Sequential Decision Making for Intelligent Agents
Papers from the AAAI 2015 Fall Symposium

Online Transfer Learning in Reinforcement Learning Domains

Yusen Zhan, Matthew E. Taylor
School of Electrical Engineering and Computer Science
Washington State University
{yzhan,taylorm}@eecs.wsu.edu

Abstract

This paper proposes an online transfer framework to capture the interaction among agents and shows that current transfer learning in reinforcement learning is a special case of online transfer. Furthermore, this paper re-characterizes existing agents-teaching-agents methods as online transfer and analyze one such teaching method in three ways. First, the convergence of Q-learning and Sarsa with tabular representation with a finite budget is proven. Second, the convergence of Q-learning and Sarsa with linear function approximation is established. Third, the we show the asymptotic performance cannot be hurt through teaching. Additionally, all theoretical results are empirically validated.

Introduction

Agents can autonomously learn to master sequential decision tasks by reinforcement learning (Sutton and Barto 1998). Traditionally, reinforcement learning agents are trained and used in isolation. More recently, the reinforcement learning community became interested in interaction among agents to improve learning.

There are many possible methods to assist agent's learning (Erez and Smart 2008; Taylor and Stone 2009). This paper focuses on *action advice* (Torrey and Taylor 2013): as the student agent practices, the teacher agent suggests actions to take. This method requires only agreement of the action sets between teachers and students, while allowing for different state representations and different learning algorithms among teachers and students.

Although this advice method is shown to empirically provides multiple benefits (Torrey and Taylor 2013; Zimmer, Viappiani, and Weng 2014), existing work does not provide a formal understanding of teaching or advice. Therefore, this paper proposes a framework — **an online transfer framework** — to characterize the interaction among agents, aiming to understand the teaching or advice from the transfer learning perspective. We extend the transfer learning framework in reinforcement learning proposed by Lazaric (2012) into online transfer learning which capture the online interaction between agents. Also, we show that 1) transfer learning is a special case of online transfer framework, and 2) our framework is similar to that of of active learning (Settles 2010), but in a reinforcement learning setting.

After introducing our novel framework, it can be used to analyze existing advice methods, such as action advice (Torrey and Taylor 2013). First, we prove the convergence of Q-learning and Sarsa with tabular representation with a finite amount of advice. Second, the convergence of Sarsa and Q-learning with linear function approximation is established with finite advice. The convergence means the algorithms converge to the optimal Q-value. Third, we show that a non-infinite amount of advice cannot change the student's asymptotic performance. These three results are then confirmed empirically in a simple Linear Chain MDP and a more complex Pac-Man simulation.

Background

This section provides necessary background, adopting some notation introduced elsewhere (Sutton and Barto 1998; Melo, Meyn, and Ribeiro 2008).

Markov Decision Process

Let $M = \langle S, A, P, R, \gamma \rangle$ be a Markov decision process (MDP) with a compact state set S and a finite action set A. P is the transition probability kernel. For any $(s, a, s') \in S \times A \times S$ tuple the probability of transition from state s taking action a to state s' is defined as $\mathcal{P}[s' \in U|s, a] = P(U|s, a)$, where U is a Borel-measurable subset[1] of S. $R : S \times A \times S \to \mathbb{R}$ is a bounded deterministic function which assigns a reward $R(s, a, s')$ to transition from state s to state s' taking action a. The discount factor is γ such that $0 \leq \gamma \leq 1$. The expected total discounted reward for M under some policy can be defined as $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, where t is the time step and $r(s_t, a_t)$ denotes the reward received for taking action a_t in state s_t at time step t, according to reward distribution. For convenience, we omit the state and the action and only use r_t to denote the reward received at time step t, so the expected total discounted reward can be written as $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. $r(s, a)$ and $R(s, a, s')$ have following relationship: $\mathbb{E}[r(x, a)] = \int_S R(s, a, s') P(ds'|s, a)$.

A policy is a mapping that outputs for each state-action pair (s, a). A deterministic policy π is a mapping defined as

[1] Details on Borel-measurable subsets can be found elsewhere (Rudin 1986).

$\pi : S \to A$, while a stochastic policy is a mapping defined over $S \times A$ (i.e., $\mathcal{P}$[choose action a|at state s] $= \pi(s, a)$.)

The state-action function is the expected return for a state action pair under a given policy: $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a\right]$. Solving an MDP usually means finding an optimal policy that maximizes the expected return. An optimal policy $\pi^\star$ is such that $Q^{\pi^\star} \geq Q^\pi$ for all $s \in S$, all $a \in A$ and all policies π. We can define the optimal state-value function $Q^\star$ as $Q^\star(s, a) = \int_S (R(s, a, s') + \gamma \max_{a' \in A} Q^\star(s', a')) P(ds'|s, a)$, representing the expected total discounted reward received along an optimal trajectory when taking action a in state s and following optimal policy $\pi^\star$ thereafter. For all $s \in S$, $\pi^\star(s) = \arg\max_{a \in A} Q^\star(s, a)$. Notice that although a stochastic policy may be optimal, there will always be a deterministic optimal policy with at least as high an expected value.

Q-learning and Sarsa

Q-learning is an important learning algorithm in reinforcement learning. It is a model-free and off-policy learning algorithm which is a break-through in reinforcement learning control. Watkins (1989) introduced Q-learning as follows:

Given any estimate Q_0, Q-learning algorithm can be represented by following update rules:

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha_t(s, a)\Delta_t \tag{1}$$

where Q_t denotes the estimation of $Q^\star$ at time t, $\{\alpha_t(s, a)\}$ denotes the step-size sequence and Δ_t denotes temporal difference at time t,

$$\Delta_t = r_t + \gamma \max_{a' \in A} Q_t(s', a') - Q_t(s, a) \tag{2}$$

where r_t is the reward received at time step t. The update Equation 2 does not dependent any policies, so the Q-learning is called off-policy algorithm.

