machine learning & knowledge extraction

MDPI

Article

Learning to Teach Reinforcement Learning Agents

Anestis Fachantidis [1,*], Matthew E. Taylor [2] and Ioannis Vlahavas [1]

[1] Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece; vlahavas@csd.auth.gr
[2] Borealis AI, University of Alberta, CCIS 3-232, Edmonton, AB T6G 2M9, Canada; matthew.taylor@rbc.com
* Correspondence: afa@csd.auth.gr

Received: 19 September 2017; Accepted: 1 December 2017; Published: 6 December 2017

Abstract: In this article, we study the transfer learning model of action advice under a budget. We focus on reinforcement learning teachers providing action advice to heterogeneous students playing the game of Pac-Man under a limited advice budget. First, we examine several critical factors affecting advice quality in this setting, such as the average performance of the teacher, its variance and the importance of reward discounting in advising. The experiments show that the best performers are not always the best teachers and reveal the non-trivial importance of the coefficient of variation (CV) as a statistic for choosing policies that generate advice. The CV statistic relates variance to the corresponding mean. Second, the article studies policy learning for distributing advice under a budget. Whereas most methods in the relevant literature rely on heuristics for advice distribution, we formulate the problem as a learning one and propose a novel reinforcement learning algorithm capable of learning when to advise or not. The proposed algorithm is able to advise even when it does not have knowledge of the student's intended action and needs significantly less training time compared to previous learning approaches. Finally, in this article, we argue that learning to advise under a budget is an instance of a more generic learning problem: Constrained Exploitation Reinforcement Learning.

Keywords: machine learning; reinforcement learning; transfer learning; action advice; machine teaching

1. Introduction

In the reinforcement learning (RL) framework [1], data efficient approaches are especially important for real world and commercial applications, such as robotics. In such domains, extensive interaction with the environment needs time and can be costly.

One data efficient approach for RL is *transfer learning* (TL) [2]. Typically, when an RL agent leverages TL, it uses knowledge acquired in one or more (*source*) tasks to speed up its learning in a more complex (*target*) task. Most realistic TL settings require transfer of knowledge between different tasks or heterogeneous agents that can be vastly different from each other (e.g., humans and software agents).

Transferring between heterogeneous agents is often challenging since most methodologies involve exploiting the agents' structural similarity to transfer knowledge between tasks. As an example, TL can be applied between two similar RL agents, which both use the same function approximation method, by transferring their learned parameters. In such a case, a Q-Value transfer solution could be used, combined with an algorithm constructing mappings between the state variables of the two tasks.

Whereas solutions for extracting similarity between tasks have been extensively studied in the past [2,3], the main problem of transferring between very dissimilar agents (e.g., humans and software agents) remains.

Consider, for example, a game hint system for human players. The game hint system can not directly transfer its internal knowledge to the human player. Moreover, it should transfer knowledge in a limited and prioritized way since the attention span of humans is limited.

The only prominent knowledge transfer unit between all agents (software, mechanical, or biological) is action. Action suggestion (advice) can be understood by very different agents. However, even when transferring using advice, four problems arise: (1) deciding what to advise (production of advice); (2) deciding when to advise (distribution of advice), especially when using a limited advice budget; (3) determining a common action language in order to appropriately express the advice between heterogeneous agents; and (4) communicating the advice effectively, ensuring its timely and noiseless reception.

This article focuses on the first two problems—those of deciding when and what to advise under a budget. Moreover, we use the game of Pac-Man to test our methods' effectiveness in a complex domain.

Whereas works such as [4] provide a formal understanding of RL students receiving advice and the implications on the *student's* learning process (e.g., convergence properties) and papers like [5,6] provide practical methods for a teacher to advise agents, this work attempts a new learning formulation of the problem and proposes a novel learning algorithm based on it. We identify and exploit the similarities of the advising under a budget (AuB) problem to the classic exploration–exploitation problem in RL and identify a sub-class of reinforcement learning problems: Constrained Exploitation Reinforcement Learning.

Most successful methodologies for AuB require students to inform their teacher of their intended action. This is not a realistic requirement in many real-world TL problems, since it assumes one more communication channel between the student and the teacher; thus, it requires some form of structural compliance from the student. An example of how restrictive is this requirement for real-world applications comes from the game hint example system. The system advises the human player for his next action in real-time, but the human player could never be expected to announce its intended action beforehand. Part of this work's goal is also to alleviate such a prerequisite and propose methods that can also work without such knowledge.

Specifically, the contributions of this article are (a) an empirical study on determining an appropriate advising policy in the game of Pac-Man; (b) a novel application of average reward reinforcement learning to produce advice; (c) a novel formulation of the learning to advise under budget (AuB) problem as a problem of constrained exploitation RL; and (d) a novel RL algorithm for learning a teaching policy to distribute advice, able to train faster (lower data complexity) than previous learning approaches and advise even when not having knowledge of the student's intended action.

The remainder of this article is organized as follows. Section 2 presents background information on RL and TL, introduces some key concepts of the learning to teach problem and describes the game of Pac-Man used for the experiments. Section 3 formally defines the teaching task and formulates it as a learning problem. In Section 4, we study the critical factors for policies used to *produce* advice, whereas, in Section 5, we discuss the problem of *distributing* advice and propose a novel learning algorithm for it. Finally, in Sections 6 and 7, we describe related work, draw the conclusions of the article and propose future directions.

2. Background

This section provides an introduction to reinforcement learning and transfer learning with an emphasis in advising methodologies. In addition, Section 2.3 describes the experimental domain of this work, which is the Pac-Man video game.

2.1. Reinforcement Learning

Reinforcement Learning considers an agent acting in a dynamic environment and learning a behaviour through trial-and-error interactions with it [1]. At each time step, the RL agent observes

the environment's state, $s \in S$, where S can simply be a finite set of possible states, and then selects an action $a \in A(s)$ to execute, where $A(s)$ is the set of possible actions in state s. The agent receives a reward, $r \in \mathbb{R}$ from the environment, and observes its new state $s' \in S$, according to the transition function, T, of the environment with $T(s, a, s') = P(s'|s, a)$. The goal of the agent is to learn an action policy, $\pi : S \rightarrow A$ that will maximize its expected return, G, which is is a cumulative function of the reward sequence given also a discounting parameter, γ. The γ parameter, where $0 \leq \gamma < 1$, controls the importance of short-term rewards over the most long-term ones, discounting the later by powers of factor of γ.

An RL policy can be expressed through an action–value function, $Q^{\pi}(s, a)$, which is the expected return starting from s, taking action a, and following after that policy π. A policy that maximizes the value function in each state is called an optimal policy, π^* and the respective optimal value function is Q^*.

Probably the most well-known RL algorithm, Q-Learning [7], estimates Q^* directly (off-policy) by executing in each time step t the following update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)],$$

where $\alpha : 0 \leq \alpha \leq 1$ is the learning rate parameter adjusting the extent of value correction in each time step towards the new estimation.

The RL agent has to *explore* by trying different actions in different parts of the state space in order to discover more rewarding policies. However, the agent should also *exploit* by applying greedy action selection in order to harness the learned policy. The most simple exploration method is *ϵ-greedy* action selection where the agent takes its current best action with probability $(1 - \epsilon)$ and a random action with probability ϵ. This simple strategy is often quite effective, especially in simple RL tasks. For a more in-depth review of RL, we refer the reader to [1].

2.2. Transfer Learning and Advising under a Budget

Transfer Learning [2] refers to the process of using knowledge that has been acquired in a previously learned task, the *source task*, in order to enhance the learning procedure in a new and more complex task, the *target task*. The more similar these two tasks are, the easier it is to transfer knowledge between them. By similarity, we mean the similarity of their underlying Markov Decision Processes (MDP), that is, the transition and reward functions of the two tasks and also their state and action spaces.

The type of knowledge that can be transferred between tasks varies among different TL methods, including value functions [8], entire policies [9], actions (policy advice) [10], or a set of samples from a source task that can be used by a model-based RL algorithm in a target task [11].

Focusing specifically on policy advice under an advice budget constraint, we identify two aspects of the problem, (a) learning a policy to produce advice and (b) distributing the advice in the most appropriate way, while respecting the advice budget constraint. Most methods in the literature produce advice by greedily using a learned policy for the task in hand [4–6]. For advice distribution, most methods rely on some form of heuristic function (and not learning) based on which the teacher decides when to give advice. Examples of such methods are Importance Advice and Mistake Correcting [6].

The Importance Advice method produces advice by repeatedly querying a learned policy's value function, on each state the student faces, to obtain the best action for that state. Distribution of advice, that is deciding when to advise or not, is determined by a heuristic logical expression of the form $Q_{max}(s, a) - Q_{min}(s, a) > t$ where t is a threshold parameter determining the state-action value gap between the best and the worst action for that state. If this value gap exceeds the threshold value, t, the state is considered critical and advice is given. The algorithm continues until the advice budget finishes.

