function. Considering that data collection in RL is both time and cost consuming and Q-learning converges slowly, different kinds of transfer RL algorithms are designed to improve the sample complexity of the new tasks[1]. However, most of the previous transfer RL algorithms are similar to the transfer learning methods in deep learning and are heuristic with no theoretical guarantee of the convergence rate. Therefore, it is important for us to clearly understand how and when will transfer learning help RL method and provide the theoretical guarantee for the improvement of the sample complexity. In this paper, we rethink the transfer Rl problems in the RL perspective and propose to transfer the Q-function learned in the old task to the target Q-function in the Q-learning of the new task. We call this new transfer Q-learning method *target transfer Q-Learning* (abbrev. TTQL). The transfer process is controlled by the error condition which can help to avoid the harm to the new tasks brought by the transferred target. We design the error condition in TTQL as whether the Bellman error of the transferred target Q-function is less than the current Q-function. We show that TTQL with the error condition will achieve a faster convergence rate than Q-learning. Our experiments are consistent with our theoretical results and verify the effectiveness of our proposed target transfer Q-learning method.

Keywords: Reinforcement Learning, Transfer Learning, Q-learning, Convergence Analysis

1. Introduction

Reinforcement Learning (RL) [1] technologies are powerful to learn how to interact with environments and have been successfully applied to variants of important applications, such as robotics, computer games and so on [2, 3, 4, 5].

Q-learning [6] is one of the most popular RL algorithms that uses temporal difference method to update the Q-function. To be specific, Q-learning uses the Bellman equation to update the Q-function where the temporal difference is calculated using the current Q-function and the Q-function obtained by Bellman operator. Since Bellman operator is a contractive mapping, Q-learning will converge to the optimal Q-function [7]. Comparing to supervised learning algorithms, Q-learning converges much slower due to interactions with the environment. At the same time, the data collection is both very time and cost consuming in RL. Thus, it is crucial for us to utilize available information to save the sample complexity of Q-Learning.

Transfer learning aims to improve the learning performance on a new task by using knowledge/model learned from old tasks. Transfer learning has a long history in supervised learning [8, 9, 10]. Recently, by leveraging the experiences from supervised transfer learning, researchers developed different kinds of transfer learning methods for RL, which can be categorized into three classes: (1) *instance transfer* in which old data will be reused in the new task

[1] In order to avoid confusion, we use "old/new tasks" instead of "source/target tasks" in this paper.

* corresponding author

Email addresses: 11271012@bjtu.edu.cn (Yue Wang), ytliu@bjtu.edu.cn (Yuting Liu), wche@microsoft.com (Wei Chen), mazm@amt.ac.cn (Zhi-Ming Ma), Tie-Yan.Liu@microsoft.com (Tie-Yan Liu)

[11, 12]; (2) *representation transfer* such as reward shaping and basis function extraction [13]; (3) *parameter transfer* [14] in which the parameters of the old task will be partially merged into the model of the new task. While supervised learning is a pure optimization problem, reinforcement learning is a more complex control problem. To the best of our knowledge, most of the existing transfer reinforcement learning algorithms follow the similar schema of the traditional transfer learning in deep learning and are heuristic with no theoretical guarantee of the convergence rate [15, 16, 17]. As mentioned by [18], the transfer learning method potentially does not work or even harm the new tasks and we do not know the reason since the absence of the theory. Therefore, it is very important for us to clearly understand how and when transfer learning will help reinforcement learning save sample complexity.

In this paper, we design a novel transfer learning method for Q-learning from the RL perspective rather than just borrow the methods from transitional transfer learning in deep learning. In addition, we provide theoretical guarantees for the novel transfer Q-learning method. Different from the existing transfer RL algorithms, we propose to transfer the Q-function learned in the old task as the temporal difference update target of the new task. The transfer process is controlled by the error condition. We call this new transfer Q-learning method *target transfer Q-learning*. The intuition is that when the two RL tasks are similar to each other, their optimal Q-function will be similar which means the transferred target is better (more close to the optimal Q-function than the current Q-function). Combining it with that a better target Q-function in Q-learning will help to accelerate the convergence, we may expect that the *target transfer Q-learning* method will outperform the Q-learning. Intuitively, how larger improvement of target transfer Q-learning with error condition can be achieved depends on the MDPs of the two tasks. However, the intuition cannot tell us more about when to implement the transferred Q-function and how will *target transfer Q-learning* achieve faster convergence rate compared with Q-learning.

To answer these questions, we analyze the convergence of target transfer Q-learning and prove its convergence rate. Specifically, we prove that the error of target transfer Q-learning consists of two parts: the initialization error and the sampling error. For a new task with fixed discounted factor γ, both of the errors are increasing with the *relative Q-function error ratio* β (*error ratio* for simplicity) which reflects whether the target Q-function is closer to the optimal Q-function in the new task or not compared with the current Q-function. The smaller the error ratio is, the closer target Q-function is to the optimal Q-function, and the faster the convergence rate is. Furthermore, if the error ratio is smaller than 1 in each update step, the target transfer Q-learning will converge faster than Q-learning.

Based on the convergence rate analysis, we design error condition for target transfer Q-learning as whether the error ratio is smaller than 1. Specifically, in each step, we transfer old task learned Q-function as the target if it will lead the error ratio β smaller than 1. If not, we clip the error ratio β to be 1 by still using the current Q-function as the target. Because the optimal Q-function in the error ratio cannot be obtained, we estimate the error ratio using the ratio of the Bellman error, which is a standard way in Q-learning.

Our experiments on different kinds of tasks fully support our convergence analysis and verify the effectiveness of our proposed target transfer Q-Learning with error condition.

The paper is organized as follows: In Section 2, we discuss the related works and review the background of Q-learning. In section 3, we present the Target Transfer Q-Learning (TTQL) intuitively and show the intuition behind the proposed method. In Section 4, we theoretically analyze the TTQL. We show the influence factor of the convergence rate of the TTQL and design error condition which is one of the key components of the TTQL method from the theoretical view. In section 5, we discuss the practical designing of the error condition. In section 6, we show the experiment results to verify the theoretical results and the effectiveness of our proposed algorithms. Finally, we conclude this paper in Section 7.

2. Preliminary

2.1. Related works

Transfer Learning in RL [16, 17] aims to improve learning in the new MDP task by borrowing knowledge from a related but different learned MDP task. In paper [19], the authors propose to use instance transfer in the Transfer Reinforcement Learning with Shared Dynamics (TRLSD) setting in which only the reward function is different between MDPs. In paper [20], the authors propose to use the representation transfer to learn the invariant feature space. The papers [21, 14] propose to use the parameter transfer to guide the exploration or to initialize the Q-function of the new task directly. In paper [22], the authors propose to use the meta-learning method to do transfer learning in RL. All these works are empirically evaluated and have no theoretical analysis for the convergence rate.

There are few works that have the convergence analysis. In paper [13], the authors use the representation transfer but only consider the special TRLSD setting. [12] proposes a transfer method based on instance transfer. They give the theoretical analysis of the asymptotic convergence without a finite sample performance guarantee. [23] proposes to transfer the Q-function learned from the old tasks directly to the new tasks using the variational method under the Bayesian setting. All these transfer reinforcement learning methods are similar to the transfer learning methods in deep learning and less involve the reinforcement learning algorithms such as Temporal-Difference update.

The proposed TTQL method can be naturally applied to the single agent reinforcement learning domains such as computer games[3], robotics[24], decision support system[25] and so on. Furthermore, recently, there have been many works that extend the traditional RL problems and methods to the multi-agent systems[26, 27, 28, 29]. Multi-agent systems are widely used in several domains, including robotics teams, distributed optimal control, multi-player games and so on [30, 31, 32, 33]. Transfer learning problems in multi-agent systems can be divided into two categories. Our method can be extended to multi-agent systems for the first type of problems that reusing the knowledge learned from the previous tasks. Another interesting problem of transfer learning in the multi-agent system is how one agent transfers the learned knowledge to another agent[34]. It is non-trivial and left for future work.

