2017 Joint IEEE International Conference on Development and Learning and
Epigenetic Robotics (ICDL-EpiRob)
Lisbon, Portugal, September 18-21, 2017

Agent-advising Approaches in an Interactive Reinforcement Learning Scenario

Francisco Cruz*†, Peter Wüppen*, Sven Magg*, Alvin Fazrie*, and Stefan Wermter*
*Knowledge Technology, Department of Informatics, Universität Hamburg
†Escuela de Computación e Informática, Facultad de Ingeniería, Universidad Central de Chile
Emails: {cruz, 5wueppen, magg, 4fazrie, wermter}@informatik.uni-hamburg.de

***Abstract*—Reinforcement learning has become one of the fundamental topics in the field of robotics and machine learning. In this paper, we expand the classical reinforcement learning framework by the idea of external interaction to support the learning process. To this end, we review a number of proposed advising approaches for interactive reinforcement learning and discuss their implications, namely, probabilistic advising, early advising, importance advising, and mistake correcting. Moreover, we implement the advice strategies for interactive reinforcement learning based on a simulated robotic scenario of a domestic cleaning task. The obtained results show that the mistake correcting approach outperforms a purely probabilistic advice approach as well as the early and importance advising approaches allowing to collect more reward and also to converge faster.**

I. Introduction

Recently, novel Artificial Intelligence (AI) technology has been built for human-agent interaction. The interaction between humans and AI systems is increasing, e.g., by using companion robots, autonomous cars, and even video games since AI players can make the games become more impressive and challenging. In this regard, AI agents should implement learning approaches which could make them learn to act in unpredicted scenarios through interaction with humans or other AI systems.

Reinforcement Learning (RL) [1] has proven to have great potential as a learning method, especially when learning multiple tasks. The idea behind RL is to continuously observe the current environment and to respond accordingly with a set of desired actions. With an RL approach, the agent explores the environment around it with the aim of maximizing the rewards that it could gain by interacting with it.

Although RL works well in most cases, an open issue is the time needed to train an autonomous agent in terms of the repetition of actions and episodes [2]. To overcome this issue, an alternative is Interactive Reinforcement Learning (IRL) where a trainer is added to advise over the policy by suggesting actions to perform in selected states. Fig. 1 shows the IRL framework where the learner-agent continuously interacts with the environment and a trainer-agent provides advice in some episodes.

In IRL, a trainer-agent advises a learner-agent not in every single episode but rather in just some of them [3], [4]. Therefore, the question arises when to deliver the limited amount of advice in order to use it effectively in the episodes to achieve best learning and thus minimize the trainer-learner communication [5], [6]. In this paper, we show four agent-advising approaches: probabilistic advising, early advising, importance advising, and mistake correcting. All these approaches are tested within a domestic robot scenario using two artificial agents in an IRL framework, i.e. a learner-agent receiving advice from an artificial trainer-agent.

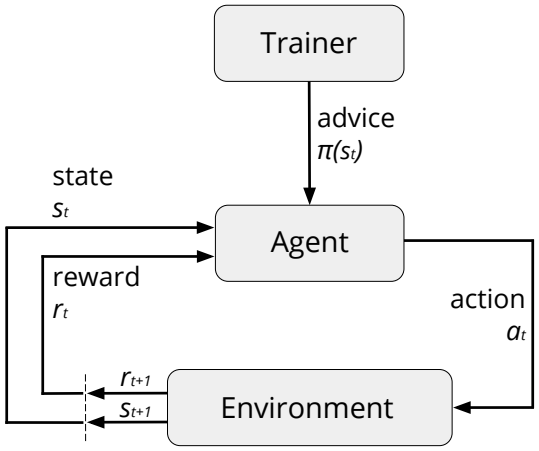

Fig. 1. Interactive reinforcement learning diagram showing an agent's interaction with its environment and a trainer providing action advice.

II. Reinforcement Learning and Interactive Feedback

Up to now, several fields have implemented the RL method and it is proven to show great performance in machine learning problems [7]. RL is a learning framework where an agent observes the environment continuously and learns to respond with a set of allowed actions within the environment. The agent, once it performed the desired action, will receive feedback from the environment. Each of the actions performed by the agent is selected based on an internal decision-making policy π which is simply a probability of selecting an action a to be performed for a given state s.