In contrast to off-policy algorithms, there are some on-policy algorithms in which Sarsa is the analogy of Q-learning (Rummery and Niranjan 1994). Given any estimate Q_0 and a policy π, the difference between Q-learning and Sarsa is that the temporal difference Δ_t:

$$\Delta_t = r_t + \gamma Q_t(s', a') - Q_t(s, a) \tag{3}$$

where a' is determined by the policy π and r_t is the reward received at time step t. Notice that the action selection in Equation 3 involves the policy π, making it on-policy.

If both S and A are finite sets, the Q-value function can be easily represented by an $|S| \times |A|$ matrix and it can be represented in a computer by a table. This matrix representation is also called tabular representation. In this case, the convergence of Q-learning, Sarsa, and other related algorithms (such as TD(λ)) have been shown by previous work (Peter 1992; Watkins and Dayan 1992; Singh et al. 2000). However, if S or A is infinite or very large, it is infeasible to use tabular representation and a compact representation is required (i.e., function approximation). This paper focuses on Q-learning with linear function approximation and Sarsa with linear function approximation. The linear approximation means that state-value function Q can be represented by a linear combination of features $\{\phi_i\}_{i=1}^d$, where $\phi_i : S \times A \to \mathbb{R}$ is the feature and d is the number of features. Given a state $s \in S$ and an action $a \in A$, the action value at time step t is defined as

$$Q_t(s, a) = \sum_{i=1}^{d} \theta_t(i)\phi_i(s, a) = \boldsymbol{\theta}_t^\intercal \boldsymbol{\phi}(s, a) \tag{4}$$

where $\boldsymbol{\theta}_t$ and $\boldsymbol{\phi}$ are d-dimensional column vectors and $^\intercal$ denotes the transpose operator. Since $\boldsymbol{\phi}$ is fixed, algorithms only are able to update $\boldsymbol{\theta}_t$ each time. Gradient-descent methods are one of most widely used of all function approximation methods. Applying a gradient-descent method to Equation 1, we obtain approximate Q-learning:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha_t(s, a)\nabla Q_t(s, a)\Delta_t \\ &= \boldsymbol{\theta}_t + \alpha_t(s, a)\boldsymbol{\phi}(s, a)\Delta_t \end{aligned} \tag{5}$$

where $\{\alpha_t\}$ is the update parameter at time t and Δ_t is the temporal difference at time step t (Equation 2). Similarly, given a policy π, the on-policy temporal difference can be defined as

$$\begin{aligned} \Delta_t &= r_t + \gamma Q_t(s', a') - Q_t(s, a) \\ &= r_t + \gamma \boldsymbol{\theta}_t^\intercal \boldsymbol{\phi}(s', a') - \boldsymbol{\theta}_t^\intercal \boldsymbol{\phi}(s, a), \end{aligned} \tag{6}$$

where a' is determined by the policy π at time t. Combining Equation 5 and Equation 6, we obtain Sarsa with linear approximation. For a set fixed features $\{\Phi_i : S \times A \to \mathbb{R}\}$, our goal is to learn a parameter vector $\boldsymbol{\theta}_\star$ such that $\boldsymbol{\theta}_\star^\intercal \boldsymbol{\Phi}(s, a)$ approximates the optimal Q-value $Q^\star$.

Online Transfer Framework

This section introduces a framework for online transfer learning in reinforcement learning domains, inspired by previous work (Lazaric 2012).

Online Transfer

Transfer learning is a technique that leverages past knowledge in one or more *source tasks* to improve the learning performance of a learning algorithm in a *target task*. Therefore, the key is to describe the knowledge transferred between different algorithms. A standard reinforcement learning algorithm usually takes input some raw knowledge of the task and returns a solution in a possible set of solutions. We use $\mathscr{K}$ to denote the space of the possible input knowledge for learning algorithms and $\mathscr{H}$ to denote the space of hypotheses (possible solutions, e.g., policies and value functions). Specifically, $\mathscr{K}$ refers to all the necessary input information for computing a solution of a task, e.g., samples, features and learning rate.

In general, the objective of transfer learning is to reduce the need for samples from the target task by taking advantage of prior knowledge. An online transfer learning algorithm can been defined by a sequence of transferring and learning phases, e.g., 1) transferring knowledge, 2) learning, 3) transferring based on previous learning, 4) learning, etc. Let $\mathscr{K}_s^L$ be the knowledge from L source tasks, $\mathscr{K}_t^i$ be the knowledge collected from the target task at time i and $\mathscr{K}_{learn}^i$ be the knowledge obtained from learning algorithm

at time i (including previous learning phases). We define one time step as one-step update in a learning algorithms or one batch update in batch learning algorithms. Thus, the algorithm may transfer **one-step** knowledge, or **one-episode** knowledge, or even **one-task** or **multi-task** knowledge to the learner, depending on the setting. $\mathscr{K}^i$ denote the knowledge space with respect to time i such that $\mathscr{K}^i \subseteq \mathscr{K}$, for all $i = 0, 1, 2, \ldots$. The online transfer learning algorithm can be defined as

$$\mathscr{A}_{transfer} : \mathscr{K}_s^L \times \mathscr{K}_t^i \times \mathscr{K}_{learn}^i \to \mathscr{K}_{transfer}^i \quad (7)$$

