Interaction in reinforcement learning reduces the need for finely tuned hyperparameters in complex tasks

Chris Stahlhut * **Nicolás Navarro-Guerrero** **
Cornelius Weber ** **Stefan Wermter** **

* *Ubiquitous Knowledge Processing Lab (UKP-TUDA), Department of Computer Science, Technische Universität Darmstadt, (e-mail: kognitivesysteme@cstahlhut.de)*
** *Universität Hamburg, Fachbereich Informatik, Knowledge Technology, WTM, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany*

Abstract: Giving interactive feedback, other than *well done / badly done* alone, can speed up reinforcement learning. However, the amount of feedback needed to improve the learning speed and performance has not been thoroughly investigated. To narrow this gap, we study the effects of one type of interaction: we allow the learner to ask a teacher whether the last performed action was good or not and if not, the learner can undo that action and choose another one; hence the learner avoids bad action sequences. This allows the interactive learner to reduce the overall number of steps necessary to reach its goal and learn faster than a non-interactive learner. Our results show that while interaction does not increase the learning speed in a simple task with 1 degree of freedom, it does speed up learning significantly in more complex tasks with 2 or 3 degrees of freedom.

Keywords: Interactive Reinforcement Learning; Continuous Actor Critic Automaton; Cacla; Multilayer Perceptron; Learning to Reach; Robot Teaching

1. INTRODUCTION

In recent developmental robotics research, teachers are becoming more and more common. Especially in Reinforcement Learning an additional interaction between the learner and a teacher can speed up learning. However, how much interaction between teacher and learner is necessary for the learner to benefit from it, has not yet been thoroughly studied.

Reinforcement Learning (RL) draws inspiration from the work of Thorndike [1911] as it models instrumental conditioning. Instrumental conditioning describes how a learner can improve performing a task based on positive and negative feedback. If an action a results in a positive feedback r in a particular situation or state s, the learner will more likely select the same action in this situation. If an action leads to a negative feedback, the likelihood of choosing the same action decreases.

One early example of the study of this phenomenon in animal behaviour is Thorndike's puzzle box. For instance, if a cat is placed inside a box, it will want to come out; but to escape the box, it needs to perform a sequence of actions, such as pushing a button and pulling a rope. At first the cat performs random actions until it, quite by accident, releases the door and escapes and gets fed. If the cat is again placed in the same box, it still behaves in most cases randomly, but tries those actions more often that it assumes to let it free. With repeated experiments and the cat's growing experience, the cat learns to escape the box and the time required decreases. Figure 1 shows an example of such a puzzle box. The line graph shows a sketch of the decreasing time for the cat to escape the box.

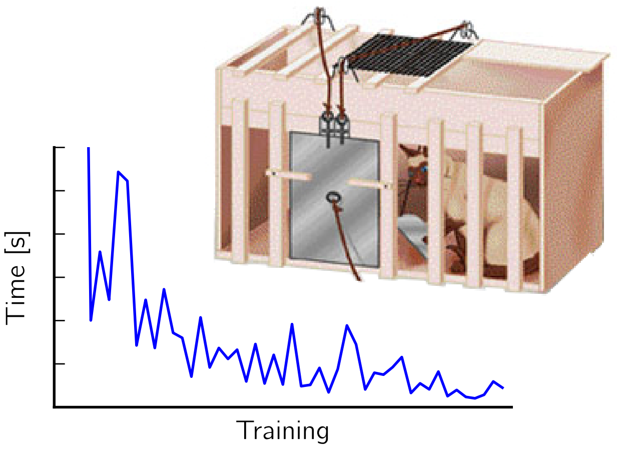

Fig. 1. A puzzle box in the top figure. The cat has to push a lever to open the door and escape. The plot in the bottom figure shows the decreasing time to escape with increasing experience. [1]

RL algorithms use mathematical models of the states, actions and rewards to simulate this kind of behaviour.

For example, if the probability of taking an action a is proportional to its reward r_a, then the highest rewarding action will be selected most often, similar to "probability matching" in humans [Shanks et al., 2002]. Initially, the algorithm has only a rough or random estimate of the real associations between the actions and rewards, but comparing the estimates with the real, received rewards and adapting the estimates allows the algorithm to learn.

The suitability of an action may depend on the state, e.g. pushing the floor down does not work if there is no lever and the stochastic behaviour or *policy* needs to depend on the state as well. The policy therefore maps from the set of states and actions to the probability of taking that action in this particular state.