During the learning process, an RL agent perceives a certain state s_t each time ($s_t \in S$) and needs to select a possible action a_t to perform it ($a_t \in A(s_t)$). Once the action has been executed by the agent, a positive or a negative value will be provided as a reward r_{t+1} for the agent by the environment together with the next state s_{t+1} as shown in

Fig. 1. Moreover, the learning process changes the policy in order to gain experience from this episode, as the main goal of the agent is maximizing the accumulated reward [1].

The total reward estimation which an agent could obtain by performing an action in the current state (action-value pairs) can be estimated as follows:

$$Q^{\pi}(s,a) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} | s_t = s, a_t = a\} \quad (1)$$

where $Q^{\pi}(s,a)$ is the state-action pair value and r_t is the reward for action $a = a_t$ under the policy π in the state $s = s_t$. Moreover, γ is the discount rate determining future action's influence $(0 \leq \gamma < 1)$. The optimal estimation of state-action pair values $Q^{\pi}(s,a)$ can be obtained by the Bellman equation as follows:

$$Q^{*}(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q(s',a')] \quad (2)$$

where $Q^{*}(s,a)$ is the optimal state-action pair value estimation, P is the probability to achieve the subsequent state $s' = s_{t+1}$ by performing action a in the current state s, and a' represents the possible action in the future state s' [1].

The basic strategy of RL is very closely related to how humans and animals initially learn to fulfill tasks and interact with their environment [8]. However, human learning mechanisms have a clear advantage in the direct comparison: they do not exclusively work tediously by trial and error, but have helpful outside influences to guide learning in social environments. For example, imitation [9] and demonstration [10] play a big role for transfer of behavioral knowledge. Another important aspect is guidance during autonomous learning by having a knowledgeable other person give input on the learner's decisions [3], [4], essentially teaching them crucial strategies to solve the task at hand.

Thomaz et al. [3] have explored fundamental properties of guidance in an IRL task. Their goal was to find out in what ways humans want to teach an agent and thus how the agent should be set up to interpret the advice it gets in an optimal way, so that eventually a person with no expert knowledge of the inner workings or learning procedures of an agent could support it in learning a previously unknown task in a reasonable amount of time. The authors concluded that IRL approaches should involve both feedback as well as guidance channels to improve the learning process as much as possible and to offer intuitive ways for trainers to teach agents.

Additionally, Taylor et al. [6] have discussed the viability of a different form of advising to have an agent teaching another one about a certain IRL task. Their chosen problem domain was one of video games. By finding good teaching strategies between agents, they expect to transfer this knowledge into the construction of agents that can also teach humans effectively about games after autonomously learning about them themselves.

III. Agent Advising Approaches

One critical limitation for trainer-agents when teaching other agents is that they cannot give an unlimited amount of advice. This limitation comes from the idea that human trainers have indeed limited patience and attention when delivering advice to others [3]. As an additional motivation for this limitation, one could also argue that learner agents sometimes want to figure out how to solve a task by themselves instead of just being guided through the process with no perceived decision-making on their part.

To model this limited advice, the notion of a budget, available to the trainer, has been previously used [5]. The budget is referred to as n and is a fixed number that denotes how often the trainer-agent is allowed to intervene during a learning episode of the learner to suggest a certain action. This naturally leads to the question of how this budget is best spent and what criteria the trainer should look at to determine whether to help the learner with a particular choice or not.

In the following subsections, we describe three different strategies for spending the teaching advice budget, namely, early advising, importance advising, and mistake correcting [6]. Additionally, we also describe a probabilistic advising strategy which does not use an advice budget, nonetheless, we use it as a baseline to compare the obtained results.

A. Probabilistic advising

Probabilistic advising does not use an advice budget, therefore, the trainer-agent can advise the learner at any time during the learning process based on a fixed interaction probability $\mathcal{I}$. At each learner state, a random number is drawn to determine whether the learner-agent is advised or not. Algorithm 1 shows the probabilistic advising approach.