where $\mathscr{K}_{transfer}^i$ denotes the knowledge transferred to the learning phase at time i, $i = 0, 1, 2, \ldots$. Notice that $\mathscr{K}_{learn}^i$ is generated by the learning algorithm. Thus, the reinforcement learning algorithm can be formally described as

$$\mathscr{A}_{learn} : \mathscr{K}_{transfer}^i \times \mathscr{K}_t^i \to \mathscr{K}_{learn}^{i+1} \times \mathscr{H}^{i+1} \quad (8)$$

where $\mathscr{K}_{learn}^{i+1}$ is the knowledge from learning algorithm at time $i+1$ and $\mathscr{H}^{i+1}$ is the hypothesis space at time $i+1$, $i = 0, 1, 2, \ldots$. $\mathscr{K}_{learn}^{i+1}$ is used as input for next time step in online transfer Equation 7. Then, $\mathscr{A}_{transfer}$ generates the transferred knowledge $\mathscr{K}_{transfer}^i$ for learning phase in Equation 8. $\mathscr{A}_{learn}$ computes the $\mathscr{K}_{learn}^{i+1}$ for the next time step, and so on. In practice, the initial knowledge from the learning phase, $\mathscr{K}_{learn}^0$ can be empty or any default value. In this framework, we expect the hypothesis space sequence $\mathscr{H}^0, \mathscr{H}^1, \mathscr{H}^2, \ldots$ will become better and better over time under some criteria (e.g., the maximum average reward or the maximum discounted reward), where $\mathscr{H}^i \subseteq \mathscr{H}$ is the space of hypothesis with respect to i, $i = 0, 1, 2, \ldots$, that is, the space of possible solutions at time i. See Figure 1 for an illustration.

Example 1. *Consider the Active Relocation Model (Mihalkova and Mooney 2006). In this setting, there is an expert and a learner, which can be treated as the transfer algorithm $\mathscr{A}_{transfer}$ and the learning algorithm $\mathscr{A}_{learn}$, respectively. The learner is able to relocate its current state to a visited state, but the learner may become stuck in a sub-optimal state. Thus, the expert is able to help the learner to relocate its current state to a better state according to the expert's knowledge. This algorithm can be represented in our framework as $\mathscr{K}_s = (S \times A \times S \times R)^{N_s}$, where N_s is the number of samples the expert collect from the source tasks, $\mathscr{K}_t^i = (S_i \times A_i \times S_i \times R_i)^{N_i}$, $\mathscr{K}_{learn}^i = (\hat{Q}_i \times S_i \times A_i)$, $\mathscr{K}_{transfer}^i = (S_i)$ and $\mathscr{H}^{i+1} = \{\hat{Q}_{i+1}\}$, $i = 0, 1, \ldots, n$*[2].

Although we explicitly introduce $\mathscr{K}_t^i$ and $\mathscr{K}_{learn}^i$ in Equation 7 and 8, in most settings, it is impossible for the transfer algorithm and learning algorithm to explicitly access the knowledge from target tasks or it only has a limited access to it. For example, the communication failure and restrictions may cause these problems.

[2] S_i, A_i, R_i are all subsets of the set S of states, the set A of actions, and the set R of rewards, respectively. $\hat{Q}_i$ is the estimate of Q-value function at time i. We introduce the index to distinguish the the difference in different time steps. For example, the learning algorithm is able to reach more states at time step $i+1$ than at time step i. Thus, $S_i \subseteq S_{i+1}$.

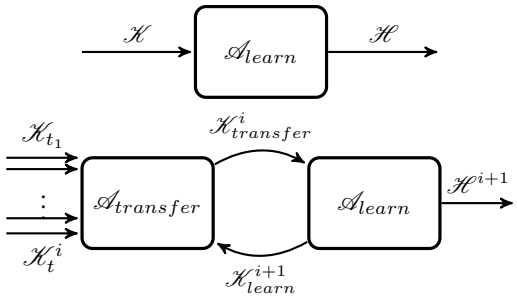

Figure 1: *(Top)* The standard learning process only requires original knowledge from the target tasks. *(Bottom)* In the online transfer learning process, transfer algorithm takes input knowledge from source tasks, target task and learner at time i and output transfer knowledge at time i, then the leaning algorithm takes the transfer knowledge at time i to generate hypothesis at time $i+1$. This process will repeat until a good hypothesis is computed.

Transfer Learning and Online Transfer Learning

Our online transfer learning framework can be treated as an online extension of the transfer learning framework (Lazaric 2012). If we set all $\mathscr{K}_{learn}^i = \emptyset$ and set $i = 0$, we have

$$\mathscr{A}_{transfer} : \mathscr{K}_s^L \times \mathscr{K}_t^0 \times \emptyset \to \mathscr{K}_{transfer}^0 \quad (9)$$

$$\mathscr{A}_{learn} : \mathscr{K}_{transfer}^L \times \mathscr{K}_t^0 \to \emptyset \times \mathscr{H}^1 \quad (10)$$

where the the $\mathscr{A}_{transfer}$ transfers the knowledge to the $\mathscr{A}_{learn}$ once, returning to the classic transfer learning scenario.

Advice Model with Budget

Now we discuss an advice method in previous work (Torrey and Taylor 2013), a concrete implementation of online transfer learning. Suppose that the teacher has learned an effective policy π_t for a given task. Using this fixed policy, it will teach students beginning to learn the same task. As the student learns, the teacher will observe each state s the student encounters and each action a the student takes. Having a budget of B advice, the teacher can choose to advise the student in $n \leq B$ of these states to take the "correct" action $\pi_t(s)$.