Mistake correcting (MC) [6] differs from Importance Advising only in presuming knowledge of the student's intended action. Consequently, it validates the Importance Advising criterion only if the student action is wrong, not wasting advice when the student does not need it.

The method presented in [5] (Zimmer's method) formulates the teaching problem as an RL one in order to learn an advice distribution policy. The teacher agent has an action set with two actions, $A = \{\text{advice, no advice}\}$. The teacher's state space is an augmented version of the student's one and is of the form: $s_{teacher} = (s_{student}, a_{student}, b, n_e)$, where $s_{student}$ is the current state vector of the student, $a_{student}$ is the intended action of the student (this method assumes that the student announces the intended action on every step), b the remaining advice budget and n_e the student's training episode number. Moreover, the teacher's reward signal is a transformed version of the student's reward with an extra positive reward in case the student reaches its goal in a small number of steps. We note that this method is tested only on the Mountain Car domain and the reward signal proposed for the teacher is domain-dependent.

The policy advice methods [4–6] presented in this section will also be used for comparisons in the experiments presented later in this article.

2.3. Pac-Man

Pac-Man is a famous 1980s arcade game in which the player navigates a maze like the one in Figure 1, trying to earn points by touching edible items while also trying to avoid being caught by four different types of ghosts. In our setting, ghosts will chase the player 80% of the time and choose actions randomly 20%.

Figure 1. The Pac-Man arcade game.

The small dots on the grid represent food pellets and are worth 10 points each. The larger ones are power pellets, which are worth 50 points each and getting those makes the ghosts edible for a short time, during which they slow down and flee the player. Eating a ghost is worth 200 points and respawns it in the lair at the center of the maze. An episode ends if any ghost catches Pac-Man, or after 2000 steps.

The player's action space has four actions—move up, down, left, and right—but some of the actions can occasionally be unavailable (e.g., due to maze walls). Although discrete, this domain has a very large state space with 1293 distinct locations in the maze, with a complete state consisting of the locations of Pac-Man, the ghosts, the food pellets, and the power pills, along with the trajectory and the state of each ghost (i.e., edible or not). The combinatorial explosion of possible states makes it essential to construct high-level state features and use Q-function approximation.

In this article, we follow previous work [6] that used a high-level feature set (high-asymptote feature set) consisting of seven action-specific features that count objects at a range of distances

from Pac-Man. For a detailed description of these features, we refer the reader to [6]. When using action-specific features, a feature set is really a set of functions $\{f_1(s,a), f_2(s,a), ...\}$. All actions share one Q-function, which associates a weight with each feature. A Q-value is $Q(s,a) = w_0 + \sum_i w_i f_i(s,a)$. To achieve gradient-descent convergence, it is important to have the extra bias weight w_0 and also to normalize the features to the range $[0,1]$.

A perfect score in an episode is 5600 points, which is quite difficult to achieve for both human and agent players. An agent executing random actions earns an average of 250 points. The 7-feature set allows an agent to learn to catch some edible ghosts and achieve a per-episode average of 3800 points.

The experiments of this article use a JAVA implementation of the game provided by the Ms. Pac-Man vs. Ghosts League [12], which conducts annual competitions.

3. The Teaching Task

In this section, we attempt a more formal understanding of teaching tasks that are based on action advice. The necessary notation is presented in Table 1.

Table 1. The notation used in this article.

Notion † \ Agent	Student (Acting)	Teacher (Acting)	Teacher (Advising)
Index used	NONE	Σ	T
MDP	M	M_Σ *	M_T
Action Set	A	A_Σ	A_T
State Space	S	S_Σ	S_T
Reward	R	R_Σ •	R_T
Value Function	Q	Q_Σ	Q_T
Policy	π	π_Σ	π_T
Agent Goal	L	L_Σ •	L_T

† A teacher agent may not have a teaching value function Q_T, relying in a hand-coded or heuristic teaching policy; * If the teacher has learned to act in the same Markov Decision Process (MDP) as the student, $M_\Sigma = M$; • In this work, we assume $R_\Sigma = R$ and $L_\Sigma = L$. All agents acting in the task have the same rewards and goals.

3.1. Definitions

Definition 1 (Student). *A student agent is an agent acting in an environment and capable of accepting advice from another agent.*

Definition 2 (Teacher). *A teacher agent is an agent capable of executing and informing a teaching policy (see Definition 7) to provide action advice to a student agent acting for a specific task.*

Definition 3 (Acting Task). *The acting task is the task for which the teacher gives advice and can be defined as an MDP of the form $M = \langle S, A, T, R, \gamma \rangle$ on an environment E.*

Definition 4 (Teaching Task). *The teaching task is the task of providing action advice to a student agent to assist him in learning faster or learning better the acting task. Any teaching task is accompanied by a finite advice budget, B.*

Definition 5 (Teaching Action Space). *Given the action space A of the acting task, the action space of the teacher in timestep t is:*

$$A_T = \begin{cases} \{a, \perp\} & , \quad b_t > 0, \\ \{\perp\} & , \quad b_t \leq 0, \end{cases}$$

where $a \in A$ an action of the acting task given as advice and the no advice action, $\perp$, meaning that the teacher will not give advice in this step, allowing the student to act on its own. b_t is the advice budget left in time-step t.

Definition 6 (**Teaching State Space**). *The teacher agent state space in timestep t has the following form:*

$$S_T = \langle \Theta_t, b_t, Q_\Sigma \rangle, \tag{1}$$

where, on timestep t, b_t is the remaining advice budget, Q_Σ is the teacher's acting value function or any structure representing its policy, and Θ is a tuple containing any knowledge we can have for the student and its MDP.

For example, if the student's MDP is $M_\Sigma = \langle S, A, T, R, \gamma \rangle$ and the teacher observes the current state of the student, $s_t \in S$, reward, $r_t \in R$, and action $a_t \in A$, then $\Theta_t = \langle s_t, r_t, a_t \rangle$.

Definition 7 (**Teaching Policy**). *A teaching policy, π_T, is a deterministic policy of the form:*

$$\pi_T : S_T \to A_T, \tag{2}$$

where S_T and A_T are the teaching state and action spaces respectively (see Definitions 5 and 6).

The teaching policy, π_T, actually transforms the acting policy, π_Σ, of an actor agent (expressed through its respective state-action value function, Q_Σ), to a policy producing advice under budget. Such a teaching policy will usually [4–6] set $a = \arg\max_a(Q_\Sigma(s, a))$, which means that the teaching policy is greedy with respect to the acting value function, Q_Σ.

As a minimal example of the proposed formulation, the Importance Advising method [6] which uses the state importance criterion (see Section 2.2) can be said to use a teaching state space, $S_T = \langle \Theta = s_t, b_t, Q_\Sigma \rangle$ as it requires knowledge only about the current state, s_t, of the student, the remaining advice budget, b_t, and an acting value function, Q_Σ, from which it produces advice.

Finally, and concerning Definitions 1 and 3, in this work we assume that a student agent always follows the given advice and that γ is part of the agent since it can be chosen and is used only for the agent's value estimation and not for its performance evaluation, which is based only on externally provided game scores.

3.2. Learning to Teach

The definitions presented in Subsection 3.1 apply to any teacher agent even if it advises based on a heuristic function. In the following, we focus on teachers that use RL to *learn* a teaching policy (i.e., advice distribution policy).

In its most simplified version, the learning to teach task employs two agents: the teacher and the student. In the *first learning phase*, a teacher agent has the role of the actor: it learns the acting task alone. It observes a state space S_Σ and has an action set A_Σ. Based on a reward signal R_Σ received from the environment, it learns a policy π_Σ to achieve the acting task goal L_Σ. In our context, this first learning phase can be seen as the *advice production* phase since the teacher learns the policy that will be used to advise a student later on.

At any time-step t, the teacher agent may have to stop acting and a new agent, *the student*, enters the acting task and the corresponding environment.

Consequently, the teacher agent has to now *learn and use a teaching policy* for the specific task to achieve the teaching goal, L_T. Additionally to the definitions given in Section 3, this *second learning phase* (learning a teaching policy) requires the identification and formulation of the following:

- Return horizon. Even if the teaching task is formulated as an episodic one, the teaching episode, also referred to as a session, is not necessarily matching the student's learning episode. The teacher's episode scope is greater and could track several learning episodes of the student.

- Reward signal. A different return horizon implies a different task goal and consecutively the teacher's reward signal can be different from the student's (e.g., encouraging more the learning progress of the student over its absolute learning performance).