2.2. Q-Learning Background

Consider the reinforcement learning problem with Markov decision process (MDP) $M \triangleq (\mathcal{S}, \mathcal{A}, P, r, \gamma)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P = \{P^a_{s,s'}; s, s' \in \mathcal{S}, a \in \mathcal{A}\}$ is the transition matrix and $P^a_{s,s'}$ is the transition probability from state s to state s' after taking action a, $r = \{r(s,a); s \in \mathcal{S}, a \in \mathcal{A}\}$ is the reward function and $r(s,a)$ is the reward received at state s if taking action a. $\gamma \in (0,1)$ is the discount factor. We use $\pi : \mathcal{A} \times \mathcal{S} \to [0,1]$ to denote the policy which is the probability to take each action at each state. Value function for policy π is defined as: $V^\pi(s) \triangleq E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, \pi\right]$. Action value function is also called Q-function which is defined as: $Q^\pi(s,a) \triangleq E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a, \pi\right]$. The optimal policy is denoted as π^* and its corresponding value function and Q-function are denoted as $V^*_M(s)$ and $Q^*_M(s,a)$ respectively.

As we know, the Q-function in RL satisfies the following Bellman equation:

$$Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{\substack{\tilde{a} \sim \pi(a|s) \\ s' \sim P(s'|s,a)}} [Q^\pi(s', \tilde{a}) | s_t = s]$$

We denote the right hand side(RHS) of the above equation as $T^\pi Q^\pi(s,a)$ and T^π is called Bellman operator for policy π. Similarly, consider the optimal Bellman equation:

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{\substack{\tilde{a} \sim \pi(a|s) \\ s' \sim P(s'|s,a)}} [Q^*(s', \tilde{a}) | s_t = s] \tag{2.1}$$

The RHS of the equation has been denoted as $T^\pi Q^\pi(s,a)$ and T^* is called optimal Bellman operator. It can be proved that the optimal Bellman operator is a contraction mapping for the Q-function. We know that there is a unique fix point which is optimal Q-function by contraction mapping theorem. Q-learning algorithm is designed by the above theory. Watkins introduced the Q-learning algorithm to estimate the value of state-action pairs in discounted MDPs [6] using the following update rule:

$$Q_{t+1}(s,a) = (1 - \alpha_t) Q_t(s,a) + \alpha_t \left(r_t(s,a) + \gamma \max_{\tilde{a}} Q_t(s', \tilde{a}) \right)$$

In the following, we will use Q to denote $Q(s,a)$ if omit (s,a) will not cause misunderstanding.

Similar to many previous work[35, 36], we use $\|Q - Q^*\|_\infty$ to measure the quality of Q-function and denote it as $\mathbf{MNE}$: $\mathbf{MNE}(Q) = \|Q - Q^*\|_\infty = \max_{s,a} |Q(s,a) - Q^*(s,a)|$.

3. Target Transfer Q-Learning

First of all, we formalize transfer learning in the RL problem. Secondly, we propose our new transfer Q-learning method *Target Transfer Q-Learning* (TTQL) and introduce the intuition.

Transfer Learning in RL[16, 17] aims to improve learning in the new MDP task by borrowing knowledge from a related but different learned MDP task.

According to the definition of MDPs: $M \triangleq (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, we consider the situation that two MDPs are different in the transition probability P, the reward function r and the discount factor γ. The assumptions we used in our paper are standard assumptions to analyze theory in transfer RL, which are also widely used in other related works in transfer RL on both experimental and theoretical point of view [19, 13, 37].

Assume there are two MDPs: old MDP $M_1 = (\mathcal{S}, \mathcal{A}, P_1, r_1, \gamma_1)$ and new MDP $M_2 = (\mathcal{S}, \mathcal{A}, P_2, r_2, \gamma_2)$ and Q_1^* and Q_2^* are the corresponding optimal Q-functions. Let M_1 be the old MDP and we have already learned the Q_1^*. The goal of transfer in RL considered in this work is how we can use the information of M_1 and Q_1^* to achieve an improvement of learning speed in M_2.

To solve the problem mentioned above, we propose to use TTQL method. TTQL uses the Q-function learned from the old task as the target Q-function (refer to Algorithm 1) in the new task with error conditions. The error condition ensures that the transferred target only appears if it can help to accelerate the training. Otherwise, we will replace it with the current Q-function in the learning process of the new MDP. We describe the TTQL in Algorithm 1. The designing of **error-condition** is analyzed in Section 5 and the conclusion is shown in the Theorem 4. Detailed discussion is put in Section 6.

Algorithm 1 Target Transfer Q Learning

Input: initial Q-function Q_0 , old task learned Q-learning Q_{old}^*, total step n

for $t = 1, \ldots, n$ **do**
  $\alpha_t = \frac{1}{t}$
  flag = `error-condition`$(Q_{old}^*, Q_{t-1}(\cdot, \cdot))$
  if flag = True **then**
    $Q_{target} = Q_{old}^*$
  else
    $Q_{target} = Q_{t-1}$
  end if
  for $s \in \mathcal{S}, a \in \mathcal{A}$ **do**
    $Q_t(s,a) = (1-\alpha_t)Q_{t-1}(s,a) + \alpha_t \left(r(s,a) + \gamma \max_{\tilde{a}} Q_{target}(s', \tilde{a})\right)$
  end for
end for

Output: Q_t

The intuitive motivation is that when the two RL tasks are similar to each other, their optimal Q-function will be similar. Thus the transferred target is better (the max norm error is smaller than the current Q-function) and the better target can help to accelerate the convergence.

We define the distance between two MDPs as $\Delta(M_1, M_2)$

$$\Delta(M_1, M_2) = \max_{s,a} |Q_1^*(s,a) - Q_2^*(s,a)|.$$

The following Proposition 1 shows the relation between the distance of two MDPs and the components of two MDPs

Proposition 1. *Suppose that there are two MDPs, $M_1 = (\mathcal{S}, \mathcal{A}, P_1, r_1, \gamma_1)$ and $M_2 = (\mathcal{S}, \mathcal{A}, P_2, r_2, \gamma_2)$, Let the corresponding optimal Q-functions be Q_1^* and Q_2^*, then we have*

$$\begin{aligned} \Delta(M_1, M_2) &= \|Q_1^* - Q_2^*\|_\infty \leq \tilde{\Delta}(M_1, M_2) \\ &\triangleq \frac{\|r_1 - r_2\|_\infty}{1-\gamma'} + \frac{\gamma'' \|r'\|_\infty}{(1-\gamma'')^2} \|P_1 - P_2\|_\infty + \frac{|\gamma_1 - \gamma_2|}{(1-\gamma_1)(1-\gamma_2)} \|r''\|_\infty. \end{aligned} \tag{3.1}$$

for $\forall(\gamma', \gamma'', r', r'') \in \Omega$*, where* Ω *is the set of available combination of the* $(\gamma_1, \gamma_2, r_1, r_2)$.

Proof. Without loss of generality, we assume $\gamma_1 \leq \gamma_2$, $\|r_2\|_\infty \leq \|r_1\|_\infty$. We will show that other cases can be proved similarly. We define the following auxiliary MDPs: $\hat{M}_3 = (\mathcal{S}, \mathcal{A}, P_1, r_2, \gamma_1)$, $\hat{M}_4 = (\mathcal{S}, \mathcal{A}, P_2, r_2, \gamma_1)$, and let the corresponding optimal Q-functions be Q_3^* and Q_4^*. We have

$$\|Q_1^* - Q_2^*\|_\infty \tag{3.2}$$
$$= \|Q_1^* - Q_3^* + Q_3^* - Q_4^* + Q_4^* - Q_2^*\|_\infty \tag{3.3}$$
$$\leq \|Q_1^* - Q_3^*\|_\infty + \|Q_3^* - Q_4^*\|_\infty + \|Q_4^* - Q_2^*\|_\infty. \tag{3.4}$$

Notice that in each term, two MDPs are only different in one component. Using the results of [38], we have that $\|Q_1^* - Q_3^*\|_\infty \leq \frac{\|r_1 - r_2\|_\infty}{1-\gamma_1}$, $\|Q_3^* - Q_4^*\|_\infty \leq \frac{\gamma_1 \|r_2\|_\infty}{(1-\gamma_1)^2}\|P_1 - P_2\|_\infty$, $\|Q_4^* - Q_2^*\|_\infty \leq \frac{|\gamma_1 - \gamma_2|}{(1-\gamma_1)(1-\gamma_2)}\|r_2\|_\infty$. Combining the above upper bound and set $\gamma' = \gamma_1, \gamma'' = \gamma_1, r' = r_2, r'' = r_2$, we can get the in-equation (3.1).