The authors (Torrey and Taylor 2013) assumed the teacher's action advice is always correct and that students were required to execute suggested actions. Suppose that a reinforcement learning teacher agent T is trained in a task and has access to its learned Q-Value function Q_t. Then, a student agent S begins training in the task and is able to accept advice in the form of actions from the teacher. We use notation (T, S, π_d) to denote the advice model where T is the teacher agent, S is the student agent and π_d is the policy teacher that provides its advice to student. The following example illustrates the how to characterize this advice model in the context of our online transfer learning framework.

Example 2. *Let us consider the advice model using Mistake Correcting approach with limited budget and linear function approximation (Torrey and Taylor 2013). In this model, there is a teacher and a student, which can be treated as the transfer algorithm and the learning algorithm, respectively. First, the transfer algorithm $\mathscr{A}_{transfer}$ collects N_s samples from L source tasks. Then, it will return an advice action a to the learning algorithm $\mathscr{A}_{learn}$ according to the current state and the action observed from the learning algorithm (initial knowledge is empty). The learning algorithm $\mathscr{A}_{learn}$ takes the advice action a and N_i samples from target task and returns a state and a action for next step, meanwhile, the $\mathscr{A}_{learn}$ maintains a function in the space $\mathscr{H}^{i+1}$ spanned by the features $\{\phi_j\}_{j=1}^{n}$, where $\phi_j : S \times A \to \mathbb{R}$ is defined by a domain expert. Moreover, the teacher has a limited budget n for advising the student, so the time step $i = 0, 1, \ldots, n$. Therefore, we have $\mathscr{K}_s = (S \times A \times S \times R)^{N_s}$, $\mathscr{K}_t^i = (S_i \times A_i \times S_i \times R_i)^{N_i}$, $\mathscr{K}_{learn}^i = (S_i \times A_i)$, $\mathscr{K}_{transfer}^i = (A_i)$ and $\mathscr{H}^{i+1} = \{f(\cdot,\cdot) = \sum_{j=1}^{d} \theta_{i+1}(j)\phi_j\}$, $i = 0, 1, \ldots, n$.*

Theoretical Analysis

For an advice model (T, S, π_d) we propose in this paper, the most important theoretical problem is to resolve the convergence of algorithms **since it guarantees the correctness of algorithms**. In the next subsection, we will discuss how action advice interacts with the tabular versions of Q-learning and Sarsa. After, the corresponding algorithms with linear function approximation are discussed.

Tabular Representation

The convergence of Q(0) (Q-learning) has been established by many works (Watkins and Dayan 1992; Jaakkola, Jordan, and Singh 1994; Tsitsiklis 1994; Mohri, Rostamizadeh, and Talwalkar 2012). We will use the one of convergence results from (Mohri, Rostamizadeh, and Talwalkar 2012)

Lemma 1. *((Mohri, Rostamizadeh, and Talwalkar 2012) Theorem 14.9 page 332) Let M be a finite MDP. Suppose that for all $s \in S$ and $a \in A$, the step-size sequence $\{\alpha_t(s_t, a_t)\}$ such that*

$$\sum_t \alpha_t(s_t, a_t) = \infty \qquad \sum_t \alpha_t(s_t, a_t)^2 < \infty,$$

Then, the Q-learning Algorithm converges with probability 1.

Notice that the conditions on $\alpha_(s_t, a_t)$ ensure the infinity visits of action-state pairs.

Theorem 1. *Given an advice model (T, S, π_d), the student S adopts the Q-learning Algorithm and conditions in Lemma 1 all hold, convergence of Q-learning still holds in the advice model setting.*

Proof. Notice that the conditions on $\alpha_t(s_t, a_t)$ verifies that each state-action pair is visited infinitely many times. And there is finite advice in our advice model. Therefore, the assumptions still hold in advice model setting. Apply Lemma 1, the convergence result follows. □

Compared to Q-learning, Sarsa is a on-policy algorithm which requires a learning policy to update the Q values. (Singh et al. 2000) prove that Sarsa with GLIE policy converges. We use their result to prove the convergence of Sarsa in advice model. First of all, we need to define GLIE policy.

Definition 1. *A decaying policy π is called GLIE, greedy in the limit with infinite exploration, policy, if it satisfies following two conditions:*

- *each state-action pair is visited infinity many times;*
- *the policy is greedy with respect to the Q-value function with probability 1.*

It is not hard to verify that the Boltzmann exploration policy satisfies the above two conditions. Then we provide the result from Singh *et al.*.

Lemma 2. *((Singh et al. 2000)) Let M be a finite MDP and π is a GLIE policy. If the step-size sequence $\{\alpha_t(s_t, a_t)\}$ such that*

$$\sum_t \alpha_t(s_t, a_t) = \infty \qquad \sum_t \alpha_t(s_t, a_t)^2 < \infty,$$

Then, the Sarsa Algorithm converges with probability 1.

Proof. (Singh et al. 2000) prove a similar convergence result under a weaker assumption, they assume that $Var(r(s, a)) < \infty$. In this paper, we assume that $r(s, a)$ is bounded, that is $|r(s, a)| < \infty$ for all (s, a) pairs, which implies $Var(r(s, a)) < \infty$. □

Theorem 2. *Given an advice model (T, S, π_d), the student S adopts the Sarsa Algorithm and conditions in Lemma 2 all hold, convergence of Sarsa still holds in the advice model setting*

Proof. Notice that the GLIE policy guarantee that each state-action pair is visited infinitely many times. And there is finite advice in our advice model. Therefore, the assumptions still hold in advice model setting. Apply Lemma 2, the convergence result follows. □

Remark 1. *On one hand, the key for the convergence results is that each state-action pair is visited infinitely often. For an advice model, the finite budget does not invalidate the infinite visit assumption. Therefore, the results follows from previous convergence results hold. On the other hand, the infinite visit assumption is a sufficient condition for the convergence result — if the assumption does not hold, the convergence may still hold. Moreover, the algorithms converge even if the budget is infinite as long as the student is still able to visit all state-action pairs infinitely many times.*

Linear Function Approximation

In the previous subsection, we discuss some results regarding tabular representation learning algorithms that require an MDP with finite states and actions at each state. However, infinite or large state-action space in practice is very important since they can characterize many realistic scenarios.