Algorithm 1 Probabilistic Advising

```
1) procedure ProbabilisticAdvising (π, I).
2)     for each learner state s do
3)         if rand(0,1) < I then
4)             Advice π(s)
5)         end if
6)     end for
7) end procedure
```

B. Early advising

Early advising simply lets the trainer-agent spend its budget as soon as possible, i.e. in the first n states the learner-agent encounters. The intuition behind this approach is that it might be best to help the learner most in the beginning when it still knows very little about the task. Algorithm 2 shows the early advising approach.

Algorithm 2 Early Advising

```
1) procedure EarlyAdvising (π, n).
2)     for each learner state s do
3)         if n > 0 then
4)             n ← n − 1
5)             Advice π(s)
6)         end if
7)     end for
8) end procedure
```

C. Importance advising

Importance advising takes into account the fact that giving advice may be more crucial in some situations than in others. To that end, an importance function $I(s)$ maps states to real values and indicates how important it is to get the decision about the action right in this particular state. Many implementations of the importance function have been suggested [6], including measuring the maximal difference between Q-values for the state as well as their variance and absolute deviation. The latter two produce more consistent results, as they take more Q-values into consideration than the two most extreme ones. The trainer-agent then evaluates this function in every state and, if it exceeds a certain threshold, decides to give advice to the learner-agent in this step. Algorithm 3 shows the importance advising approach.

Algorithm 3 Importance Advising

```
1) procedure ImportanceAdvising (π, n, t).
2)     for each learner state s do
3)         if n > 0 and I(s) ≥ t then
4)             n ← n − 1
5)             Advice π(s)
6)         end if
7)     end for
8) end procedure
```

D. Mistake correcting

Mistake correcting differs from the other strategies as it takes into account the decision that the learner-agent makes autonomously before deciding whether to give advice or not. The idea behind this is that a sparse advice budget should not be spent when the learner is already making the right decision. The downside of this is another layer of communication, as learners would need to inform the trainer about their decision first and wait for potential advice before carrying out any actions. Advice is thus dependent both on the action taken by the learner as well as the importance metric (both preconditions need to be fulfilled). Algorithm 4 shows the mistake correcting approach.

Algorithm 4 Mistake Correcting

```
1) procedure MistakeCorrecting (π, n, t).
2)     for each learner state s do
3)         Observe learners announced action a
4)         if n > 0 and I(s) ≥ t and a ≠ π(s) then
5)             n ← n − 1
6)             Advice π(s)
7)         end if
8)     end for
9) end procedure
```

IV. Robotic Scenario

To test the aforementioned advising approaches, we implemented a domestic robot scenario proposed in [11]. The scenario features a humanoid robot facing the task of cleaning a table in front of it. The table is separated into the two sides *left* and *right*, both of which are initially dirty. There is a third location *home* and two objects: a *cup*, initially located at either *left* or *right*, and a *sponge*, initially located at *home*. Every state s_t is represented as a vector as follows:

$$s_t =< handObj, handPos, cupPos, sideCond[] >, \quad (3)$$

where *handObj* is the object held in the robot's hand if any, *handPos* is the robot's hand position, *cupPos* is the cup position, i.e. *left* or *right*, and *sideCondition[]* a tuple which indicates if each side of the table has been already cleaned or not.

The robot's arm, with which the cleaning task is to be carried out, starts at the location *home* and can perform the following actions: *get* to pick up an object at its current location, *drop* to place the currently carried object, *go left*, *go right*, and *go home* to move the arm to any of the locations, *clean* to attempt to clean the current location, using the object being carried, and *abort* to cancel the current learning episode and return to the initial state.

After cleaning, the robot is meant to return the sponge to the *home* position, which is the final goal state of the system. In this task, the optimal sequence consists of 15 actions. Many of the actions can lead to failure, like trying to *clean* something with the *cup* in hand or trying to clean a location that the cup is currently at. We refer to these states as failed-states which in this context are the states from where it is not possible to reach the goal state anymore. The reward for every state encountered is -0.01 to discourage redundant behavior, with the exception of the goal state and the failed-states, which reward 1 and -1 respectively.