In other situation, we can construct auxiliary MDPs like above and use the similar procedure to prove the theorem. After traversing all the available combinations of the $(\gamma_1, \gamma_2, \gamma_1, \gamma_2)$, we can prove the Proposition 1

□

By the Proposition 1, we can conclude that if the two RL tasks are similar, in the sense that the components of two MDPs are similar, the learned Q-function in the old task will be close to the optimal Q in the new task. A question is that when to transfer the target will have a performance guarantee. Here, we need error conditions that help to avoid harm to the new tasks and thus ensure the convergence of the algorithm. We can now heuristically relate it to the distance between two MDPs and the current learning quality. The concrete definition of the error condition needs to be further investigated through quantified theoretical analysis, and we present these results in the following section.

4. Convergence Rate of TTQL

In this section, we present the analysis of the convergence rate of the *Target Transfer Q-Learning* (TTQL) and make discussions on the key factors that influence the convergence. Theorem 1 shows the convergence of the target transfer method. Theorem 2 and 3 show two key factors of the convergence rate. Theorem 4 summarizes the convergence rate for the TTQL. Through the theorem, we can conclude that (1) The distance between two MDPs influences that convergence rate. (2) TTQL method does converge with the error condition. (3) TTQL method converges faster than traditional Q-learning.

First of all, We analyze the convergence rate for the target transfer method which is

$$Q_{n+1}(s,a) = (1 - \frac{1}{n})Q_n(s,a) + \frac{1}{n}\left(r(s,a) + \gamma \max_{\tilde{a}} Q_{target}(s', \tilde{a})\right).$$

We denote the error ratio $\beta_n = \frac{\mathbf{MNE}(Q_{target})}{E_n}$ .For simplicity, we sometimes use β to denote the error ratio if we do not specify the learning steps n and denote $E_n = \mathbf{MNE}(Q_n)$. Without loss of generality, we assume that all the rewards lie between 0 and 1. All the results in the paper holds for bounded rewards only with minor modifications.

Theorem 1. *The Q-function is update by the above update role. Then if* $0 \leq \beta_n \leq 1$*, with probability* $1 - \delta$ *we have*

$$E_n \leq \underbrace{\alpha_n E_1}_{\textit{initialization error}} + \underbrace{\sqrt{\frac{\ln 1/\delta \sum_{k=0}^{n-1} w_k^2(\beta_{n-k:n})}{2}}}_{\textit{sampling error}}.$$

where $w_k(\beta_{n-k:n}) = \frac{\prod_{i=n-k}^{n-1}(i+\gamma\beta_i)}{\prod_{i=n-k}^{n} i}$, $\alpha_n = \frac{\prod_{i=1}^{n-1}(i+\gamma\beta_i)}{\prod_{i=2}^{n} i}$.

Before showing the proof of Theorem 1, we first introduce a modified Hoeffding inequality which bounds the distance between the weighted sum of the bounded random variables and its expectation.

Lemma 1. *Let $a < x_i < b$ in probability 1 , $S_n = \sum_{i=1}^n w_i x_i$, then with probability at least $1-\delta$ we have*

$$S_n - E[S_n] \leq \sqrt{\frac{1}{2}\log\frac{1}{\delta}\sum_{k=1}^n w_k^2 (b-a)^2}. \tag{4.1}$$

Proof. We first prove the inequality $\mathbb{P}\left(S_n - E[S_n] \geq \epsilon\right) \leq exp\left(-\frac{2\epsilon^2}{\sum_{k=1}^n w_k^2 (b-a)^2}\right)$

For $s, \epsilon \geq 0$, Markov's inequality and the independence of x_i implies

$$\begin{aligned}
&\mathbb{P}\left(S_n - \mathrm{E}\left[S_n\right] \geq \epsilon\right) && (4.2)\\
&= \mathbb{P}\left(e^{s(S_n - \mathrm{E}[S_n])} \geq e^{s\epsilon}\right) && (4.3)\\
&\leq e^{-s\epsilon}\mathrm{E}\left[e^{s(S_n - \mathrm{E}[S_n])}\right] && (4.4)\\
&= e^{-s\epsilon}\mathrm{E}\left[e^{s(\sum_{i=1}^n w_i x_i - \mathrm{E}\left[\sum_{i=1}^n w_i x_i\right])}\right] && (4.5)\\
&= e^{-s\epsilon}\prod_{i=1}^n \mathrm{E}\left[e^{s w_i (x_i - \mathrm{E}[x_i])}\right] && (4.6)\\
&\leq e^{-s\epsilon}\prod_{i=1}^n e^{\frac{s^2 w_i^2 (b-a)^2}{8}} && (4.7)\\
&= \exp\left(-s\epsilon + \tfrac{1}{8}s^2(b-a)^2\sum_{i=1}^n w_i^2\right). && (4.8)
\end{aligned}$$

Now we consider the minimum of the right hand side of the last inequality as a function of s, and denote

$$g(s) = -s\epsilon + \tfrac{1}{8}s^2(b-a)^2\sum_{i=1}^n w_i^2$$

Note that g is a quadratic function and achieves its minimum at $s = \frac{4\epsilon}{(b-a)^2\sum_{i=1}^n w_i^2}$, Thus we get

$$\mathbb{P}\left(S_n - \mathrm{E}\left[S_n\right] \geq \epsilon\right) \leq exp\left(-\frac{2\epsilon^2}{\sum_{k=1}^n w_k^2 (b-a)^2}\right) \tag{4.9}$$

We can easily obtain the second part of the Lemma 1 by inverse the inequality. □

Proof of Theorem 1. Our analysis are derived based on the following synchronous generalized Q-learning setting. Comparing with the traditional synchronous Q-learning [2], we replace the target Q-function by the independent Q-function $Q'(s,a)$ rather than the current one $Q_n(s,a)$.

$$\begin{aligned}
&\forall s, a \ : \ Q_0(s,a) = q(s,a)\\
&\forall s, a \ : \ Q_n(s,a) =\\
&\quad (\frac{n-1}{n})Q_{n-1}(s,a) + \frac{1}{n}\left(r(s,a) + \gamma \max_{\tilde{a}} Q'_{n-1}(s', \tilde{a})\right)
\end{aligned} \tag{4.10}$$

[2] It is the same as the commonly used setting or more general([39, 35, 40, 41]).

Let $Q'_n(s,a)$ satisfy the following condition ,

$$0 \leq \frac{\max_{s,a}\left(Q'_n(s,a) - Q^*(s,a)\right)}{\max_{s,a}\left(Q_n(s,a) - Q^*(s,a)\right)} \leq 1. \tag{4.11}$$

Note that if we set $Q'_n(s,a) = Q^*_{source}$, we can verify $0 \leq \beta_n \leq 1$ according to inequality (4.11).

First of all, we decompose the update role as

$$\begin{aligned}
&Q_n(s,a) \\
&= \frac{n-1}{n} Q_{n-1}(s,a) + \frac{1}{n}\left[r(s,a) + \gamma \max_{\tilde{a}} Q'_{n-1}(s',\tilde{a})\right] \\
&= \frac{n-1}{n} Q_{n-1}(s,a) + \frac{1}{n}\Big[r(s,a) + \gamma \max_{\tilde{a}} Q^*(s',\tilde{a}) \\
&\qquad + \gamma \max_{\tilde{a}} Q'_{n-1}(s',\tilde{a}) - \gamma \max_{\tilde{a}} Q^*(s',\tilde{a})\Big].
\end{aligned}$$

If we denote $\epsilon_n(s,a) = Q_n(s,a) - Q^*(s,a)$, $x(s') = \gamma \max_{\tilde{a}} Q^*(s',\tilde{a})$ and recall the definition of β_n, we can have

$$\begin{aligned}
&\epsilon_n(s,a) \\
\leq &\frac{n-1}{n}\epsilon_{n-1}(s,a) + \frac{1}{n}\left[x(s') - \mathbb{E}_{s'}x(s')\right] + \frac{1}{n}\gamma\beta_n\epsilon_{n-1}(s',\tilde{a}) \\
\leq &\frac{n-1}{n}\epsilon_{n-1}(s,a) + \frac{1}{n}\left[x(s') - \mathbb{E}_{s'}x(s')\right] + \frac{1}{n}\gamma\beta_n E_{n-1}.
\end{aligned}$$

The last step in the above inequality is right because $\epsilon_n(s,a) \leq E_n$ for $\forall s, a$. Taking maximization of the both sides of the inequality and using recursion of E, we have

$$\begin{aligned}
E_n &\leq \frac{n-1+\gamma\beta_n}{n} E_{n-1} + \frac{1}{n}\left[x(s') - \mathbb{E}_{s'}x(s')\right] \\
&\leq \frac{\prod_{i=1}^{n-1}(i+\gamma\beta_i)}{\prod_{i=2}^{n} i} E_1 + \sum_{k=1}^{n-1} \frac{\prod_{i=n-k}^{n-1}(i+\gamma\beta_i)}{\prod_{i=n-k}^{n} i}\left[x(s'_k) - \mathbb{E}_{s'}x(s')\right] \\
&= \alpha_n E_1 + \sum_{k=1}^{n-1} w_k(\beta)\left[x(s'_k) - \mathbb{E}_{s'}x(s')\right]
\end{aligned}$$

According to Lemma 1(weighted Hoeffding inequality), with probability 1-δ, we can prove the Theorem 1

$$E_n \leq \alpha_n E_1 + \sqrt{\frac{\ln 1/\delta \sum_{k=0}^{n-1} w_k^2(\beta_{n-k:n})}{2}}. \tag{4.12}$$

□

The convergence result reveals how the error ratio β influence the convergence rate. In short, if we can find a better target Q-function, the learning process will be faster.