The convergence of Q-learning and Sarsa with linear approximation in standard setting has been proved (Melo, Meyn, and Ribeiro 2008), provided the relevant assumptions

hold. Our approach is inspired by this work, which assumes that the algorithm (Q-learning or Sarsa, with linear approximation) holds under the convergence conditions in (Melo, Meyn, and Ribeiro 2008). We then apply the convergence theorems to the action advice model (T, S, π_d), and the results follow.

We need to define some notations for simplifying our proofs. Given an MDP $M = (S, A, P, R, \gamma)$ with a compact state set S and a fixed policy π, $\mathcal{M} = (S, P_\pi)$ is the Markov chain induced by policy π. Assume that the chain $\mathcal{M}$ is uniformly ergodic with invariant probability measure μ_S over S and the policy π satisfies $\pi(s, a) > 0$ for all $a \in A$ and all $s \in S$ with non-zero μ_S measure[3]. Let μ_π be the probability measure for all Borel-measurable set $U \subset S$ and for all action $a \in A$,

$$\mu_\pi(U \times \{a\}) = \int_U \pi(s, a)\mu_S(ds).$$

Suppose that $\{\phi_i\}_{i=0}^d$ is a set of bounded, linearly independent features, we define matrix Σ_π as

$$\Sigma_\pi = \mathbb{E}[\boldsymbol{\phi}^\intercal(s, a)\boldsymbol{\phi}(s, a)] = \int_{S\times A} \boldsymbol{\phi}^\intercal(s, a)\boldsymbol{\phi}(s, a)d\mu_\pi$$

Notice that Σ_π is independent of the initial probability distribution due to uniform ergodicity.

For a fixed $\boldsymbol{\theta} \in \mathbb{R}^d$, $d > 1$ and a fixed state $s \in S$, define the set of optimal actions in state s as

$$A_{\boldsymbol{\theta},s} = \left\{ a^\star \in A \middle| \boldsymbol{\theta}^\intercal\boldsymbol{\phi}(s, a^\star) = \max_{a\in A} \boldsymbol{\theta}^\intercal\boldsymbol{\phi}(s, a) \right\}.$$

A policy π is greedy w.r.t. $\boldsymbol{\theta}$ which assigns positive probability only to actions in $A_{\boldsymbol{\theta},s}$. We define $\boldsymbol{\theta}$-dependent matrix

$$\Sigma^\star_\pi(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\boldsymbol{\phi}^\intercal(s, a_{\boldsymbol{\theta},s})\boldsymbol{\phi}(s, a_{\boldsymbol{\theta},s})\right],$$

where $a_{\boldsymbol{\theta},s}$ is a random action determined by policy π at state s in set $A_{\boldsymbol{\theta},s}$. Notice that the difference between Σ_π and $\Sigma^\star_\pi(\boldsymbol{\theta})$ is that the actions are taken according to π in Σ_π while in $\Sigma^\star_\pi(\boldsymbol{\theta})$ they are taken greedily w.r.t. a fixed $\boldsymbol{\theta}$, that is, actions in $A_{\boldsymbol{\theta},s}$.

We will show that Q-learning with linear function approximation still converges in the advice model setting at first. We introduce following lemma:

Lemma 3. *((Melo, Meyn, and Ribeiro 2008) Theorem 1) Let M, π and $\{\phi_i\}_{i=0}^d$ be defined as above. if, for all $\boldsymbol{\theta}$, $\boldsymbol{\Sigma}_\pi > \gamma^2\boldsymbol{\Sigma}^\star_\pi(\boldsymbol{\theta})$ and the step-size sequence $\{\alpha_t(s_t, a_t)\}$ such that $\sum_t \alpha_t(s_t, a_t) = \infty, \sum_t \alpha_t(s_t, a_t)^2 < \infty$, then the Algorithm Q-learning with linear approximation converges with probability 1*

Theorem 3. *Given an advice model (T, S, π_d), if the Markov chain which is induced by π_d is also **uniformly ergodic** and the student S adopts the Q-learning with linear approximation and conditions in Lemma 3 all hold, the convergence of Q-learning with linear approximation still hod in the advice model setting.*

[3]This condition is able to be interpreted as the continuous counterpart of "infinite visit" in finite action-state space scenario.

Proof. Apply Lemma 3, the convergence result still hold in the advice model setting. □

Next, we will analyze the convergence of Sarsa with linear approximation in the advice model. Sarsa is an on-policy algorithm, we need some different assumptions. A policy π is ϵ-greedy with respect to a Q-value function Q for a fixed $\boldsymbol{\theta}$, if it chooses a random action with probability $\epsilon > 0$ and a greedy action $a \in A_{\boldsymbol{\theta},s}$ for all state $s \in S$. A $\boldsymbol{\theta}$-dependent policy $\pi_{\boldsymbol{\theta}}$ satisfies $\pi_{\boldsymbol{\theta}}(s, a) > 0$ for all $\boldsymbol{\theta}$. Now we consider a policy $\pi_{\boldsymbol{\theta}_t}$ is ϵ-greedy with respect to $\boldsymbol{\theta}_t^\intercal\boldsymbol{\Phi}(s, a)$ at each time step t and Lipshitz continuous with respect to $\boldsymbol{\theta}$, where K denotes the Lipshitz constant (refers to a specific metric)[4]. Moreover, we assume that induced Markov chain $\mathcal{M} = (S, P_{\boldsymbol{\theta}})$ is uniformly ergodic.