Moreover, considering the teacher's state space as a superset of the student's state space (see Definition 6) reveals one more difficulty of the learning to teach task. From the teacher's point of view, the student can be considered a time-inhomogeneous Markov-Chain (MC) [13], $X = (X_t : t \geq 0)$. This is because the transition matrix P_t of the student's MC is dependent on time, since the student agent is learning and constantly changing its policy over time. The time inhomogeneity of this MC poses significant difficulties in handling the problem theoretically. Homogenizing this MC by defining it as a space-time MC, (X_t, t) can make practical solutions, such as those presented in this article, feasible but still theoretical treatment is difficult (e.g., no stationary distributions exist in this case) and is beyond the scope of this article.

In general, every *learning task* can have its corresponding *teaching task*, which could be thought as its dual. As learning to act in a specific task and teaching that task can be considered different tasks, they have their own goals and, consequently, are "described" by different reward signals.

As an example, in [5], a teacher agent for the mountain car domain has a different reward signal to that of the student, encouraging teaching policies that help the student reach its goal sooner.

Learning a teaching policy, as this is described above, could be modelled by many different types of Markov Processes. However, none of the classic MDP formulations completely models the specific learning problem as a whole either by not handling the non-stationarity of the problem or by not handling the specific budget constraint imposed on the advising action. This fact is the main motivation of Section 5, where we present our proposed method for learning teaching policies.

4. Learning to Produce Advice

In this section, we focus on the advice itself and its production (not its distribution). The main challenge in producing advice based on the Q-Values of an RL value function is that these values are valid only if the policy they represent is fully followed, not when this policy is sparingly sampled to produce advice.

Based on previous methods in the literature (see Section 6), the most common teacher's criterion for selecting which action to advise is $\pi_\Sigma(s) = argmax_a Q_\Sigma(s, a)$, that is, greedy selection of the best action based on the teacher's acting value function. However, the value of $Q_\Sigma(s, a)$ is not correct under the advising scenario since it is accurate only if the student *will continue following teacher's acting policy,* π_Σ *thereafter*. Unfortunately, this is usually not the case in our context since the student, after receiving advice, will often continue for a long period using its own policy exclusively. Even worse, in the early training phases—when advice is needed the most—the student's policy will be vastly different from the teacher's.

This realization is even more important if we consider how different the teacher and the student agents are allowed to be in our context. Consider a human student receiving advice in the game of Pac-Man. Human players often play fast-paced action games in a myopic and reactive manner, seeking short-term survival and not a long-term strategic advantage.

In that case, a human student infrequently advised by a policy learned using a high γ value close to 1 will often be mislead to locally sub-optimal actions because these actions may be highly valued for the teacher's far-sighted policy. The human player will probably not follow such a policy thereafter and he has therefore been misled to an action that would be useful only if he would also follow the rest of the teacher's acting policy too.

Ideally, we would like to use a teacher's acting policy that would be mostly *invariant* to the student's particularities. Such a teacher's policy would advise actions that are good on average, whatever policy is followed thereafter by the student and whatever the student's internal architecture, processes and parameters are (e.g., γ).

In this article, we propose that the above considerations should affect the way we learn policies intended for teachers. Selecting a specific policy for advising, the RL algorithm producing it and its parameters, form a *model selection problem* for RL teachers.

4.1. Model Selection for Teachers

In this section, we want to investigate how factors such as the teacher's γ value (see Section 2.1) influence advice quality for students that can possibly have very different characteristics (e.g, a myopic student and far-sighted teacher). This is important in order to understand which teacher–agent differences affect the teaching performance the most.

To assess the influence of the γ value in the teaching process, we experiment using an RL algorithm like R-Learning [1,14], which does not use a γ value for the calculation of state-action values and relies on estimating the average reward received by the agent, using its policy from any state and thereafter.

Specifically, R-Learning is an infinite-horizon RL algorithm where a different optimality criterion is used such that the value $Q(s, a)$ given action a and state s under policy π is defined as the expectation:

$$Q(s,a) = \sum_{k=1}^{\infty} E_{\pi}\{r_{t+k} - \rho^{\pi}\}, \tag{3}$$

where ρ^{π} is the *average* expected reward per time step under policy π. The intuition behind R-Learning is that, in the long run, the average reward obtained by a specific policy is the same, but some state-action pairs receive better-than-average rewards for a while, while others may receive worse-than-average rewards. This transient, the difference to the average reward received, ρ^{π}, is what defines the state-action value. The R-Learning algorithm makes the following two updates at every step:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r - \rho + max_{a'}Q(s',a') - Q(s,a)] \tag{4}$$

and if $Q(s,a) = max_a Q(s,a)$, then:

$$\rho \leftarrow \rho + \beta[r - \rho + max'_a Q(s',a') - max_a Q(s,a)], \tag{5}$$

where the second update, Equation (5), keeps a running estimate of the average reward, ρ, and β is the learning rate of that update. Using R-Learning to learn a teacher's acting policy along with the rest of the experiments presented in Section 4.2, we can assess the importance of the γ value and the γ value mismatch between student and teacher. Moreover, we assess other factors that possibly influence the quality of advice, such as the performance of the teacher in the acting task, its performance variance, and a possible relation of its average *TD error* [1], with the quality of advising.

As defined in [1], the TD error, δ_t, represents the value estimation error of a value function for a specific state s and action a in time t. For the Q-Learning [7] algorithm, this is:

$$\delta_t = r + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t). \tag{6}$$

This is also part of the Q-Learning update rule. Furthermore, by dividing Equation (6) with the previous value estimation, $Q(s,a)$, we get the percentage of error in relation with it, which we can call TD error percentage:

$$\delta_t^{\%} = \delta_t / Q(s_t, a_t), \tag{7}$$

where $Q(s_t, a_t) \neq 0$. In our context, when the teacher uses an acting policy to produce advice it can still compute, for each student's experience, its own TD error just as it would do if it was actually making a learning update. In the same context, we can intuitively say that $\delta_t^{\%}$ represents the *teacher's surprise* on its new estimation of a state-action value. Please note that this definition of surprise, although similar, is different to that presented in [15], which normalizes for different learners and not for different state-action pairs.

Consequently, a candidate teacher with high average TD error percentage, $\overline{\delta^{\%}}$, is a teacher with more unreliable value estimation, and, therefore, it can be less suitable to become one since its action suggestions are probably based on a non-converged value function.

4.2. Experiments and Results

Based on the discussion in the previous section (Section 4.1), the main goal of the following experiments is to discover how the teacher's policy parameters (such as γ) affect the quality of advice and how different student parameters affect teaching performance. The experimental design is as follows. In the first phase, we created γ-specific teachers by training five Q-Learning agents and one R-Learning agent for 1000 episodes. The Q-Learning agents had all the same parameters, except γ, which took values in $\{0.05, 0.2, 0.6, 0.9, 0.999\}$. The rest of their parameters were the same and fixed—specifically, $\epsilon = 0.05$ and $\alpha = 0.001$ (consistent with previous work [6]). The λ parameter accounting for the decay of the eligibility traces was set to zero (i.e., no eligibility traces) so that the effect of experimentally controlling the γ parameter is isolated. Finally, the parameter β of R-Learning (see Equation (5)) was set to 0.0001 (preliminary results found it produced good results in Pac-Man).

After training for 1000 episodes, the γ-specific Q-Learning teachers and the R-Learning teacher were evaluated on 500 episodes of acting alone in the environment. We calculated their average episode score and the coefficient of variation of these scores as both being possible determining factors of advice quality. The coefficient of variation (CV) was used as a measure of score discrepancy as it shows the extent of variability in relation to the mean of the score (μ), allowing a more clear comparison of variance between methods with different average performance. It is a unit-less measure calculated as $CV = \frac{SD}{\mu}$.

In Table 2, we can see their average episode score on 500 episodes along with the coefficient of variation of that score. R-Learning had significantly worse average *acting* performance than all versions of Q-Learning. Interestingly, episodic Q-Learning (with γ close to 1) did not perform as well as expected. Moreover, a very low γ value (0.05) came up second, showing that a myopic RL agent can perform well in Pac-Man. This result indicates the highly stochastic nature of the game where reactive short-sighted strategies, based more on survival, can perform better than far-sighted strategies.

Table 2. Average Score, Coefficient of Variation (CV) and Standard Deviation (SD) of γ-specific Q-Learning agents and R-Learning, acting alone in 500 episodes, ordered by score (denoted with the symbol ▼).

	Gamma	**Average Episode Score ▼**	**Coefficient of Variation**	**Standard Deviation**
Q-Learning	0.9	3633.78	0.33	1189.89
	0.05	2754.48	0.44	1132.77
	0.2	2668.36	0.47	1254.13
	0.999	2608.04	0.28	730.25
	0.6	2585.26	0.48	1240.92
R-Learning	-	2493.17	0.28	698.09

After the initial training and the evaluation of the acting policies they learned, these agents could be used as teachers for tabula-rasa student agents. In these experiments, we used a simple fixed teaching-advising policy called Every-4-Steps for all teachers since we focus only on the quality of the advice itself and not on the quality of its distribution to the student (teaching policy).