From Theorem 1, we can see that there are two key factors that influence the convergence rate. One is the initialization error $\alpha_n E_1$, the other one is the sampling error $\sqrt{\frac{\ln 1/\delta \sum_{k=0}^{n-1} w_k^2(\beta_{n-k,n})}{2}}$. To make it clear, we analyze the order of these two terms in Theorem 2 and Theorem 3 respectively.

Theorem 2. *Denote* $w_k(\beta_{n-k:n}) = \frac{\prod_{i=n-k}^{n-1}(i+\gamma\beta_i)}{\prod_{i=n-k}^{n} i}$, *and* $\beta_i \leq \beta^*$ *for* $\forall i \leq n$. *We have*

$$\sum_{k=0}^{n-1}\left(w_k(\beta_{n-k,n})\right)^2 \leq \begin{cases} \frac{e^{2\gamma\beta^*}}{n^{2-2\gamma\beta^*}}\left(\frac{n^{1-2\gamma\beta^*}}{1-2\gamma\beta^*} - \frac{1}{1-2\gamma\beta^*} + 1\right), \\ \qquad\qquad \gamma\beta^* \neq 0.5 \\ \frac{(n-2)^{2\gamma\beta^*}}{n^2} e^{2\gamma\beta^*}(1+\ln(n)), \\ \qquad\qquad \gamma\beta^* = 0.5 \end{cases}.$$

Based on the results of Theorem 2, we have

Corollary 1. *The order of* $\sum_{k=0}^{n-1}\left(w_k(\beta_{n-k:n})\right)^2$ *is:*
$\mathcal{O}(\frac{1}{n})$, *if* $\gamma\beta^* < 0.5$,.
$\mathcal{O}(\frac{1}{n^{2-2\gamma\beta^*}})$, *if* $0.5 < \gamma\beta^* < 1$.
$\mathcal{O}(\frac{1}{n^{2-2\gamma\beta^*}}\ln(n))$, *if* $\gamma\beta^* = 0.5$.
The sufficient condition for $\lim_{n\to\infty}\sum_{k=0}^{n-1}\left(w_k(\beta^*)\right)^2 = 0$ *is* $\gamma\beta^* < 1$.

Before showing the proof of Theorem 2, we first introduce a Lemma which will be used.

Lemma 2. *If* $a < b$, $\sum_{i=a}^{b}\frac{1}{i} \leq \frac{1}{a} + \ln(b) - \ln(a)$.

Proof.

$$\begin{aligned}\sum_{i=a}^{b}\frac{1}{i} &\leq \frac{1}{a} + \sum_{i=a+1}^{b}\frac{1}{i} \leq \frac{1}{a} + \sum_{i=a+1}^{b}\int_{k=i-1}^{i}\frac{1}{k}dk \\ &\leq \frac{1}{a} + \int_{k=a}^{b}\frac{1}{k}dk \leq \frac{1}{a} + \ln(b) - \ln(a)\end{aligned}$$

□

Proof of Theorem 2.

$$\begin{aligned}&\sum_{k=0}^{n-1}\left(w_k(\beta_{n-k:n})\right)^2 \leq \sum_{k=0}^{n-1}\left(\frac{\prod_{i=n-k}^{n-1}(i+\gamma\beta^*)}{\prod_{i=n-k}^{n} i}\right)^2 \\ &\underbrace{=}_{(a)} \sum_{k=0}^{n-1}\exp\left\{2\left[\sum_{i=n-k}^{n-1}\ln(i+\gamma\beta^*) - \sum_{i=n-k}^{n}\ln i\right]\right\} \\ &\underbrace{=}_{(b)} \frac{1}{n^2}\sum_{k=0}^{n-1}\exp\left\{2\sum_{i=n-k}^{n-1}\left[\ln(i+\gamma\beta^*) - \ln i\right]\right\} \\ &\underbrace{\leq}_{(c)} \frac{1}{n^2}\sum_{k=0}^{n-1}\exp\left\{2\sum_{i=n-k}^{n-1}\frac{\gamma\beta^*}{i}\right\} \\ &\underbrace{\leq}_{(d)} \frac{1}{n^2}\sum_{k=0}^{n-1}\exp\left\{2\gamma\beta^*\left[\ln(n-2) - \ln(n-k) + 1\right]\right\} \\ &= \frac{(n-2)^{2\gamma\beta^*}}{n^2}e^{2\gamma\beta^*}\sum_{t=1}^{n}\frac{1}{t^{2\gamma\beta^*}} \qquad (4.13)\end{aligned}$$

We rewrite the product term into the summarization term in Equation (a). Then we drop one term outside of the summarization to sum the index i from $n-k$ to $n-1$ in Equation (b). Inequality (c) follows the concave property of the $\ln$ function. Inequality (d) follows the relationship between summarization and integral as shown in Lemma 2.

If $\gamma\beta^* = 0.5$, $2\gamma\beta^* = 1$,

$$\sum_{k=0}^{n-1} \left(w_k(\beta_{n-k:n})\right)^2 \leq \frac{1}{n^{2-2\gamma\beta^*}} e^{\frac{2\gamma\beta^*}{n-1}} (1 + \ln(n))$$

If $\gamma\beta^* \neq 0.5$,

$$\sum_{k=0}^{n-1} \left(w_k(\beta^*)\right)^2 \leq \underbrace{\frac{1}{n^{2-2\gamma\beta^*}}}_{(e)} \underbrace{e^{2\gamma\beta^*}}_{(f)} \left(\underbrace{\frac{n^{1-2\gamma\beta^*}}{1-2\gamma\beta^*}}_{(g)} - \underbrace{\frac{1}{1-2\gamma\beta^*}}_{(h)} + 1 \right)$$

Note that term (f) is a constant.
If $\gamma\beta^* < 0.5$, term(g) will dominate the order, $\sum_{k=0}^{n-1} \left(w_k(\beta_{n-k:n})\right)^2$ will be $\mathcal{O}(\frac{1}{n})$.
If $\gamma\beta^* > 0.5$, term(h) will dominate the order, $\sum_{k=0}^{n-1} \left(w_k(\beta_{n-k:n})\right)^2$ will be $\mathcal{O}(\frac{1}{n^{2-2\gamma\beta^*}})$.
If $\gamma\beta^* = 0.5$, $\sum_{k=0}^{n-1} \left(w_k(\beta_{n-k:n})\right)^2$ will be $\mathcal{O}(\frac{1}{n^{2-2\gamma\beta^*}} \ln(n))$.
In all cases, $\sum_{k=0}^{n-1} \left(w_k(\beta_{n-k,n})\right)^2$ will converge to 0 as n will go to ∞. □

The next theorem shows the upper bound of the coefficient α_n in initialization error.

Theorem 3. *Denote $\alpha_n = \frac{\prod_{i=1}^{n-1}(i+\gamma\beta_i)}{\prod_{i=2}^{n} i}$, and $\beta_i \leq \beta^*$ for $\forall i \leq n$, We can bound α_n as:*

$$\alpha_n \leq \frac{(n-1)^{\gamma\beta^*}}{n}(1+\gamma\beta^*)e^{(0.5-\ln 2)\gamma\beta^*} = \frac{C^1_{\gamma,\beta^*}}{n^{1-\gamma\beta^*}}. \tag{4.14}$$

where $C^1_{\gamma,\beta^*} = (1+\gamma\beta^*)e^{(0.5-\ln 2)\gamma\beta^*}$ is a constant.