In our scenario, first, an agent learns autonomously the cleaning-table task and, afterward, this agent becomes the trainer-agent delivering advice in some episodes to learner-agents to speed up the learning process. The artificial trainer-agent uses all the aforementioned advising approaches, i.e. probabilistic advising, early advising, importance advising, and mistake correcting.

V. Results and Discussion

Using the aforementioned robotic scenario, we trained an agent autonomously on the cleaning-table task using SARSA temporal-difference learning for 1000 episodes. The chosen parameters were: learning rate $\alpha = 0.1$, discount factor $\gamma = 0.9$, and an ϵ-greedy action selection with $\epsilon = 0.1$. In a second run, a trained agent is used to teach a new one using IRL. In both RL and IRL approaches a set of 30 agents was used and the obtained results were averaged.

Initially, we implemented the probabilistic advising approach. For every state encountered, the trainer-agent had a probability of 0.3 of suggesting its own preferred action to the learner-agent. The result was a drastically increased rate of learning when compared to RL as can be seen in Fig. 2 where the learner-agent is able to collect greater reward rapidly. In all the figures RL is shown in yellow and IRL in red. To smooth the view of the results a convoluted gray line, in a window of 30 values, is shown through the results. By using a probabilistic advising approach, the trainer-agent ends up giving advice around 3500 times during an entire run, so roughly 3.5 times during a single episode. This leads to the question whether this advice could be given in a more efficient way so that, with a comparable number of interventions of the trainer, the learner-agent could learn even faster than it already does. To examine this, we implemented a fixed maximum for the amount of advice that can be given during a single episode as well as a number of strategies that the trainer could employ to make the advice more effective. Specifically, we experimented with early advising, importance advising and mistake correcting as described earlier.

For the early advising strategy, we set the advice budget per episode to 3, so that the trainer-agent would give advice for exactly the first 3 steps of each episode and never afterward. This adds up to exactly 3000 instances of advice-giving throughout all episodes, slightly less than the amount that probabilistic advising ended up giving. The average rewards obtained this way are shown in Fig. 3. We can see that the learner-agent actually learns much quicker initially, already acquiring an average reward of roughly -0.3 after 400 episodes. On the flip side, however, its behavior does not improve significantly anymore even after 600 additional episodes, whereas the probabilistic advice approach eventually reaches an average reward of just over -0.2. This seems to be in line with the philosophy of the early advising strategy: the learner is quickly brought on the right track at the beginning of each episode, but when exploring other solutions during later stages of the episodes is left completely on its own. Nevertheless, this could be useful for developing some kind of hybrid strategy, where early advising is gradually transformed into a more far-sighted approach as the number of episodes grows. Also, in the context of a human teacher working with a learning robot, this strategy is very easy and intuitive to apply.

The next strategy we implemented was importance advising. We chose as importance function the mean absolute deviation of Q-values of the state as observed by the trainer-agent [6]:

$$I(s) = \frac{1}{|A|} \sum_{i=1}^{|A|} |Q(s,a) - \overline{Q(s,a)}|. \tag{4}$$

where $|A|$ is the number of actions and $\overline{Q(s,a)}$ is the average Q-value over the actions for the state s. This way, we tried to focus advice on states that making mistakes in would be especially bad and to let the learner-agent make less important decisions by itself to preserve the budget. Finding the right threshold for the importance function proved to be non-trivial: if chosen too low, the strategy starts to closely resemble early advising; if chosen too high, the total amount of advice gets too low to allow for a fair comparison with the other approaches. We ended up with a threshold of 0.02 for the absolute deviation of Q-values, which resulted in approximately 3000 instances of the trainer giving advice throughout all episodes. The resulting average rewards can be seen in Fig. 4. Disappointingly, this approach does actually not seem to perform any better than the probabilistic method with a comparable amount of advice. The initial learning process is not accelerated significantly, and the average reward after 1000 episodes of training is slightly lower. One reason for this could be the fact that there is actually only the rather small number of 53 regular states that are either not the goal state or result in immediate failure, i.e. failed-states. This, combined with the fact that the typical episode does not last very long, could result in the trainer giving advice in the same states over and over, even if the learner has already developed the correct policy for those cases.