In the Every-4-Steps teaching policy, the teacher gives one piece of advice to the student every four steps. Using this fixed advising policy, we can test and compare the efficacy of the advice when this is not given consecutively, thus testing how useful the advice is when the student does not take a complete policy trajectory from the teacher, but has to use its own policy in between.

Using the teaching policy Every-4-Steps and a budget of $B = 1000$ advice, we ran 30 trials of advising learning students for *each* γ-specific teacher–student pair. Specifically, the γ parameters

of these teacher–student pairs come from the Cartesian product $\{0.05, 0.2, 0.6, 0.9, 0.999, -\} \times \{0.05, 0.2, 0.6, 0.9, 0.999\}$ (30 pairs), where the R-Learning teacher in the first set is denoted with a "-" since it does not have a γ value.

In Figure 2, we can see the average performance of each teacher–student pair compared to the same student not receiving advice at all. Combining these results with Table 2 of the teachers' performance when they were acting alone, we can see that *the best performer is not the best teacher*, with *best* defined as the best average score when acting alone in the task. The best example of this is R-Learning whose average score was worse than any γ-specific Q-Learning agent; however, as we can see in Figure 2, it is almost as good of a teacher as the $\gamma = 0.999$ Q-Learning teacher. R-Learning advising improved all students' scores whatever their γ value, while not resulting in a negative transfer for any of them.

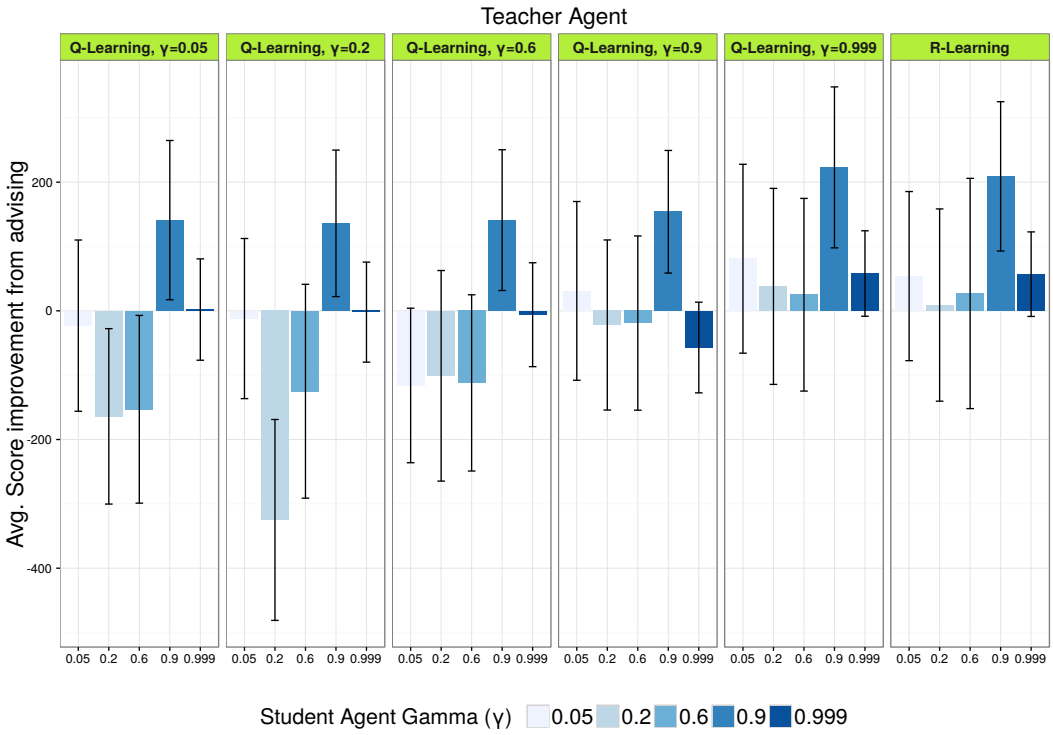


Figure 2. Average score improvement from 30 trials of each teacher–student pair compared to no-transfer (no advising). Negative bars indicate negative transfer (average score decrease). Error bars indicate 95% confidence intervals (CI) of the means. Non-overlapping CIs indicate statistically significant differences of the means, whereas overlapping CIs are inconclusive (for the non-conclusiveness of overlapping confidence intervals, a simple and intuitive explanation can be found in [16]).

Moreover, we can see a pattern where the lower the coefficient of variation (CV) for the acting performance is, the better the teacher, indicating that CV can be an important criteria in model selection for teachers. This is non-trivial since average agent performance (and not its variance) is the dominant model selection criteria adopted in most of the relevant literature in RL. Performance variance expressed by CV seems especially important in our context, that of sparse advising, where the advice should be good whatever the next actions of the student will be.

Based on the results presented here, we can not observe any particular pattern relating teaching performance with the γ values of a teacher-student pair. Interestingly though a $\gamma = 0.999$ teacher is not the most helpful for a $\gamma = 0.999$ student. Even more, a $\gamma = 0.2$ for a $\gamma = 0.2$ student results in significant negative transfer. The teacher with the episodic γ value, $\gamma = 0.999$ and the no discounting

R-Learning one were the most helpful to all students showing that R-Learning can perform well in settings where the student's γ is unknown or varying, such as in the case of human students.

Having identified the possible use of R-Learning for producing acting policies suitable for advising and the importance of performance CV to model selection, we conducted one more experiment between identical teachers.

Specifically, we independently trained 30 Q-Learning teachers with the same parameters, feature sets and characteristics for 1000 episodes. Due to their different experiences and the stochasticity of the game, they naturally learned different policies (i.e., final feature weights in their function approximators). Then, the trained teachers played alone for 500 episodes and we recorded their average performance, average performance variance as also their average TD error percentage, $\overline{\delta^{\%}}$, as this was defined in Section 4.1. We then used the Every-4-step teaching policy with each one of them advising a standard Sarsa [17] student who would learn the task for 1000 episodes. Finally, we recorded the student's average score.

In Figure 3, we can see a correlation plot of the factors mentioned above using a one-tailed non-parametric Spearman correlation test at $p < 0.05$. Confirming the previous results, we can see the negative and statistically significant relation of CV to teaching performance with a correlation coefficient, $r_c = -0.3$. Acting performance also has a medium and positive correlation of $r_c = 0.3$ with teaching performance (student's score) but it is statistically insignificant on the limit. By weighing average performance in its calculation, CV has a stronger relation to teaching performance than standard statistic variance. Moreover, we see that teacher's surprise, $\overline{\delta^{\%}}$ relates strongly ($r_c = -0.66$) and negatively to the acting performance of the teacher and not to its teaching performance ($r_c < -0.05$).

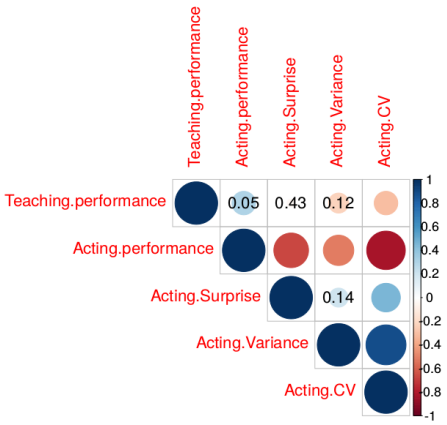


Figure 3. Correlation plot of critical factors influencing teaching performance for 30 Q-Learning teachers. The circle size indicates correlation strength and the color its sign (direction). Statistically significant correlations are at $p < 0.05$. Insignificant correlations are marked with their p-value.

5. Learning to Distribute Advice

In this section, we change focus from advice production to advice distribution: learning a teaching policy in order to most effectively distribute the advice budget.

5.1. Constrained Exploitation Reinforcement Learning

We attempt a more natural formulation of the AuB learning problem described in Section 3.2 by identifying it as an instance of a more generic reinforcement learning problem. This RL problem can be simply described as learning control with constraints imposed on the *exploitation* ability of the learning agent. These constraints can either be a finite number of times the agent can exploit using its policy, possibly states where it is only allowed to explore, or even perhaps a task where it is costly to have

access to an optimal policy and we are allowed to use it only for a limited number of times. How does this RL problem relate to the learning to teach problem? The first insight is that the advise/not advise decision problem has a striking resemblance to the core exploration–exploitation problem of RL agents. Consider the learning to teach problem. We can view the problem as follows: when the teacher agent is advising, it is actually acting on the environment; this is because an obedient student agent will always apply its advice, thus becoming a deterministic actuator for the teacher. In the case of a non-obedient student, the teacher could be said to be using a stochastic actuator.