Proof of Theorem 3.

$$\alpha_n \leq \frac{\prod_{i=1}^{n-1}(i+\gamma\beta^*)}{\prod_{i=2}^{n} i} \tag{4.15}$$

$$= \exp\left\{\sum_{i=1}^{n-1} \ln(i+\gamma\beta^*) - \sum_{i=2}^{n} \ln i\right\} \tag{4.16}$$

$$= (1+\gamma\beta^*)\exp\left\{\sum_{i=2}^{n-1} \left(\ln(i+\gamma\beta^*) - \ln i\right) - \ln n\right\} \tag{4.17}$$

$$\leq (1+\gamma\beta^*)\exp\left\{\sum_{i=2}^{n-1} \left(\frac{\gamma\beta^*}{i}\right) - \ln n\right\} \tag{4.18}$$

$$\leq (1+\gamma\beta^*)\exp\left\{\gamma\beta^*(0.5 + \ln(n-1) - \ln 2) - \ln n\right\} \tag{4.19}$$

$$\leq \frac{(n-1)^{\gamma\beta^*}}{n}(1+\gamma\beta^*)e^{(0.5-\ln 2)\gamma\beta^*} \tag{4.20}$$

We rewrite the product term in the second equation into the summarization term. The third equation rearranges the terms. The first inequality follows the concave property of $\ln$ function. The second inequality follows the relation between summarization and integral(Lemma 2). □

We rewrite the product term in the second equation into the summarization term. The third equation rearranges

the terms. The first inequality follows the concave property of $\ln$ function. The second inequality follows the relation between summarization and integral(Lemma 2).

Let us consider the case if $\gamma\beta^* < 1$. The theorem 3 shows, α_n converge to 0 and the convergence rate is in order $\mathcal{O}(\frac{1}{n^{1-\gamma\beta^*}})$.

Combining Theorem 1, 2 and 3, we have the following Theorem:

Theorem 4. *The TTQL will converge if we set the **error condition** as whether*

$$\hat{\beta}_n \triangleq \frac{\Delta(M_1, M_2)}{E_n} \leq 1.$$

And the convergence rate is:

$$E_n \leq \begin{cases} \mathcal{O}(\frac{1}{n^{1-\gamma\beta}} E_1 + \sqrt{\frac{1}{n}}), & if\ \gamma\beta < 0.5 \\ \mathcal{O}(\frac{1}{n^{1-\gamma\beta}} E_1 + \frac{1}{n^{1-\gamma\beta}}\sqrt{\ln n}), & if\ \gamma\beta = 0.5 \\ \mathcal{O}(\frac{1}{n^{1-\gamma\beta}} E_1 + \frac{1}{n^{1-\gamma\beta}}), & if\ 0.5 < \gamma\beta < 1 \end{cases} . \tag{4.21}$$

We would like to make the following discussions:

(1) The distance between two MDPs influence the convergence rate. According to the Proposition 1, if two MPDs have the similar components(P, r, γ), the optimal Q-function of these two MDPs will be closed. The discounted error ratio $\gamma\beta_n$ will be relatively small in this situation and the convergence rate will be improved.

(2) The TTQL method does converge. As shown in Theorem 4, the TTQL method will converge. And the convergence rate changes under different discounted error ratios $\gamma\beta$. The smaller $\gamma\beta$ will lead to a quicker convergence rate. Intuitively, smaller β means that Q' provides more information about the optimal Q-function. Besides, the discount factor γ can be viewed as the "horizon" of the infinite MDPs. Smaller γ means that the expected long-term return is less influenced by the future information and the immediate reward is assigned larger weights.

(3) Error condition matters. As mentioned above, the error condition is defined as whether $\hat{\beta}_n \leq 1$. If the error condition is true, we set $Q_{target} = Q^*_{source}$ and $\gamma\beta_n = \gamma\hat{\beta}_n \leq \gamma < 1$. If error condition is false, we set $Q_{target} = Q_n$ and $\gamma\beta_n = \gamma < 1$. So with the error condition, TTQL algorithms do converge at any situation. At the beginning of the new task training, due to the large error of the current Q-function, $\beta_n = \hat{\beta}_n$ will be relatively small and the transfer learning will be greatly helpful. Speedup would come down as the error of current Q-function becomes smaller. Finally, when β is equal to or larger than one, we need to remove the transfer Q target which means to set $\beta = 1$ to avoid the harm brought by the transfer learning.

(4) Q-learning is the special case. Please note that the traditional Q-learning is a special case for target transfer Q-learning with $Q_{target} = Q_{n-1}$. Thus, the error ratio is a constant and $\beta_n = 1$. Our results are reduced to the traditional Q-learning. [42]. It shows that if $\beta < 1$ in TTQL, the TTQL converge faster than traditional Q-learning.

5. Discussion for Error Condition

Until now, we can conclude that TTQL will converge. TTQL method needs the error condition to guarantee the convergence. In this section, We discuss the error conditions.

In the beginning, our proposed error condition can guarantee the convergence of the algorithms generally. Heuristically, the error condition is related to the distance between two MDPs and the quality of the current value function. Then according to the Theorem 4, we know that the error condition is whether $\hat{\beta}_n \leq 1$. Under the transfer learning in RL setting, it means that the distance between two MDPs needs to be smaller than the error of the current Q-function.

If the two RL tasks are similar, the learned Q-function in the old task will be close to the optimal Q-function. Thus, the error ratio will be small (especially for the early learning stage) when we transfer the learned Q-function from the old task to the target of the new task. Specifically, in the early stage of the training, the Q-function in the new task is not fully trained, the learned Q-function in the old task is a better choice with a smaller error ratio. With the updating of the Q-function in the new task, its error ratio becomes larger. When its error ratio is close to or larger than 1, the error condition will change to false, and we will stop transferring the target to avoid the harm brought by the transfer learning.

In the real algorithms, it is impossible to calculate the error of the current Q-function $\mathbf{MNE}(Q_n)$ and the distance between two MDPs precisely. However it is easy to calculate the bellman error $\mathbf{MNBE}(Q(s,a)) = \max_{s,a} |Q(s,a) - (r(s,a) + \gamma E_{s'} \max_{\tilde{a}}(Q(s',\tilde{a})))|$. We can prove that these two metrics follow the relationship as:

$$\mathbf{MNE}(Q) \leq \frac{\mathbf{MNBE}(Q)}{1-\gamma}.$$

Following the standard way in Q-learning, we estimate the error ratio about the error of the Q-function w.r.t the optimal Q-function by the Bellman error.

Algorithm 2 Error Condition

Input: learned Q_1^* , current Q-function Q_n
1: **if** $\mathbf{MNBE}(Q_1^*) \leq \mathbf{MNBE}(Q_n)$ **then**
2: flag = True
3: **else**
4: flag = False
5: **end if**
Output: flag

Proof of the relation between $\mathbf{MNE}$ *and* $\mathbf{MNBE}$. Denote $\mathcal{B}Q(s,a) = r(s,a) - \gamma \mathbb{E}_{s'} \max_{\tilde{a}} Q(s',\tilde{a})$ as bellman operator.

$$\begin{aligned}
&\mathbf{MNE}(Q) \\
\leq &\|Q(s,a) - \mathcal{B}Q^*(s,a)\|_\infty + \|\mathcal{B}Q^*(s,a) - Q^*(s,a)\|_\infty \\
\leq &\mathbf{MNBE}(Q) + \|\gamma \mathbb{E}_{s'} \max_{\tilde{a}} Q(s',\tilde{a}) - \gamma \mathbb{E}_{s'} \max_{\tilde{a}} Q^*(s',\tilde{a})\| \\
\leq &\mathbf{MNBE}(Q) + \gamma \mathbf{MNE}(Q)
\end{aligned}$$

So we get

$$\mathbf{MNE}(Q) \leq \frac{\mathbf{MNBE}(Q)}{1-\gamma}.$$

□

6. Experiment

In this section, we conduct experiments to support our convergence analysis and verify the effectiveness of our proposed target transfer Q-Learning with error condition. Experiments are conducted on the simulation environments and a classical control problem. Firstly, our experiments on simulated MDPs verify our theoretical results. Secondly, we do realistic experiments on CartPole which are also used in many other transfer RL works. Considering that it is hard to implement the transfer setting in commonly used large scale RL environments, to the best of our knowledge, many transfer RL works still benchmark on these synthetic but standard environments([43, 19, 13, 37]).