The natural answer to this problem would be only to give advice to the learner-agent if that advice actually leads it to perform a different action than what it otherwise would have done. This mistake correcting approach is a direct extension of importance advising: advice still only gets considered if the current state is considered important by the trainer, but now the learner's decision also has to actually differ from the trainer's in order to prompt a reaction. Running the scenario as a simulation, we are in a good position to be able to predict the learner's decision without actually having to let them perform it first, which makes the implementation of this approach less complicated. Fig. 5 shows average obtained rewards of agents trained using this interactive approach. Not only is the initial learning process of the agents significantly faster than with probabilistic advising, the final result shows a great difference to any of the other approaches as well. The average reward achieved after 1000 episodes ends up at about 0.81, where probabilistic advising would be at around -0.2 accumulated reward per episode. In comparison, the theoretical maximum reward per episode that can be achieved is 0.86, in an episode of 15 actions.

By using the mistake correcting approach, the majority of agents thus actually learns the optimal policy to fulfill the task. We can see this effect in Fig. 6, where the average amount of actions per episode is being shown. The missing averages from the earlier stages of the learning process result from the fact that all the agents ended up in failed-states for those episodes.

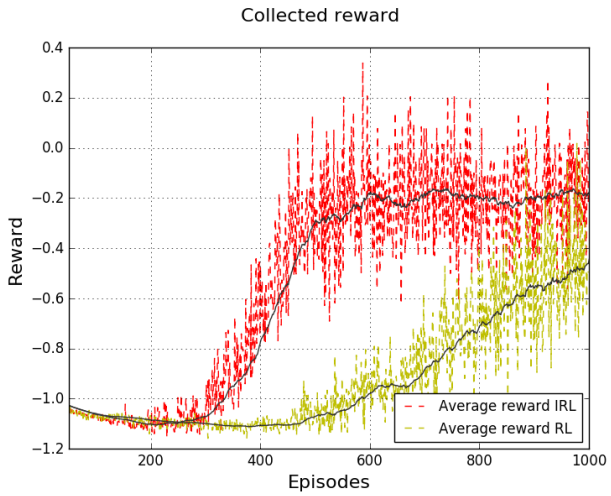

Fig. 2. Collected reward by RL and IRL agents using the probabilistic advising approach.

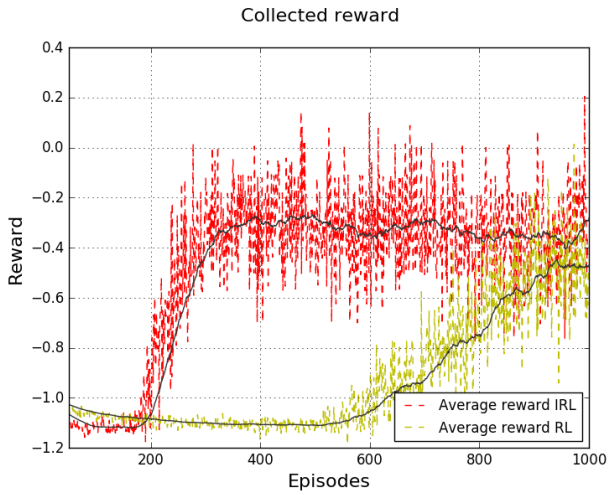

Fig. 3. Collected reward by RL and IRL agents using the early advising approach.

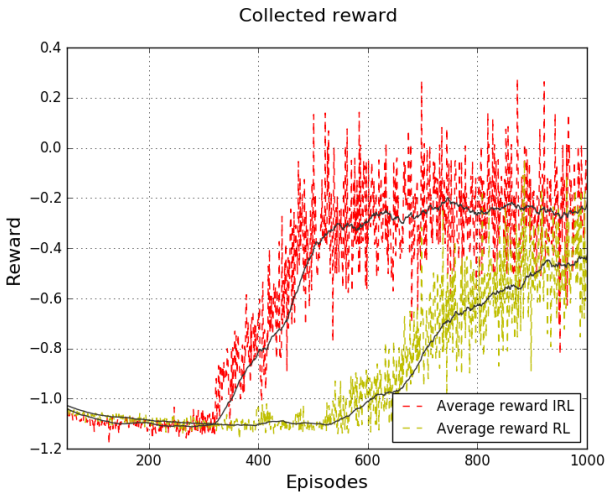

Fig. 4. Collected reward by RL and IRL agents using the importance advising approach.