Consequently, we can view the teacher agent as an *acting* agent using a student agent as its actuator for the environment. Moreover, the teacher is acting greedily by advising his/her best action; thus, he/she exploits. Under this perspective, with advice seen as action, how could we view the no advice action of a teacher? The no advice action can be seen as "trusting" the student to control the environment autonomously. Thus, choosing not to advise in a specific state can be seen as denoting that state to be non-critical with respect to the remaining advice budget and the student's learning progress, or denoting a lack of teacher's knowledge for that state. From the teacher's point of view, not advising can be seen as an exploration action. Thus, controlling when *not to advise* can be seen as a directed exploration problem in MDPs. Imposing a budget constraint, which is a constraint on the number of times a teacher agent can advise (i.e., exploit) is a problem of *constrained and directed exploitation*.

We will consider a simple and motivating example of such a domain. In a grid world 10×10, a robot learns an optimal path towards a rewarding goal state while it should keep away from a specific damaging state. The robot is semi-autonomous; it can either control itself using its own policy or it can be teleoperated for a specific limited number of times. For the robot's operator, what is an optimal use of this finite number of control interventions? What are the states that it would be best to control the robot directly, leaving control of the rest to the robot?

Similarly to the previous example, learning and executing advising policies in a game can be another example of the constrained exploitation problem, which is also the main focus of this article. For example, in a video game like Pac-Man, a game hints system plays the role of the external optimal controller with a limited intervention budget. Such a hint system could suggest actions to human players—when these are most necessary—depending also on the player's policy.

In the rest of this section, we use the term exploitation where one can think of advising and the term exploration when not-advising, focusing on the broader learning problem.

5.2. Learning Constrained Exploitation Policies

Formulating the constrained exploitation task as a reinforcement learning problem itself first requires defining a horizon for the returns. This horizon should be different from that of the actual underlying task (e.g., Pac-Man) because (a) if the underlying task is episodic, then the scope of an exploration–exploitation policy is naturally greater than that and spans across many episodes of the learning agent; (b) if the underlying task is continuing or requires several training episodes for the student, the exploration–exploitation policy may have to be evaluated in a shorter (finite) horizon (e.g., for the first x training episodes). The importance of exploration is usually limited in the late episode(s) where the student may have already converged to a policy. A teaching policy should be primarily evaluated for a training period where advice still matters.

Concerning the return horizon of a constrained exploitation task (and similarly to [4] but in a different perspective), we propose algorithmic convergence [4] as a suitable *stopping criterion* when learning an exploration–exploitation policy. This defines a meaningful horizon for exploration–exploitation tasks since their goal is completed exactly then, not in the end of an episode and not in the continuous execution of an RL algorithm—after convergence—where exploration may not affect the underlying policy any more. We proceed by defining the Convergence Horizon Return.

Definition 8 (Convergence Horizon Return). *Let G be the return of the rewards r_t received by an exploration–exploitation policy, Q the value function of the underlying MDP and $\epsilon \in \mathbb{R}$ a small constant then:*

$$G = \sum_{t=0}^{T} r_t, \tag{8}$$

which, for the time step T, applies:

$$|Q_{T+1} - Q_T| - \epsilon \leq 0. \tag{9}$$

Given a small constant ϵ and the algorithmic convergence of the RL algorithm learning in the *underlying* MDP, the quantity $|Q_{t+1} - Q_t| - \epsilon \xrightarrow[t\to\infty]{} 0$. The algorithmic convergence will be realized either if the learning rate α is discounted or if some temporal difference Δ_t of the underlying algorithm tends to ϵ.

Using the convergence horizon for the return of a teaching task too, the next question can be what are the rewards r_t constituting the return of a teaching task.

One possible goal for any teacher advising with a finite amount of advice would be to help minimize *student's regret* with respect to the reward obtained by an optimal policy. However, since we do not assume such knowledge, and because there is a finite amount of advice, a better goal could be to advise based on the state-action *value* of the advised action and not its immediate *reward*. If the student was able to follow the rest of the teacher's policy after receiving advice, then the action $a = argmax_a(Q_\Sigma(s, a))$ for the current state s would be the best possible. Consequently, we define the notion of *value regret*.

Definition 9 (Value Regret). *In a convergence horizon T, the value regret, R^V of an exploration–exploitation policy (i.e., teaching policy) with respect to both an acting policy π^* obtained after the T period and an acting policy (i.e., student's policy), π^t, in time step t is:*

$$R^V = \sum_{t \in T} [max_a Q^*(s_t, a) - Q^*(s_t, \pi^t(s_t))], \tag{10}$$

where Q^* denotes the corresponding value function of π^*.

The intuition behind this definition of regret in our context (where the acting agent is the student) is that the best teacher for any specific student would ideally be the *student himself*, when it would have reached convergence or its near-optimal policy.

The important thing to note here is that because a student agent receives a finite amount of advice, he/she cannot improve their asymptotic performance [4]; consequently, the evaluation of a teaching policy should ideally be based on the student's optimal policy and not to that of some probably very different teacher because that is its *sustainable optimality*.

For example, consider two states in a teacher's acting MDP, A and B. A student agent learning with a very simplistic state representation may observe these states as just one, C, and not differentiate between them. Then, the student's optimal action in state C will have a different expected return than that obtained by the teacher from either A or B. Its sustainable optimality is defined as to what is optimal given its simplistic internal representation. Any advice based on a finer representation may not be supported with consistency by the student in the long run. A teaching policy should be ideally evaluated on how much it speeds up the student converging to its own optimal policy.

In the next section, we propose a reward signal for teachers based on Value Regret.

5.3. The Q-Teaching Algorithm

The Q-Teaching algorithm described and proposed in this section is an RL advising (teaching) algorithm learning a teaching policy. For this, we propose a novel reward scheme for the teacher based on the value regret (see Definition 9).

The key insight of the method is that of rewarding a teaching policy with quantities of the form $max_a Q^*(s_t, a) - Q^*(s_t, \pi^t(s_t))$, where $\pi^t(s_t)$ is an estimation of the student's action in s_t and $max_a Q^*(s_t, a)$ is the teacher's greedy action in s_t (i.e., the action used for advice). This reward has a high value when the value of the greedy action is significantly higher than the value of the action that the student would take. This means that the teacher is encouraged to advise when the advised action is significantly better than the action the student would take.

For terms of efficiency and to emphasize the value impact of the advising action, Q-Teaching rewards all no-advice actions with zero. The advantages of such a scheme is that the teacher's cumulative reward is based only on the value gain produced when advising and a teaching episode can finish when the budget finishes, not having to observe all the student's episodes after its budget finishes. From preliminary experiments, rewarding no advice actions too (which occur significantly more than the maximum B advice actions) was overpowering the advice actions, resulting in an imbalanced expression of the two actions in the teaching value function.

Still, when advising, the teacher should estimate $Q^*(s_t, \pi^t(s_t))$ in order to compute its reward. The simplest solution is that, since we do not have access to the value function of the student or its internals, we use the acting value function Q_Σ of the teacher as an approximation for the optimal value function of the student, Q^*. To estimate $\pi^t(s_t)$, the teacher has several options. If the teacher is notified of the intended action of the student beforehand, it can use that to compute the reward. If we assume no knowledge of the student's intended action, then some other estimation method for the student's intended action should be used. An example of such an estimation method is used in the Predictive Advice method [6].

While predicting the actual student's action ($\pi^t(s_t)$) is possible, there are other—simpler—choices for this estimation too. For example, the Importance Advising (see Section 2.2) uses a very similar quantity for the advising threshold, of the form $max_a Q^*(s_t, a) - min_a Q^*(s_t, a)$. For Importance Advising, we can say that $\pi^t(s_t) = min_a Q^*(s_t, a)$—it pessimistically assumes the student will take the worst action, representing the risk of the state. The advantage of such an assignment is that it is based on a well-tested criterion [6] and that it does not need knowledge of the student's intended action (desirable for most realistic settings). The disadvantage is that we have a less detailed reward, which is also not adapting to the student's specific necessities but mostly to the domain's characteristics.

Based on this dichotomy, we propose two versions of Q-Teaching (see Algorithm 1), the off-student's policy Q-Teaching and the on-student's policy Q-Teaching. The on-student's policy Q-Teaching uses the value of the actual student's action to compute the reward (thus, it is directly influenced by its policy). We can intuitively say that on-student's policy Q-Teaching will advise when the student is mostly expected to act sub-optimally with respect to the acting value function of the teacher, Q_Σ. On the other hand, the off-student's policy Q-Teaching uses the criterion discussed above and the teaching policy is not *directly* influenced by the policy of the student. Specifically, it is rewarding its teaching policy, π_T, at time-step $t + 1$ with the Q-value difference of the best action a^* to the worst action, as these were found at time t.