Lemma 4. *((Melo, Meyn, and Ribeiro 2008) Theorem 2) Let M, π_{θ_t} and $\{\phi_i\}_{i=0}^d$ be defined as above. Let K be the Lipshcitz constant of the learning policy π_θ w.r.t. θ. If the step-size sequence $\{\alpha_t(s_t, a_t)\}$ such that $\sum_t \alpha_t(s_t, a_t) = \infty, \sum_t \alpha_t(s_t, a_t)^2 < \infty$, Then, there is $K_0 > 0$ such that, if $K < K_0$, the Sarsa with linear approximation converges with probability 1.*

Theorem 4. *Given an advice model (T, S, π_d), if π_d is $\boldsymbol{\theta}$-dependent and ϵ-greedy w.r.t. a fixed $\boldsymbol{\theta}_t$ at each time step t. The student S adopts the Sarsa with linear approximation and conditions in Lemma 4 all hold, Sarsa with linear approximation still converges with probability 1.*

Proof. Apply Lemma 4, the convergence result still hold in the advice model setting. □

Remark 2. *Notice that we assume the budget of the teacher is finite which implies that any finite policies do not affect the convergence results as long as the conditions in Lemma 1, 2, 3 and 4 still hold. Therefore, the student will eventually converge even if the teacher is sub-optimal.*

Asymptotic Performance

Next, we will investigate the asymptotic behavior in the advice model. Most of convergence results rely on infinite experience, which is not suitable in practice — we first redefine the concept of convergence.

Definition 2 (Convergence in Algorithm Design)**.** *If an algorithm $\mathfrak{A}$ converges, then there exits a $N \in \mathbb{N}$, for all $t \geq N$ such that $||Q_{t+1} - Q_t||_\infty \leq \epsilon$, where ϵ is very small constant.*

Theorem 5. *If an algorithm $\mathfrak{A}$ converges in terms of Definition 2, then finite advice cannot improve the asymptotic performance of algorithms $\mathfrak{A}$.*

[4]Given two metric spaces (X, d_x) and (Y, d_y), where d_X and d_Y denotes metric on set X and Y, respectively. A function $f : X \to Y$ is called **Lipshitz continuous**, if there exists a real constant $K \geq 0$ such that for all $x_1, x_2 \in X$,

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2),$$

where the constant K is called **Lipshitz constant**.

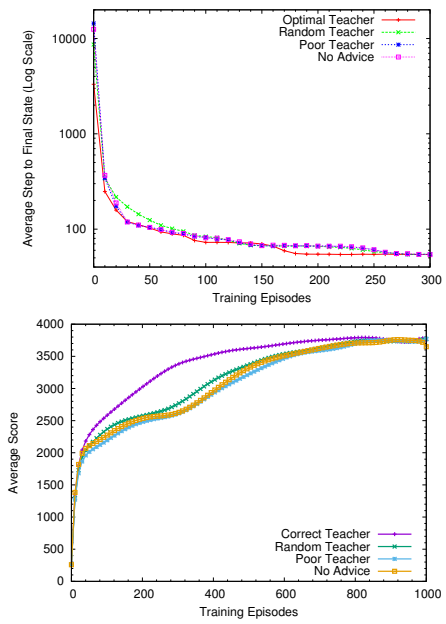

Figure 2: Top: Q-learning students in Linear Chain MDP. Bottom: Sarsa students in Pac-Man domain.

Proof. If an algorithm $\mathfrak{A}$ converges, then there is a $N \in \mathbb{N}$ for all $t \geq N$ such that $||Q_{t+1} - Q_t||_\infty \leq \epsilon$, where ϵ is very small constant. Therefore, even if the advice is sub-optimal the student will always find the optimal action according to its own Q-value after N updates, that is, finite advice can not affect the asymptotic performance in the sense of infinite horizon. The asymptotic performance is determined by the algorithms that the student uses, not the advice provided by a teacher. □

Remark 3. *Theorem 5 indicates the limitation of the advice model. Generally, there are two intuitive methods to improve the performance of student in the advice model: (1) higher amounts of advice, or (2) redistribution of the advice(e.g., delay the advice for when it is most useful). Our theorem points out that, with a finite budget for advice, the asymptotic performance is still determined by the algorithm that the student adopts as long as the algorithm converges. Furthermore, advice delay is limited also due to the convergence of the algorithm that the student uses.*

Experimental Domain and Results

In this section, we introduce the experimental results in two domains. The goal of experiments is to provide **experimental support for convergence proofs** from the previous section, as well as to justify that **action advice improves learning**. The first domain is a simple linear chain of states: Linear Chain. The second is Pac-Man, a well-known arcade game. We will apply Q-learning with tabular learning to the Linear Chain and Sarsa with linear function approximation to Pac-Man.

Group	FR	FR STD	TR	TR STD
Optimal Teacher	-53.99	3.30	-29007.01	1384.31
Random Teacher	-54.56	3.59	-41670.98	2398.87
Poor Teacher	-54.28	3.58	-43964.97	2394.78
No Advice	-54.13	3.221	-42355.24	2660.75

Table 1: FR is the final reward of the last episode, FR STD is final reward's standard deviation, TR is the total reward accumulated reward in all episodes, and TR STD is the standard deviation of total reward.