First of all, we consider the general MDP setting. We construct the random MDP by generating the transition probability $P^a_{s,s'}$, reward function $r(s,a)$ and discount factor γ and fixing the state and action space size as 50. We generate 9 different MDPs ($M_{11}, M_{12}, M_{13}, M_{21}, M_{22}, M_{23}, M_{31}, M_{32}, M_{33}$) as old tasks and then generate the new MDP M_0 to be our new tasks. These 9 MDPs are in 3 groups and MDPs in the same groups are different in only one factor. Specifically, let M_{11}, M_{12}, M_{13} be different from M_0 only in discount factor γ. The order of their distances to $M_0$is $d(M_{11}, M_0) < d(M_{12}, M_0) < d(M_{13}, M_0)$. Similarly, MDPs M_{21}, M_{22}, M_{23} are different from M_0 only in reward function r, and MDPs M_{31}, M_{32}, M_{33} are different from M_0 only in transition probability P. The specific MDP parameters is listed in the appendix. We present the value of γ, r, P of each MDPs in the Table 1 of appendix.

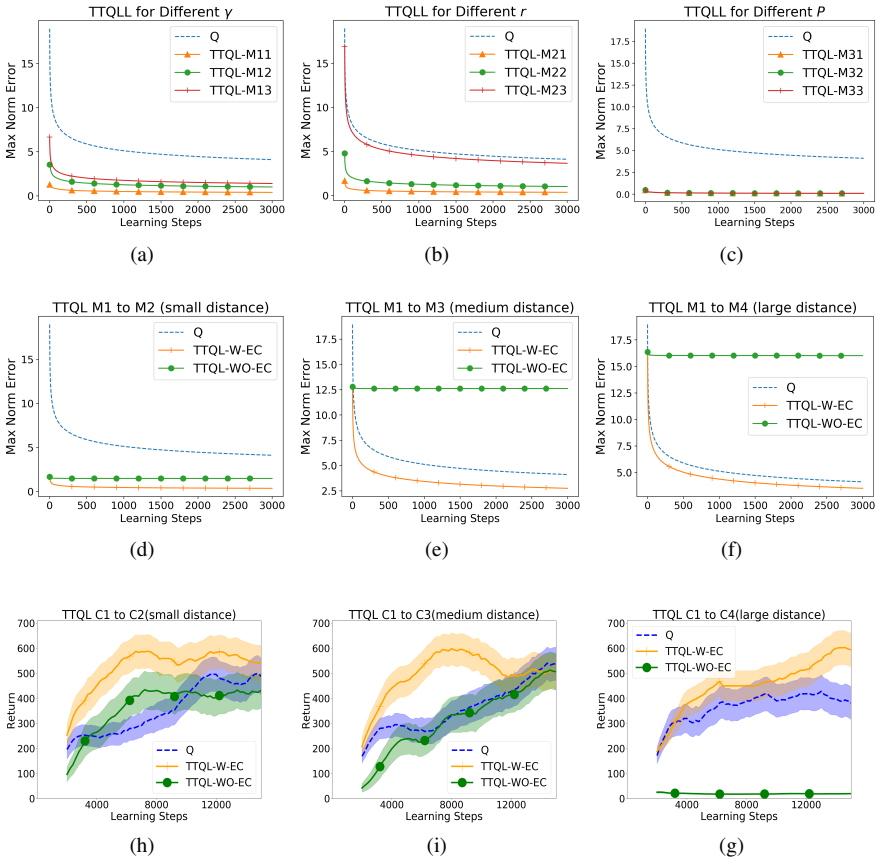


Figure 1: Above three figures are the learning max norm errors w.r.t the three types of different MDPs (Be different in γ, r, P respectively). Middle three figures are the learning error w.r.t the three different distance transfer task and both training with/without the error condition. Below three figures are three experiments results on CartPole environments. C1, C2, C3, C4 are 4 different MDPs(CartPole environments)

Then we run our algorithm to transfer the Q-function learned on these 9 source MPDs to the new MDP M_0. In this experiment, the Q function is represented by the look-up table. The result is shown in Figure1 (a), (b) and (c). Note that the dashed line is the result of the Q-learning algorithm without transfer learning, and the solid line with various markers are the TTQL algorithms.

Secondly, we design three MDPs M_4, M_5, M_6 similarly as above and leverage them as old task MDPs. They are different with the new tasks M_0 in all three factors and the order of their distances to $M_0$is $d(M_4, M_0) < d(M_5, M_0) < d(M_6, M_0)$. Then we use TTQL to transfer the Q-function learned from them to new MDP M_0 with and without the error condition. In this experiment, the Q function is represented by the look-up table. The results are shown in Figure1 (d), (e) and (f). We use "W-EC" and "WO-EC" to denote experiments that are run with and without error condition, respectively.

Thirdly, we conduct experiments on a classical control task: Cartpole. CartPole problem is a standard benchmark on control tasks[44]. As shown in Figure 2a, A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. we need to control the cart to prevent the pole from falling over(Figure 2b). We change the physical parameters (cart mass, pole mass, pole length) to get different but similar CartPole environments. Here we generate 4 CartPole environments denoted as C_1, C_2, C_3, C_4 respectively. We let one of the environments(C_1) as the old MDP and solve it by using Q-learning algorithms. Then we use TTQL to transfer the Q-function learned from the old MDP to a new MDP. The order of their distances to $C_1$is $d(C_2, C_1) < d(C_3, C_1) < d(C_4, C_1)$. The specific CartPole configures are listed in the appendix. We present the value of cart mass, pole mass and pole length of each CartPole tasks in Table 2 of appendix.

In the CartPole experiments, we follow the experiment settings of DQN[3]. Specifically, the Q-function is approximated by a 3 layer fully connected neural network. Specifically, we use the experience replay and freeze target network tricks as introduce by the DQN paper. The detailed explanation for the DQN method is shown in the last of the Appendix. The hyper-parameters of CartPole experiments are listed in the Table3 in Appendix. We evaluate the agent every 200 learning steps. The evaluation runs 10 episodes and the average return is reported. Since the realistic CartPole environments have a large variance, we run each experiment for 20 times and calculate the mean and standard deviation of the performances under each setting. The results are shown in Figure1.

(a) (b)

Figure 2: CartPole Environment

More experiments including the Atari games and the combination between our method with commonly used transfer learning method is shown in the Section 8.2 of Appendix.

We have the following observations. (1) TTQL method outperforms Q-learning in all experiments. (2) Running TTQL between the more similar MDPs will lead to a faster convergence rate. (3) The error condition is important to ensure the convergence of the algorithms in various situations. All these observations are consistent with our theoretical findings.

7. Conclusion and Future Work

In this paper, we have proposed a new transfer learning method in RL named *target transfer Q-learning* (TTQL). The method transfers the Q-function learned in the old task to the target of Q-learning in the new task and uses a theoretically designed error condition to control the transfer process. We have proved the TTQL method does converge with the error condition and the convergence rate is influenced by the distance of two MDPs. The theoretical analysis helps us to design error conditions that are important to guarantee the convergence of the transfer algorithms. The theoretical results have been verified by the experiments. In the future, we will apply the TTQL to more complex tasks and study convergence rate for the TTQL with complex function approximation such as the neural network.

Acknowledgment

This work is supported by the Fundamental Research Funds for the Central Universities(2018JBM320). Support by the German Research Foundation (DFG) through the CRC 1283 “Taming uncertainty and profiting from randomness and low regularity in analysis, stochastics and their applications” is acknowledged.

8. Appendix

8.1. Hyper-parameters

The parameters that we use to generate the random MDPs are listed in Table 1. The value in column γ means that we set the γ as that value. The range in column r means that we generate the reward function uniformly randomly in the range. The value in column P means that we multiply the diagonal elements of the transition probability matrix by that value and then renormalizes the matrix to a transition probability matrix.

MDPs	γ	r	P
M0	0.8	[3,5]	1
M11	0.79	[3,5]	1
M12	0.76	[3,5]	1
M13	0.75	[3,5]	1
M21	0.8	[3.5,4.5]	1
M22	0.8	[5,6]	1
M23	0.8	[0.52,0.47]	1
M31	0.8	[3,5]	1.02
M32	0.8	[3,5]	2
M33	0.8	[3,5]	10
M1	0.8	[3,5]	1
M2	0.8	[3.5,4.5]	1
M3	0.78	[7,8]	1
M4	0.75	[8,9]	1

Table 1: Parameters used to generate the random MDPs

The physical parameters for the 4 CartPole environments are listed in Table 3:

	cart mass	pole mass	pole length
C1	1	0.1	0.2
C2	1	0.3	0.2
C3	1	1	0.2
C4	1	1	0.8

Table 2: Parameters used to generate the CartPole environments

The extra hyper-parameters for the CartPole experiments are listed in Table 3.