The Q-Teaching algorithm proceeds as follows (see Algorithm 1). A teacher agent enters an RL acting task to learn an acting policy. It initializes two action-value functions, Q_Σ and Q_T, the acting value function and the teaching value function, respectively (lines 1–2). Of course, it can also use an existing acting value function.

Being in time step t and state s, the teacher queries its acting value function for the greedy action in that state (line 6). Depending on whether we use the off-student's policy or the on-student's policy Q-Teaching, the teacher sets a baseline action, $\hat{a}$, to either the worst possible action for that state or to the action just announced by the student (lines 7–11).

Then, the teacher chooses an action from $A_T = \{advice, no\_advice\}$ based on Q_T and its exploration strategy. If the teacher chooses to advise (line 13), it gives the action a^* as an advice to the student agent. If the teacher chooses not to advise, the student will proceed with its own policy.

Algorithm 1 Q-Teaching

1: Initialize $Q_T(s,a)$ arbitrarily ▷ teaching value function
2: Use existing $Q_\Sigma(s,a)$ or initialize it ▷ acting value function
3: **repeat** (for each teaching episode) ▷ teacher–student session
4: Initialize s
5: **repeat** (for each step)
6: $a^* \leftarrow \max_a Q_\Sigma(s,a)$
7: **if** (Off-Student's policy Q-Teaching) **then**
8: $\hat{a} \leftarrow \min_a Q_\Sigma(s,a)$
9: **else**
10: $\hat{a} \leftarrow a$ ▷ where a is the action announced by the student
11: **end if**
12: Choose a_T from s_T using policy derived from Q_T (e.g. ϵ-greedy)
13: **if** a_T = {advice} **then**
14: Advice the student with the action a^*
15: $b \leftarrow b - 1$ ▷ update remaining budget
16: **else if** a_T = {no_advice} **then**
17: Send a $\perp$ (no advice message) to the student
18: **end if**
19: Observe student's actual action a and its new state and reward, s', r
20: $Q_\Sigma(s,a) \leftarrow Q_\Sigma(s,a) + \alpha[r + \gamma \max_{a'} Q_\Sigma(s',a') - Q_\Sigma(s,a)]$ ▷ possibly continue learning an acting policy
21: **if** a_T = {advice} **then**
22: $r_T \leftarrow Q_\Sigma(s,a^*) - Q_\Sigma(s,\hat{a})$
23: **else if** a_T = {no_advice} **then**
24: $r_T \leftarrow 0$
25: **end if**
26: $Q_T(s,a_T) \leftarrow Q_T(s,a_T) + \alpha[r_T + \gamma \max_{a'} Q_T(s',a') - Q_T(s,a_T)]$
27: $s \leftarrow s'$
28: **until** $b = 0$ OR teacher reached the estimated convergence horizon episode of the student
29: **until** end of teaching episodes

On line 19, the teacher observes the student's actual action a and its new state and reward, s', r. Once again, the student may be the teacher himself; in this case, it observes its own action that was taken based on Q_Σ and its exploration strategy.

On line 20, the first Q-Learning update takes place for the acting value function Q_Σ based on the environment's reward. For the teaching value function update, the teacher's reward, r_T is calculated first, based on the *freshly updated values* of the best and baseline actions, a^* and $\hat{a}$, respectively (lines 21–25).

Finally, a Q-Learning update for the teaching value function takes place based on the reward r_T (line 26) and the algorithm continues in the same way until whatever of the following two events comes first: either the advice budget finishes or the student reaches a learning episode that we have predetermined as its convergence horizon. These complete one learning episode or session for the teacher.

In this version, the Q-Teaching algorithm is based on the Q-Learning algorithm, although, in principle, any RL algorithm could be used for the underlying learning updates of Q-Teaching. However, if an off-policy RL algorithm such as Q-Learning is chosen for the updates of both the acting and the teaching value function, then the point of transition from acting to teaching is irrelevant to the learning progress of the two policies. Reducing the impact of the exploration policy to the learning updates allows for smoother interaction between the two policies and ensures us that we continue to learn the same policies. In principle, a Q-Teaching agent is able to update both its acting and teaching value functions continually and refine not only *when* it should advise but also *what* it should advise.

Since our goal is to introduce Q-Teaching as a flexible and generic enough method to be applied to multiple domains, we propose a series of state features for the teaching task state space that we think are necessary. From our experiments, Q-Teaching works best with an augmented version of the acting task state space (see Table 3) similar to that of [5] (Zimmer's method). Also in Table 3, note the

role of the student's progress feature ($f3$): it homogenises the student's Markov chain by inducing a state feature for time (see Section 3.2).

Table 3. The augmented state feature set for the Teaching Task.

a/a	Feature	Description
f_1	Advice	Binary feature indicating if the current state is the result of advice
f_2	Budget	Remaining advice budget
f_3	Student's progress	At least one informative feature for the student's learning progress (e.g., current episode)
f_4	Student's intended action	The action announced by the student (optional)
f_i	⋮	Acting task original state features

5.4. Experiments and Results

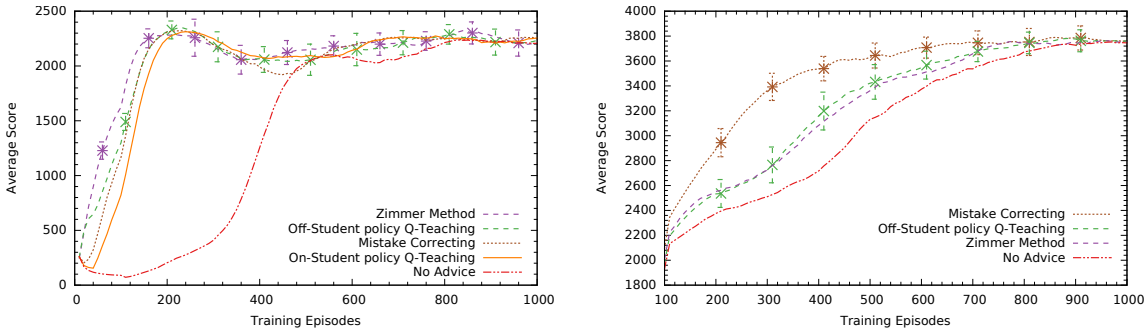
In this section, we present results from using Q-Teaching in the Pac-Man Domain. We evaluate both on-student's policy Q-Teaching and off-student's policy Q-Teaching, in two variations each: known or unknown student's intended action. Note that methods like Zimmer's and Mistake Correcting require knowledge of the student's intended action.

We use two versions of students for the experiments. A low-asymptote and a high-asymptote Sarsa students. Referring to [6] and Section 2.3, the low asymptote students receive a state vector of 16 primitive features related to the current game state while the high asymptote students receive a state vector of seven highly engineered features providing more information [6]. The low-asymptote students have significantly worse performance than the high-asymptote ones.

Additionally, we choose to bootstrap all compared teaching methods with the same acting policy in order to equally compare their advice distribution performance and not their quality of the advice. The acting policy used for producing advice comes from a high-asymptote Q-Teaching agent after 1000 episodes of learning. Moreover, we use Sarsa students in order to emphasize the ability to advise students that are different to the teacher. All learning methods (Zimmer's and Q-Teaching) were trained for 500 teaching episodes (sessions) to be equally compared for their learning efficiency too.

The Q-Teaching learning parameters for the teaching policy were $\alpha = 0.002$, decaying $\epsilon = 0.5$ and $\lambda = 0.9$, whereas all Sarsa students had $\alpha = 0.001$, $\epsilon = 0.05$ and $\lambda = 0.9$. The evaluation was based on the student performance (game score) and using the *Total Reward* TL metric [2] divided by the fixed number of training episodes. The student performance is evaluated every 10 advising episodes (while learning) for 30 episodes of acting alone (and not learning). For the comparisons between average score performances, we used pairwise t-tests with Bonferroni correction [18]. Statistically significant results are denoted with their significance level and they always refer to paired comparisons.

In Figure 4a, teacher agents advise a low-asymptote Sarsa student who always announces its intended action. We can see Zimmer's method performs best and off-student's policy Q-Teaching comes second with a statistically significant difference ($p < 0.05$). The heuristic based-method Mistake Correcting with a tuned threshold value of $t = 100$ comes third. On-student's policy Q-Teaching performed worse than the previous three methods by a small margin, having not found an as good advice distribution policy (non-significant difference to Mistake Correcting). Finally, all methods performed statistically significantly better ($p < 0.05$) than not advising, effectively speeding up the learning progress of the student.