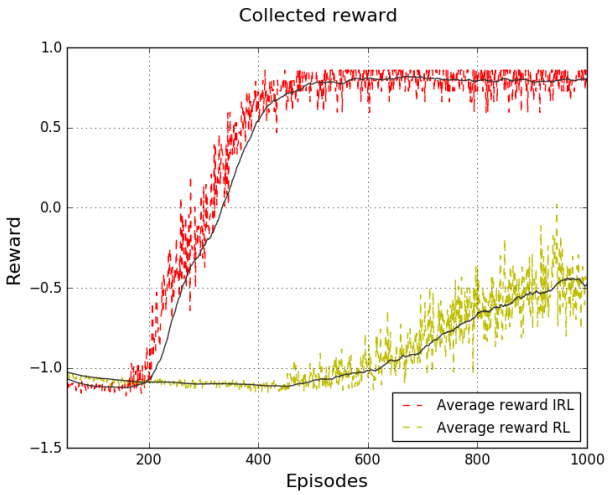

Fig. 5. Collected reward by RL and IRL agents using the mistake correcting approach.

The total amount of failures per episode for both the RL and the IRL approach is shown in Fig. 7. As can be seen, the IRL agents fail only very rarely after about 500 episodes and use the optimal sequence of 15 actions in the vast majority of the time when they succeed. The positive influence on the quality of the learned policy using mistake correcting is further confirmed when looking at the average amount of total advice given which is only around 1800, so roughly 2 instances per episode.

The additional power of having a mistake correcting mechanism certainly comes at a price. To reliably correct mistakes, we need to either wait for them to actually happen, or we need to know ahead of time what decision the learner-agent is going to take. Both approaches pose problems when we look at a real world application instead of a simulation, for instance, we often cannot simply tell a robot to undo its last action and do something else after it has already been performed, like when it tries to clean the table with a cup and crushes it in the process. Likewise, if we want to know the next action beforehand, the robot needs to constantly communicate about its plans and wait for possible feedback before actually executing anything. This is theoretically possible, especially if the number of possible actions is rather low and abstract. Nevertheless, for more complex scenarios with more parametrized actions, this could slow down the learning process considerably just by executing fewer actions in the same amount of time. A pure mistake correction strategy would probably not be feasible in those cases, but the concept is definitely always worth thinking about when looking at IRL tasks.

A possible solution for the strong constraints that the mistake correction method poses on the learning procedure would be to use predictive advising. This means that the

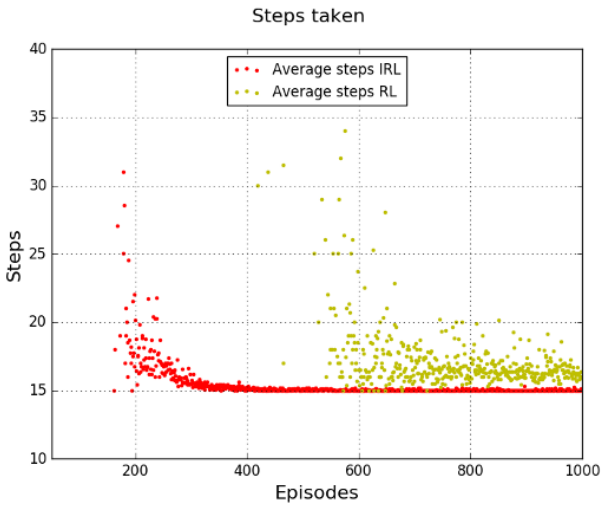

Fig. 6. Average amount of taken steps for all successful episodes using the mistake correcting approach.