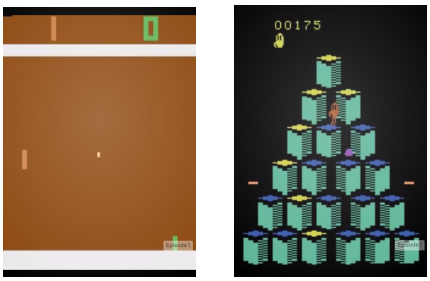
Hyperparameter	Value
MDP γ	0.99
Learning rate	0.01
Optimizer	Adam
Batch size	64
Buffer size	10000
Target network update frequency	50
NN hidden size	64

Table 3: Hyperparameters for the learning algorithms in CartPole environments

8.2. Experiments for Atari Games

We conduct our experiments on 2 Atari games: Pong and Qbert. The state space is the raw image of the game. Follow the setting in the paper [3] and most of the commonly used code base[45, 46], we preprocess the state image

from the $210 \times 160 \times 3$ RGB color image to the 84×84 gray image. The action space is 6 for all 3 games. According to the previous paper and the commonly used code base[45, 46], we usually apply the frame-skip wrapper into the environment. The frame-skip wrapper simulates the human performance that we usually press the button for a short time such as 4 frames rather than just one frame of the image. In other words, when the agent takes an action, the environment will execute the same action 4 times continuously. In the Atari games experiments, to construct the related but different environments, we set the number of skip frames into different values. The 4 skip-frame games are viewed as the old tasks and the 2 and 3 skip-frame games are viewed ad the new tasks. We transfer the knowledge learned from the old tasks into the new tasks.



(a) Pong (b) Qbert

Figure 3: Illustration of two Atari games.

The hyper-parameters for the DQN model and training process are listed in the following table. Most of the hyper-parameter is the same as the original DQN paper. The training details are explained in the last of the Appendix.

Hyperparameter	Value
MDP γ	0.99
Learning rate	0.00005
Optimizer	Adam
Batch size	32
Buffer size	1000000
Target network update frequency	10000
NN structure	Input - Conv2d(4, 32) - Conv2d(32,64) - Conv2d(64,64) - FC(3136,6) - Output

Table 4: Hyperparameters for Atari Games

We first run the DQN in the old tasks with 4 skip-frame for 60 million steps and then save the agent parameters. Then, we use our Target Transfer Q-learning method to train the agent in the new tasks with 2 and 3 skip-frame. The results are shown in Figure 4. We can see that Target Transfer Q learning can efficiently accelerate the learning process.

Additionally, we conduct experiments to further show that our transfer method can combine with other transfer learning methods proposed before. FT is a commonly used transfer learning method. For example, the model trained in the well known ImageNet dataset is always transferred to other computer vision tasks using fine-tune method[47] and the famous Bert model is always transferred to other neural language processing tasks using fine-tune method[48]. We compare plain no-transfer Q-learning algorithm(Q for short), vanilla transfer learning method(fine-tune-parameters, or FT for short) and TTQL combined fine-tune-parameters(TTQL-FT for short).

Specifically, the first experiment setting, plain no-transfer Q-learning algorithm, is the same as the algorithm proposed in DQN paper and we describe it detailedly above. The second experiment setting is fine-tune-parameters. In our experiments, we initialize the new agent parameters in the new games(2,3,5,6 skip-games) by using the agent parameters learned from the old tasks(4-skip games). Then we use the plain Q-learning algorithms to fine-tune the agent parameters. The third experiment setting is the TTQL combined fine-tune-parameters. The initialization procedure of TTQL-FT is the same as FT. And we then use TTQL to learning the agent parameters rather than the plain Q-learning.

The results are shown in Figure 5. We can see that the FT can outperform than Q-learning method since FT can

transfer the knowledge from related but different tasks. Furthermore, TTQL-FT can further improve the FT method which demonstrates that TTQL method can also combine with and improve the traditional transfer learning method.

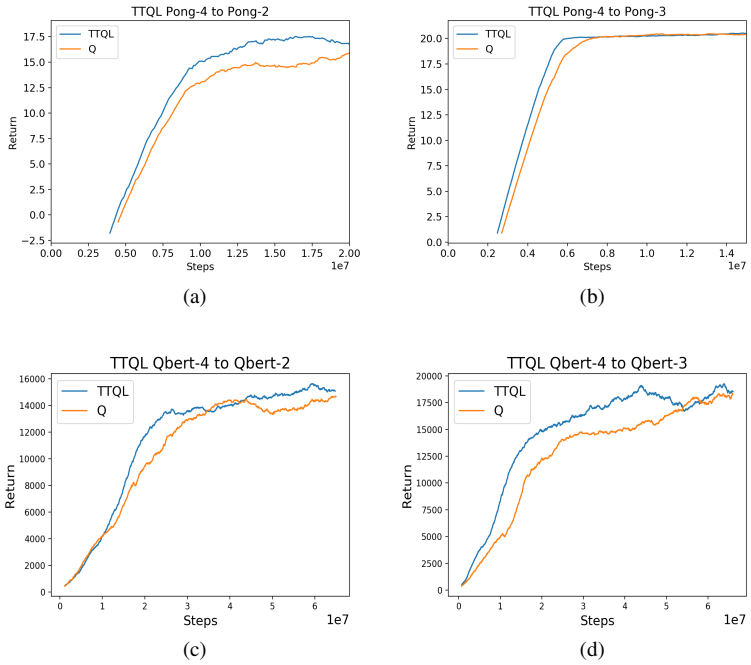


Figure 4: Results of Atari Games. x axis is the training steps and y axis is the expected return of on episode(Higher is better). Blue line(TTQL) is our method. Origin line(Q) is Q learning method.

8.3. DQN Background

Recall the main paper, MDP is defined as a five-tuple $M \triangleq (\mathcal{S}, \mathcal{A}, P, r, \gamma)$. Action value function is also called Q-function which is defined as: $Q^{\pi}(s,a) \triangleq E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a, \pi\right]$. The optimal policy is denoted as π^* and its corresponding Q-function are denoted as $Q_M^*(s,a)$. Optimal Q-function satisfy the optimal bellman equation. Traditional Q-learning find the optimal agent by minimize the discrepancy between the two side of the equation.

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{\substack{\tilde{a}\sim\pi(a|s) \\ s'\sim P(s'|s,a)}} [Q^*(s', \tilde{a}) | s_t = s] \tag{8.1}$$

In [3], the author first apply the Q-learning algorithms in to the human level computer games and achieve a super success. The proposed agent is called deepQ-network(DQN for short). It is because we use a deep neural networks to approximate the Q-function defined above. Similar to the raw definition of the Q-function, the input of the DQN is the raw image of the computer games and the agent is the action value of each action.

The algorithm modifies standard online Q-learning in two ways to make it suitable for training large neural networks without diverging. The first one is the experience replay buffer in which the algorithm stores the agent's experience at each time-step. The learning algorithm then sample the data from the buffer to update the parameters in the deep neural networks. The second one is a separate network for the target Q function. We keep the separate target Q network fixed for a certain steps and then update it rather than update it every step.