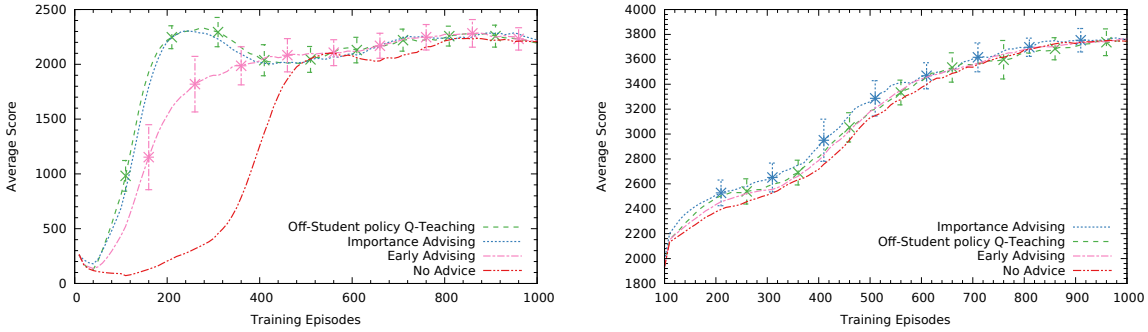


(**a**) (**b**)

Figure 4. Average student score in 1000 training episodes with teachers knowing the student's intended actions. The curves are averaged over 30 trials and the legend is ordered by score. The error bars represent the 95% confidence intervals (CI) of the means. Non-overlapping CIs indicate statistically significant differences of the means whereas overlapping CIs are inconclusive. (**a**) low-asymptote Sarsa student; (**b**) high-asymptote Sarsa student.

In Figure 4b, the teachers advise a high-asymptote Sarsa student. Here, the tuned version of Mistake Correcting ($t = 200$) performed statistically significantly better ($p < 0.05$) than all methods, with Q-Teaching methods coming second and third (respectively) and Zimmer's method coming next (having non-significant differences between them).

For the case when the teacher agent is not aware of the student's intended action (see Figure 5), in Figure 5a the off-student-policy Q-Teaching performs best while Importance Advising ($t = 200$) follows with a small performance difference (non significant). Early Advising (giving all B advice in the first B steps) performs statistically significantly worse (at $p < 0.05$) than both Q-Teaching and Importance Advising. In these experiments, we did not use on-student's policy Q-Teaching since that requires knowing the student's intended action to compute the reward.



(**a**) (**b**)

Figure 5. Average student score in 1000 training episodes with teachers *not knowing* the student's intended actions. The curves are averaged over 30 trials and the legend is ordered by total reward. The error bars represent the 95% confidence intervals (CI) of the means. Non-overlapping CIs indicate statistically significant differences of the means whereas overlapping CIs are inconclusive. (**a**) Low-asymptote Sarsa student; (**b**) High-asymptote Sarsa student.

In Figure 5b, advising a high asymptote Sarsa student, Q-Teaching had the second best performance with the heuristic-based method importance advising ($t = 200$) performing better (non significant). For high performing students, a poorly distributed advice budget can be much less effective. For example, if the teacher knows the student's intended action, it does not spend advice in states where the student would anyway choose the correct action. This fact is emphasized

in this specific case, since no advising did not perform significantly worse compared to the rest of the methods.

Finally, in Tables 4 and 5, we can see the average total reward in 1000 training episodes for all of the teaching methods. All methods knowing the student's intention performed better than those that did not, taking advantage of that knowledge.

Table 4. Average student score and session time for 1000 training episodes of Low-asymptote Sarsa students. The scores are averaged over 30 trials and ordered (▼) by total reward. Best performance and smallest session time (faster training) are denoted in **bold** letters.

	Student Intention	**Average▼ Student Score**	**Avg. Session Time**	**Avg. Budget Finish**
Zimmer method	Known	**2145.35**	1000 ep.	17 ep.
Off-Student Q-Teaching	Known	2103.28	42 ep.	31 ep.
Mistake Correcting	Known	2086.03	-	20 ep.
On-Student Q-Teaching	Known	2062.3	25 ep.	4 ep.
Off-Student Q-Teaching	Not Known	2048.47	**18 ep.**	4 ep.
Importance Advising	Not Known	2026.62	-	10 ep.
Early Advising	Not Known	1910.16	-	2 ep.
No Advice	-	1485.01	-	-

Table 5. Average student score and session time for 1000 training episodes of High-asymptote Sarsa students. The scores are averaged over 30 trials and ordered (▼) by total reward. Best performance and smallest session time (faster training) are denoted in **bold** letters.

	Student Intention	**Average▼ Student Score**	**Avg. Session Time**	**Avg. Budget Finish**
Mistake Correcting	Known	**3429.97**	-	240 ep.
Off-Student Q-Teaching	Known	3235.12	28 ep.	13 ep.
On-Student Q-Teaching	Known	3221.75	**22 ep.**	11 ep.
Zimmer method	Known	3218.05	1000 ep.	8 ep.
Importance Advising	Not Known	3162.28	-	30 ep.
Off-Student Q-Teaching	Not Known	3116.09	23 ep.	12 ep.
Early Advising	Not Known	3112.84	-	2 ep.
No Advice	-	3079.66	-	-

Q-Teaching, the only learning AuB method allowing students to not announce their intended action, performed relatively well compared to methods that know the student's intended action, which is an advantage of the proposed method.

Most importantly, while Zimmer and Q-Teaching methods were both trained for 500 episodes (sessions), Q-Teaching training was completed significantly faster since the Zimmer method has to observe all 1000 episodes of each student session to complete just one of its own, whereas Q-Teaching has an upper bound for its episode completion. This upper bound is the algorithmic convergence of the student (e.g, the low-asymptote student requires only 500 episodes to converge) and, in most cases, it will be completed much faster, when the budget finishes (around the 30th episode for the low-asymptote student). More specifically, in Tables 4 and 5, we can see the average training time needed for each teacher (in terms of the average observed *student* episodes) in each of the 500 teacher episodes. In general, our proposed methods need at least $\times 25$ less training time than the Zimmer's method. We should also note here that, although non-learning methods do not need training time, they require a significant and variable amount of manual parameter tuning to achieve the reported performance.

Another advantage of off-student's policy Q-Teaching is that it can use the same teaching policy for very different students since it is not directly influenced from the student's policy and the rewards received by the student when not advising (such as in the Zimmer method). This is a significant

advantage in terms of learning speed and versatility since heuristic methods have to be manually tuned for each student separately to find the optimum threshold, t.

On-student policy Q-Teaching did not perform as well as expected, the main problem being the non-stationary reward depending on the student's changing policy. We believe that this method needs significantly more training time than the off-student's policy Q-Teaching because of its non-stationary reward and it probably needs more informative features for the student's current status. In our case, this was only its training episode, which is the most basic information available for the student. Moreover, the training episode feature is student-dependent since its meaning varies among students—some students learn faster than others.

6. Related Work

There are several types of related work in the area of helping to learn. Some of this work focuses on teaching in non-RL settings [19,20].

In the field of *transfer learning* in RL [2], an agent uses knowledge from a source task to aid its learning in a target task. However, agents perform transfer knowledge from one task to another and in an offline manner. Other differences of this typical TL setting to Agent Advising are described in Section 2.2 of this article.

More closely related work has one RL agent teach another without a direct knowledge transfer. Examples of such works include *imitation learning* [21] and *apprentice learning* [22]. In these approaches, an expert provides demonstrations of the task to a student; then, the student has to extract a policy by either learning directly from them or building a model to generate mental experience. In our setting, the teacher does not provide a full-policy trajectory and has a limitation on the number of interventions (advice budget). Moreover, we do not require a student with special processing abilities except that of being able to receive advice.

In [23], a joint framework for advising is presented that allows advising to be controlled from both the teacher and the student agent. Although similar to our setting, this work employs a different interaction model from ours by assuming the student can actively ask for advice. Besides this, the advise distribution process in this work is based on heuristic functions and not learning.

A multi-agent advising framework is presented in [24]. This work assumes a significantly different setting with multiple agents learning simultaneously and advising each other using a common advising strategy. This is a promising approach for its multi-agent setting and could possibly be extended in the future to involve learning in its advice distribution phase, which is now based on probabilistic confidence functions.

We emphasize the setting differences since the characteristics of each setting also determine what kind of knowledge is available to the teacher and/or the student. This in turn (dis)allows different learning approaches.

For the same setting as ours (i.e., an active advisor and a passive advisee), a non-learning teaching framework for RL tasks is presented in [6] based on action advice. The methods presented there are described in more detail in Section 2.2. One drawback of these methods is that, since they are based solely on the teacher's Q-values, they are not able to handle non-stationarity in the student's learning task, and also have to be given a threshold of Q-value differences, above which a state is considered important. This parameter needs to be manually tuned for each student in contrast to off-student's policy Q-Teaching, which can learn a more generic teaching policy focusing on the criticalities of the state space.

In addition, since the methods presented in [6] are heuristic-based and not based on learning, the agent may spend all of its advising budget on early learning steps of the student that satisfy the importance threshold, while it may later experience even more important states that further exceed the given threshold.