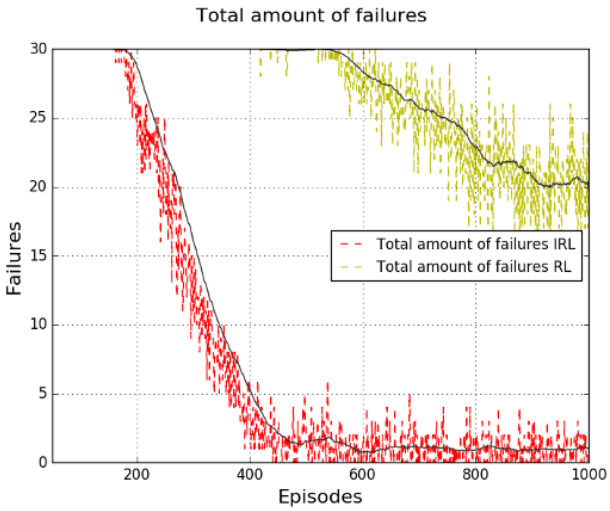

Fig. 7. Total amount of failed agents per episode using the mistake correcting approach.

learner-agent can always just execute steps without having to wait for feedback and puts the responsibility of figuring out when important wrong decisions would be taken solely on the trainer-agent. It could be expected that the learning process would be slower initially as the classifier for the learner's actions needs to be established with enough samples at first. The possible quality also depends heavily on the exploration policy of the learning agent, as random steps obviously cannot be predicted and can influence the classifier negatively. In our scenario, for example, the agents had a constant chance of 0.1 in every step of choosing a random action instead of the perceived optimal one. This could hinder the potential of a predictive advising approach, although it could work better than probabilistic, early, or importance advising.

VI. Conclusion

We have implemented an extension of the general RL approach to include the idea of a learning process that is autonomous but assisted by a trainer giving continuous advice throughout the learning process. We have shown advising approaches that aim to find efficient implementations for this concept and can be used in a real-world scenario. We have taken a closer look at the idea of advice budgets and how to efficiently act as a trainer when the amount of allowed feedback is limited. We implemented the strategies in a domestic robot scenario and evaluated their performance. To choose an appropriate advice strategy can be crucial in allowing the IRL process to achieve much better results in a shorter period of time. However, strategy-specific implications can also introduce strong restrictions on the system as a whole. Thus, the choice of the advising strategy should be carefully considered in regards to application-specific properties. In our scenario, the mistake correcting approach collected the greatest amount of reward close to the theoretical maximum possible.

Acknowledgment

The authors gratefully acknowledge partial support by CONICYT scholarship 5043, the German Research Foundation DFG under project CML (TRR 169), the European Union under project SECURE (No 642667), and the Hamburg Landesforschungsförderungsprojekt CROSS.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: Bradford Book, 1998.

[2] F. Cruz, G. I. Parisi, J. Twiefel, and S. Wermter, "Multi-modal integration of dynamic audiovisual patterns for an interactive reinforcement learning scenario," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, 2016, pp. 759–766.

[3] A. L. Thomaz and C. Breazeal, "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance," in *Proceedings of 21st National Conference on Artificial Intelligence*. Boston, MA, USA, 2006, pp. 1000–1005.

[4] W. B. Knox, P. Stone, and C. Breazeal, "Training a robot via human feedback: A case study," in *Proceedings of International Conference on Social Robotics*, 2013, pp. 460–470.

[5] L. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proceedings of International Conference on Autonomous Agents and Multi-agent Systems AAMAS*, 2013, pp. 1053–1060.

[6] M. E. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. Torrey, "Reinforcement learning agents providing advice in complex video games," *Connection Science*, vol. 26, pp. 45–63, 2014.

[7] M. Wiering and M. V. Otterlo, *Reinforcement Learning, State-of-the-Art*. Springer Heidelberg, 2012.

[8] Y. Niv, "Reinforcement learning in the brain," *Journal of Mathematical Psychology*, vol. 53, pp. 139–154, 2009.

[9] J. P. Bandera, J. A. Rodríguez, L. Molina-Tanco, and A. Bandera, "A survey of vision-based architectures for robot learning by imitation," *International Journal of Humanoid Robotics*, vol. 9, pp. 1–40, 2012.

[10] L. Rozo, P. Jiménez, and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks," *Intelligent Service Robotics*, vol. 6, pp. 33–51, 2013.

[11] F. Cruz, S. Magg, C. Weber, and S. Wermter, "Training agents with interactive reinforcement learning and contextual affordances," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, pp. 271–284, 2016.