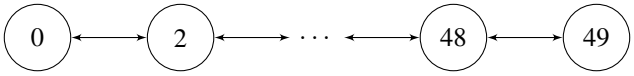

Figure 3: Linear Chain MDP with 50 states. State 0 is the start state and state 49 is the goal state.

Linear Chain MDP

The first experimental domain is the Linear Chain MDP (Lagoudakis and Parr 2003). In this domain, we adopt Q-learning with the tabular representation to store the Q-values due to the simplicity. See Figure 3 for details.

In this paper, the MDP has 50 states and two actions for each state: left and right. state 0 is the start state and state 49 is the final state in which the episode is terminated. The agent will receive -1 reward per step in non-terminated states and 0 in goal state.

To smooth the variance in student performance, we average 300 independent trials of student learning. Each Linear Chain teacher is given an advice budget of $n = 1000$. The reinforcement learning parameters of the students are $\epsilon = 0.1$, $\alpha = 0.9$ and $\gamma = 0.8$.

We use four experimental setting to demonstrate the convergence results:

- Optimal Teacher: The teacher will always give the optimal action in each state, i.e., move right.
- Random Teacher: The teacher will give action advice, 50% move left and 50% move right.
- Poor Teacher: The poor teacher gives the worst action, e.g., move left.
- No Advice: There is no advice, equivalent to normal reinforcement learning.

Figure 2 (top) shows the results of these experiments (note the log scale on the y-axis). All settings converge after 280 episodes training despite different teacher performance.

To compare methods, we calculate the area under each learning curve. We apply one-way ANOVA to test the difference between all settings and the result shows that the $p < 2 \times 10^{-16}$, indicating that we should reject the null hypothesis that "all test groups have same means." Therefore, all experimental settings are statistically different, where the optimal teacher outperforms the random teacher, which outperforms no advice, which outperforms the poor teacher. Also, we provide the final reward, standard deviation of final reward, total reward and standard deviation of final reward on Table 1.

Group	FR	FR STD	TR	TR STD
Correct Teacher	3746.75	192.18	341790.99	5936.23
Random Teacher	3649.78	167.86	313151.06	4634.88
Poor Teacher	3775.13	148.34	307926.03	7708.45
No Advice	3766.58	132.41	318072.70	7660.44

Table 2: FR, FR STD, TR and TR STD are same as those in Table 1.

Pacman

Pac-Man is a famous 1980s arcade game in which the player navigates a maze, trying to earn points by touching edible items and trying to avoid being caught by the four ghosts. We use a JAVA implementation of the game provided by the Ms. Pac-Man vs. Ghosts League (Rohlfshagen and Lucas 2011). This domain is discrete but has a very large state space due to different position combination of player and all ghosts — linear function approximation is used to represent state. Student agents learn the task using Sarsa and a state representation defined by 7 features that count objects at a range of distances, as used (and defined) in (Torrey and Taylor 2013).

To smooth the natural variance in student performance, each learning curve averages 30 independent trials of student learning. While training, an agent pauses every few episodes to perform at least 30 evaluation episodes and record its average performance — graphs show the performance of students when they are 1) not learning and 2) not receiving advice.

Each Pac-Man teacher is given an advice budget of $n = 1000$, which is half the number of the step limit in a single episode. The reinforcement learning parameters of the students are $\epsilon = 0.05$, $\alpha = 0.001$ and $\gamma = 0.999$.

To demonstrate that finite advice can not affect the convergence of students, we adopt different experimental settings:

- Correct Teacher: Provide the (near-)optimal action when it observes the student is about to execute a sub-optimal action.
- Random Teacher: Provide random action suggestion from the set of legal moves.
- Poor Teacher: Advise the student to take the action with the lowest Q-value whenever the student is about to execute a sub-optimal action.
- No Advice: There is no advice, equivalent to normal reinforcement learning.

See the experimental results in Figure 2 (bottom). All settings converges after 900 episodes training despite different teacher performance. As before, a one-way ANOVA is used to test the total reward accumulated by the four different teaching conditions. $p < 4.6 \times 10^{-13}$, showing that all experimental settings are statistically different, and that the correct teacher was better than no advice, which was better than the random teacher, which was better than the poor teacher. Also, we provide rewards on Table 2.

Related Work

This section briefly outline related work in transfer learning in reinforcement domains, online transfer learning in supervised learning, and algorithmic teaching.

Transfer learning in reinforcement domain has been studies recently (Taylor and Stone 2009; Lazaric 2012). Lazaric introduces a transfer learning framework which inspires us to develop the online transfer learning framework. Lazaric's framework is not **online** and it also suggests to use hypothesis for future usage, however it still lacks of **online** design. Lazaric classifies transfer learning in reinforcement domain into three categories: instance transfer, representation transfer and parameter transfer (Lazaric 2012). The action advice model is a method of instance transfer due to explicit action advice (i.e., sample transfer). Lazaric proposed an instance-transfer method which selectively transfers samples on the basis of the similarity between source and target tasks (Lazaric, Restelli, and Bonarini 2008).

Azar, Lazaric, and Brunskill (2013) introduced a model that takes the teacher/advice model as input and a learning reinforcement learning algorithm is able to query the input advice policy as it is necessary. However, their model does not consider the learning reinforcement learning algorithm behavior, which we believe is important in online reinforcement learning.

Zhao and Hoi propose an online transfer learning framework in supervised learning (Zhao and Hoi 2010), aiming to transfer useful knowledge from some source domain to an online learning task on a target domain. They introduce a framework to solve transfer in two different settings. The first is that source tasks share the same domain as target tasks and the second is that the source domain and target domain are different domain.