The only other learning method for advising in our context is Zimmer's method [5]. The method proposed there is described in more detail in Section 2.2. One significant difference is that the method

is based on the same reward received by the student, needing ad hoc modifications for each task to encourage teachers towards a better advising policy. Our method uses a domain-independent reward signal based on the acting task Q-values and can be directly used in any task. Moreover, their method has greater data complexity since a complete batch of student training episodes is required for just one training episode of the teacher. As discussed in the previous section, our method may finish one teaching episode as early as the budget finishes; that is multiple times faster completion of one episode. Finally, but most important, Q-Teaching can be used in the more realistic setting where there is no knowledge of the student's intended action.

Concerning the model selection criteria proposed in Section 4.1 for the teacher's acting policy, to the best of the author's knowledge, there is no other work in the relevant literature examining these criteria and furthermore proposing performance variance, and, specifically CV, as an important one. Most relevant works choose models based on their average performance, which as discussed previously, is not enough to evaluate the teaching effectiveness of a policy that will be sparsely and/or infrequently sampled to produce advice.

7. Conclusions

In this article, we discussed and proposed criteria, considerations and methods for the problem of learning teaching policies to produce and distribute advice.

Concerning advice production, we identify a model selection problem for the teacher, selecting the appropriate acting policy from which to advise. The experiments showed the significant relation of CV to the teaching performance, promoting CV as an important criterion—among others tested—for selecting acting policies for advising. Moreover, average-reward RL was found to produce effective policies for sparse advising under budget, although these policies may under-perform when used as acting ones.

Concerning advice distribution (i.e., teaching policy), we proposed a novel representation of the learning to teach problem as a constrained exploitation reinforcement learning problem. Based on this representation, we proposed a novel RL algorithm for learning a teaching policy, Q-Teaching, able to advise even when not having knowledge of the student's intended action. Compared to the other methods, Q-Teaching needs significantly less training time, while it performs equally well or better.

Advice distribution under budget is a challenging problem, both theoretically and practically, posing a series of problems such as the non-stationarity of the teaching task, as a result of having a learning student as part of the environment. Efficient and principled handling of the budget constraint is another challenge.

From our experiments, Q-Teaching can be considered a promising method based on a more formal understanding of the problem. It is significantly more efficient in terms of data complexity than Zimmer's method, and it can learn teaching policies without the assumption of having knowledge for the student's intended action.

There are several future directions. Q-Teaching could be adapted to student agents with specific "disabilities" and could also be tested under different budget constraints to examine how budget affects its teaching policies. In addition, off-student Q-Teaching could be tested on multi-student scenarios since not fitting to a particular student could be proven effective when teaching multiple different students. Moreover, the theoretical properties of the algorithms should be studied, especially the case of learning a teaching and an acting policy at the same time, e.g., under which specific assumptions a teaching policy converges.

The general usefulness of CV as a criteria for selecting teachers should be studied. Specifically, how teacher selection criteria such as CV are capturing the robustness of a policy when that policy is used sparingly for advising.

Some recent approaches such as the one presented in [25] propose novel interaction processes for learning agents allowing a transparent and interpretable model learning process while also enabling the natural integration of external information (such as advice) in the learning progress. In our specific

context, such a novel teaching architecture could possibly allow a teacher to use only one value function for both advising under a budget and acting. Such a hybrid agent transitions smoothly from its actor role to the teacher's one. A unified architecture and knowledge representation would further reveal the deep connection between acting and teaching, one we strongly believe exists.

Acknowledgments: We thank the reviewers for their constructive comments and suggestions. Part of this research has taken place in part at the Intelligent Robot Learning (IRL) Lab, Washington State University. IRL's support includes NASA NNX16CD07C, NSF IIS-1149917, NSF IIS-1643614, and USDA 2014-67021- 22174.

Author Contributions: Anestis Fachantidis, Matthew E. Taylor and Ioannis Vlahavas conceived and designed the experiments and they also wrote the paper; Anestis Fachantidis performed the experiments and analyzed the data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sutton, R.S.; Barto, A.G. *Reinforcement Learning, An Introduction*; MIT Press: Cambridge, MA, USA, 1998.
2. Taylor, M.E.; Stone, P. Transfer Learning for Reinforcement Learning Domains: A Survey. *J. Mach. Learn. Res.* **2009**, *10*, 1633–1685.
3. Lazaric, A. Transfer in reinforcement learning: A framework and a survey. In *Reinforcement Learning*; Springer: Berlin, Germany, 2012; pp. 143–173.
4. Zhan, Y.; Taylor, M.E. Online Transfer Learning in Reinforcement Learning Domains. In Proceedings of the AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (SDMIA), Arlington, VA, USA, 12–14 November 2015.
5. Zimmer, M.; Viappiani, P.; Weng, P. Teacher-Student Framework: A Reinforcement Learning Approach. In Proceedings of the AAMAS Workshop Autonomous Robots and Multirobot Systems, Paris, France, 5–9 May 2014.
6. Torrey, L.; Taylor, M. Teaching on a budget: Agents advising agents in reinforcement learning. In Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Saint Paul, MN, USA, 6–10 May 2013; pp. 1053–1060.
7. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292.
8. Taylor, M.E.; Stone, P.; Liu, Y. Transfer Learning via Inter-Task Mappings for Temporal Difference Learning. *J. Mach. Learn. Res.* **2007**, *8*, 2125–2167.
9. Fernández, F.; Veloso, M. Learning domain structure through probabilistic policy reuse in reinforcement learning. *Prog. Artif. Intell.* **2013**, *2*, 13–27.
10. Torrey, L.; Walker, T.; Shavlik, J.; Maclin, R. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proceedings of the Sixteenth European Conference on Machine Learning (ECML'05), Porto, Portugal, 2 October 2005*; Springer: Berlin, Germany, 2005; pp. 412–424.
11. Taylor, M.E.; Jong, N.K.; Stone, P. Transferring Instances for Model-Based Reinforcement Learning. In Proceedings of the European Conference on Machine Learning, Antwerp, Belgium, 15–19 September 2008; pp. 488–505.
12. Rohlfshagen, P.; Lucas, S.M. Ms Pac-Man versus ghost team CEC 2011 competition. In Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC), New Orleans, LA, USA, 5–8 June 2011; pp. 70–77.
13. Stroock, D. *An Introduction to Markov Processes*; Graduate Texts in Mathematics; Springer: Berlin/Heidelberg, Germany, 2004.
14. Schwartz, A. A reinforcement learning method for maximizing undiscounted rewards. In Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, USA, 27–29 July 1993; Volume 298, pp. 298–305.
15. White, A.; Modayil, J.; Sutton, R.S. Surprise and curiosity for big data robotics. In Proceedings of the AAAI-14 Workshop on Sequential Decision-Making with Big Data, Quebec City, QC, Canada, 27–28 July 2014.
16. Overlapping Confidence Intervals and Statistical Significance. Available online: https://www.cscu.cornell.edu/news/statnews/stnews73.pdf (accessed on 1 December).
17. Rummery, G.; Niranjan, M. *On-Line Q-Learning Using Connectionist Systems*; Technical Report CUED/F-INFENG-RT 116; Engineering Department, Cambridge University: Cambridge, UK, 1994.

18. Dunnett, C.W. A Multiple Comparison Procedure for Comparing Several Treatments with a Control. *J. Am. Stat. Assoc.* **1955**, *50*, 1096–1121.
19. Chakraborty, D.; Sen, S. Teaching new teammates. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan, 8–12 May 2006*; ACM: New York, NY, USA, 2006; pp. 691–693.
20. Stone, P.; Kaminka, G.A.; Kraus, S.; Rosenschein, J.S. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In Proceedings of the Twenty-Fourth Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010.
21. Lin, L.J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.* **1992**, *8*, 293–321.
22. Clouse, J.A. On Integrating Apprentice Learning and Reinforcement Learning. Ph.D. Thesis, University of Massachusetts Amherst, Amherst, MA, USA, 1996;
23. Amir, O.; Kamar, E.; Kolobov, A.; Grosz, B.J. Interactive teaching strategies for agent training. In Proceedings of the International Joint Conferences on Artificial Intelligence, New York, NY, USA, 9–15 July 2016.
24. Da Silva, F.L.; Glatt, R.; Costa, A.H.R. Simultaneously Learning and Advising in Multiagent Reinforcement Learning. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems, São Paulo, Brazil, 8–12 May 2017; pp. 1100–1108.
25. Holzinger, A.; Plass, M.; Holzinger, K.; Crisan, G.C.; Pintea, C.M.; Palade, V. A glass-box interactive machine learning approach for solving np-hard problems with the human-in-the-loop. *arXiv* **2017**, arXiv:1708.01104.