DQN update the parameters in the deep neural networks at iteration t uses the following loss function:

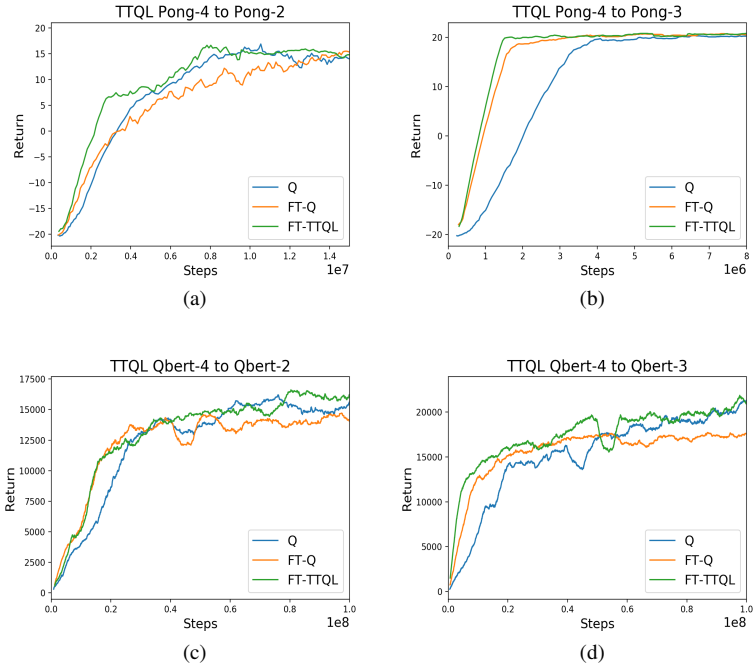


Figure 5: Results of Atari Games. x axis is the training steps and y axis is the expected return of one episode(Higher is better). Green line(FT-TTQL) is our method. Blue line(Q) is Q-learning method without any transfer learning. Origin line(FT-Q) is Q-learning method with Fine-Tune-Parameters method.

$$L(\theta_t) = E_{(s,a,r,s')\sim Buffer(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_t^-) - Q(s, a; \theta_t)\right)^2\right], \tag{8.2}$$

where θ_t is the parameter in the Q network at t iteration, θ_t^- is the parameter in the separate target Q network at t iteration, $Buffer(D)$ is the experience buffer, γ is the discount factor.

9. Reference

[1] R. S. Sutton, A. G. Barto, et al., Reinforcement learning: An introduction, MIT press, 1998.

[2] J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, The International Journal of Robotics Research 32 (11) (2013) 1238–1274.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.

[5] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, Y. Bengio, An actor-critic algorithm for sequence prediction, in: International Conference on Learning Representations, 2016.

[6] C. J. C. H. Watkins, Learning from delayed rewards, Ph.D. thesis, King's College, Cambridge (1989).

[7] T. Jaakkola, M. I. Jordan, S. P. Singh, Convergence of stochastic iterative dynamic programming algorithms, in: Advances in neural information processing systems, 1994, pp. 703–710.

[8] B. Li, Q. Yang, X. Xue, Transfer learning for collaborative filtering via a rating-matrix generative model, in: Proceedings of the 26th annual international conference on machine learning, ACM, 2009, pp. 617–624.

[9] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010) 1345–1359.

[10] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1717–1724.

[11] F. T. Sunmola, J. L. Wyatt, Model transfer for markov decision tasks via parameter matching, in: Proceedings of the 25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2006), 2006.

[12] Y. Zhan, M. E. Taylor, Online transfer learning in reinforcement learning domains, in: 2015 AAAI Fall Symposium Series, 2015.

[13] A. Barreto, R. Munos, T. Schaul, D. Silver, Successor features for transfer in reinforcement learning, in: Advances in neural information processing systems, 2017.

[14] J. Song, Y. Gao, H. Wang, B. An, Measuring the distance between finite markov decision processes, in: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 468–476.

[15] N. Bone, A survey of transfer learning methods for reinforcement learning, Computer Science Graduate and Undergraduate Student Scholarship. 5.

[16] M. E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: A survey, Journal of Machine Learning Research 10 (2009) 1633–1685.

[17] A. Lazaric, Transfer in reinforcement learning: a framework and a survey, in: Reinforcement Learning, Springer, 2012, pp. 143–173.

[18] B. Spector, S. Belongie, Sample-efficient reinforcement learning through transfer and architectural priors, in: Advances in neural information processing systems, 2017.

[19] R. Laroche, M. Barlier, Transfer reinforcement learning with shared dynamics, in: AAAI, 2017.

[20] A. Gupta, C. Devin, Y. Liu, P. Abbeel, S. Levine, Learning invariant feature spaces to transfer skills with reinforcement learning, in: International Conference on Learning Representations, 2017.

[21] T. G. Karimpanal, R. Bouffanais, Self-organizing maps as a storage and transfer mechanism in reinforcement learning, arXiv preprint arXiv:1807.07530.

[22] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, P. Abbeel, Continuous adaptation via meta-learning in nonstationary and competitive environments, in: International Conference on Learning Representations, 2018.

[23] A. Tirinzoni, R. Rodriguez Sanchez, M. Restelli, Transfer of value functions via variational methods, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2018, pp. 6179–6189.

[24] S. Barrett, M. E. Taylor, P. Stone, Transfer learning for reinforcement learning on a physical robot, in: AAMAS, 2010.

[25] M. Rabe, F. Dross, A reinforcement learning approach for a decision support system for logistics networks, in: 2015 Winter Simulation Conference (WSC), 2015, pp. 2020–2032.

[26] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, J. Wang, Mean field multi-agent reinforcement learning, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, Stockholmsmässan, Stockholm Sweden, 2018, pp. 5571–5580.

[27] J. N. Foerster, H. F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. M. Botvinick, M. Bowling, Bayesian action decoder for deep multi-agent reinforcement learning, in: Proceedings of the 36 International Conference on Machine Learning, 2019.

[28] Z. Cao, C.-T. Lin, Hierarchical critics assignment for multi-agent reinforcement learning, arXiv preprint arXiv:1902.03079.

[29] H. Liang, Y. Zhang, T. Huang, H. Ma, Prescribed performance cooperative control for multiagent systems with input quantization, IEEE Transactions on Cybernetics (2019) 1–10.

[30] Z. Yan, N. Jouandeau, A. A. Cherif, A survey and analysis of multi-robot coordination, International Journal of Advanced Robotic Systems 10 (12) (2013) 399.

[31] Y. Sheng, F. L. Lewis, Z. Zeng, T. Huang, Lagrange stability and finite-time stabilization of fuzzy memristive neural networks with hybrid time-varying delays, IEEE Transactions on Cybernetics (2019) 1–12.

[32] J. Zhang, Z. Wang, H. Zhang, Data-based optimal control of multiagent systems: A reinforcement learning design approach, IEEE Transactions on Cybernetics 49 (12) (2019) 4441–4449.

[33] M. Mazouchi, M. bagher Naghibi-Sistani, S. K. H. Sani, A novel distributed optimal adaptive control algorithm for nonlinear multi-agent differential graphical games, IEEE/CAA Journal of Automatica Sinica 5 (1) (2018) 331–341.

[34] F. L. Da Silva, A. H. R. Costa, Transfer learning for multiagent reinforcement learning systems, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, AAAI Press, 2016, pp. 3982–3983.

[35] E. Even-Dar, Y. Mansour, Learning rates for q-learning, Journal of Machine Learning Research 5 (2003) 1–25.

[36] K. Asadi, M. L. Littman, An alternative softmax operator for reinforcement learning, in: Proceedings of the 34th International Conference on Machine Learning, 2017.
[37] A. Tirinzoni, R. R. Sanchez, M. Restelli, Transfer of value functions via variational methods, in: Advances in Neural Information Processing Systems, 2018, pp. 6182–6192.
[38] B. C. Csáji, L. Monostori, Value function based reinforcement learning in changing markovian environments, Journal of Machine Learning Research 9 (Aug) (2008) 1679–1709.
[39] K. Asadi, M. L. Littman, An alternative softmax operator for reinforcement learning, in: Proceedings of the 34th International Conference on Machine Learning, Vol. 70 of Proceedings of Machine Learning Research, PMLR, International Convention Centre, Sydney, Australia, 2017, pp. 243–252.
[40] M. Ghavamzadeh, H. J. Kappen, M. G. Azar, R. Munos, Speedy q-learning, in: Advances in neural information processing systems, 2011, pp. 2411–2419.
[41] T. Haarnoja, H. Tang, P. Abbeel, S. Levine, Reinforcement learning with deep energy-based policies, in: International Conference on Machine Learning, 2017, pp. 1352–1361.
[42] C. Szepesvári, The asymptotic convergence-rate of q-learning, in: Advances in Neural Information Processing Systems, 1998, pp. 1064–1070.
[43] C. Ma, J. Wen, Y. Bengio, Universal successor representations for transfer reinforcement learning (2018).
[44] A. G. Barto, R. S. Sutton, C. W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE transactions on systems, man, and cybernetics SMC-13 (5) (1983) 834–846.
[45] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, P. Zhokhov, Openai baselines, `https://github.com/openai/baselines` (2017).
[46] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, M. G. Bellemare, Dopamine: A Research Framework for Deep Reinforcement Learning. URL `http://arxiv.org/abs/1812.06110`
[47] S. Kornblith, J. Shlens, Q. V. Le, Do better imagenet models transfer better?, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2661–2671.
[48] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805.