Finally, a branch in computational learning theory called algorithmic teaching tries to understand teaching in theoretical ways (Balbach and Zeugmann 2009). In algorithmic learning theory, the teacher usually determines a example sequence and teach the sequence to the learner. There are a lot of algorithmic teaching models such as teaching dimension (Goldman and Kearns 1995) and teaching learners with restricted mind changes (Balbach and Zeugmann 2005). However, those models still concentrate on supervised learning. (Cakmak and Lopes 2012) developed a teaching method which is based on algorithm teaching, but their work focuses on one-time optimal teaching sequence computing, which lacks the online setting.

Discussion

This paper proposes an online transfer learning framework. It then characterizes two existing works addressing teaching in reinforcement learning. A theoretical analysis of one of the methods, where teachers provide action advice, lead us to the following conclusions. First, Q-learning and Sarsa converge to the optimal Q-valuewhen there is a finite amount of advice. Second, with linear function approximation, Q-learning and Sarsa converge to the optimal Q-value , assuming normal assumptions hold. Third,there is a limit of the advice model: teacher advice can not affect the asymptotic

performance of any algorithms that converge. Fourth, our results are empirically justified in the Linear Chain MDP and in Pac-Man.

In the future, sample complexity and regret analysis for the advice model will be investigated, now that the convergence results have been established. Additional models under the online transfer framework will be developed, which will not only focus on interaction between machines, but also consider interaction between machines and humans (e.g., learning from demonstration (Argall et al. 2009)). Finally, we will consider other reinforcement learning algorithms such as R-Max and study the theoretical properties of those algorithms in the presence of the advice model.

Acknowledgments

This research has taken place in the Intelligent Robot Learning (IRL) Lab, Washington State University. IRL research is support in part by grants from AFRL FA8750-14-1-0069, AFRL FA8750-14-1-0070, NSF IIS-1149917, NSF IIS-1319412, and USDA 2014-67021-22174.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Azar, M. G.; Lazaric, A.; and Brunskill, E. 2013. Regret bounds for reinforcement learning with policy advice. In *Machine Learning and Knowledge Discovery in Databases*. Springer. 97–112.

Balbach, F. J., and Zeugmann, T. 2005. Teaching learners with restricted mind changes. In *Algorithmic learning theory*, 474–489. Springer.

Balbach, F. J., and Zeugmann, T. 2009. Recent developments in algorithmic teaching. In *Language and Automata Theory and Applications*. Springer. 1–18.

Cakmak, M., and Lopes, M. 2012. Algorithmic and human teaching of sequential decision tasks. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*.

Erez, T., and Smart, W. D. 2008. What does shaping mean for computational reinforcement learning? In *Development and Learning, ICDL 7th IEEE International Conference on*, 215–219. IEEE.

Goldman, S. A., and Kearns, M. J. 1995. On the complexity of teaching. *Journal of Computer and System Sciences* 50(1):20–31.

Jaakkola, T.; Jordan, M. I.; and Singh, S. P. 1994. On the convergence of stochastic iterative dynamic programming algorithms. *Neural computation* 6(6):1185–1201.

Lagoudakis, M. G., and Parr, R. 2003. Least-squares policy iteration. *The Journal of Machine Learning Research* 4:1107–1149.

Lazaric, A.; Restelli, M.; and Bonarini, A. 2008. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, 544–551. ACM.

Lazaric, A. 2012. Transfer in reinforcement learning: A framework and a survey. In Wiering, M., and van Otterlo, M., eds., *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*. Springer Berlin Heidelberg. 143–173.

Melo, F. S.; Meyn, S. P.; and Ribeiro, M. I. 2008. An analysis of reinforcement learning with function approximation. In *Proceedings of the 25th international conference on Machine learning*, 664–671. ACM.

Mihalkova, L., and Mooney, R. J. 2006. Using active relocation to aid reinforcement learning. In *The 19th International Conference of the Florida Artificial Intelligence Research Society*, 580–585.

Mohri, M.; Rostamizadeh, A.; and Talwalkar, A. 2012. *Foundations of machine learning*. MIT press.

Peter, D. 1992. The convergence of td (lambda) for general lambda. *Machine Learning* 8(34):341–362.

Rohlfshagen, P., and Lucas, S. M. 2011. Ms pac-man versus ghost team cec 2011 competition. In *Evolutionary Computation, IEEE Congress on*, 70–77. IEEE.

Rudin, W. 1986. *Real and complex analysis (3rd)*. New York: McGraw-Hill Inc.

Rummery, G. A., and Niranjan, M. 1994. *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering.

Settles, B. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52:55–66.

Singh, S.; Jaakkola, T.; Littman, M. L.; and Szepesvári, C. 2000. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning* 38(3):287–308.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to reinforcement learning*. MIT Press.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research* 10:1633–1685.

Torrey, L., and Taylor, M. 2013. Teaching on a budget: agents advising agents in reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*, 1053–1060. International Foundation for Autonomous Agents and Multiagent Systems.

Tsitsiklis, J. N. 1994. Asynchronous stochastic approximation and q-learning. *Machine Learning* 16(3):185–202.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.

Watkins, C. J. C. H. 1989. *Learning from delayed rewards.* Ph.D. Dissertation, University of Cambridge.

Zhao, P., and Hoi, S. C. 2010. Otl: A framework of online transfer learning. In *Proceedings of the 27th International Conference on Machine Learning*, 1231–1238.

Zimmer, M.; Viappiani, P.; and Weng, P. 2014. Teacher-Student Framework: a Reinforcement Learning Approach. In *AAMAS Workshop Autonomous Robots and Multirobot Systems*.