In simple problems, some actions may lead the actor to an immediate improvement of the state. However in typical RL problems, the reward is usually delayed, which means that the actor needs to take some apparently suboptimal actions to obtain a higher reward in the long term. The actor would get stuck in a local optimum if it only took immediate-optimal actions. To overcome this problem a so-called *value function* is used. The value function associates an expected future reward to each visited state based on the agents' policy. In this way, the otherwise strong association between a state and the immediate reward is reduced. The value function is formulated in terms of the received reward r, the value of the expected future reward $\mathrm{V}(s')$ weighted by a discount factor γ. Minimising the absolute difference between the currently received value and the current estimate of the value function $\mathrm{V}(s)$ allows learning of the function. Such algorithms learn by minimising the Temporal Difference (TD) error

$$\delta = r + \gamma\,\mathrm{V}(s') - \mathrm{V}(s).$$

All such RL algorithms fall into the class of TD learning and are a good model of reward-driven learning in animals [Schultz, 2002].

TD learning forms the basis of Q-Learning in which the value function and policy are combined into a single $\mathrm{Q}(s, a)$-function [Watkins, 1989] and is one of the most widely used RL algorithms in discrete spaces [Thomaz and Breazeal, 2008, Duguleana et al., 2012]. Another subclass of TD learning algorithm is the actor-critic algorithm [Barto et al., 1983]. Its name derives from the dual memory used in these algorithms, i.e. the behavioural model or actor and the reward prediction model or critic. The actor encodes the state-action mapping, whereas the critic stores the value function $\mathrm{V}(s)$ and evaluates the outcome of the selected action in the form of a TD error.

2. INTERACTIVE REINFORCEMENT LEARNING

Interactive Reinforcement Learning (IRL) extends this framework by allowing additional interactions, such as putting the learner through a sequence of actions and therefore demonstrating a working example [Navarro-Guerrero et al., 2012] or allowing the learner to communicate with a more knowledgeable teacher [Suay and Chernova, 2011]. In this way, the teacher can help the learner to avoid actions which do not lead to a greater reward. Figure 2 shows the general architecture and interaction of the environment, learner, and teacher based on Sutton and Barto [1998] with the additional interaction between learner and teacher.

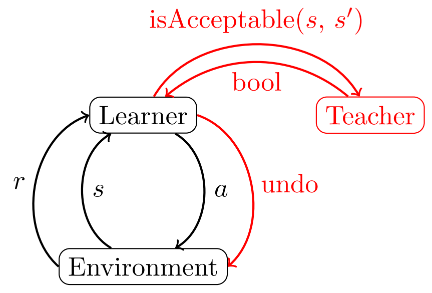

Fig. 2. The general RL framework with interactive feedback and the ability to undo the last action. The conventional model is drawn in black and the non-standard interactions in red.

The former approach of putting the learner through a working or nearly working example to create an initial behaviour is known as Learning from Demonstration (LfD) or putting-through or apprenticeship learning [Navarro-Guerrero et al., 2012]. Theodorou et al. [2010] used an algorithm called Policy Improvement with Path Integrals (PI^2) to let a simulated robotic arm with up to 50 Degree of Freedom (DoF) move from one fixed position to another fixed position via one predefined point.

The latter approach reduces the exploration space by allowing a user to suggest an object with which to do an action and therefore guide the learner by reducing the exploration space, see Thomaz [2006] and Thomaz and Breazeal [2008]. They tested the ability to guide a modified Q-learning algorithm with a simulated kitchen robot called Sophie. The task for the teacher was to help Sophie learn to bake a cake. Baking a cake included taking a bowl, adding flour, milk, etc. into the bowl and then mixing it with a wooden spoon and putting it into the oven. The teacher had the ability to not only give positive and negative reward, but to click on an object, e.g. on the bowl to suggest Sophie that it should do something with it. They also allowed the human teacher to "undo" an action if it was reversible, for instance if Sophie grabbed the milk before putting flour into the bowl or grabbed the bowl in the beginning; not undoable actions included adding the eggs or milk into the bowl or putting the eggs into the hot oven which is not only irreversible but also a catastrophic failure. Suay and Chernova [2011] used the same algorithm on a Nao robot [2] in which the robot was taught to sort objects into two boxes. Farkaš et al. [2012] trained an iCub to perform an action with the Continuous actor-critic learning automaton (Cacla), while simultaneously teaching the robot to linguistically describe its actions, which requires a teacher to be present.

Other algorithms also deal with the sparse and often imperfect nature of human feedback. For instance, Griffith et al. [2013] introduce the parameters $\mathcal{C}$, to model the consistency of the feedback, and $\mathcal{L}$ to model the likelihood of receiving feedback. If the likelihood is large but the consistency small, then the teacher will give advice very often, but not always the right one. If the likelihood is small

[1] Adapted from image under public domain by Jacob Sussman `https://commons.wikimedia.org/wiki/File:Puzzle_box.jpg`

[2] `https://www.aldebaran.com/en`

and the consistency large, then the little amount of feedback will be accurate. Griffith et al. [2013] compared their algorithm against different others with levels of consistency of 0.55, 0.8 and 1.0 and interaction likelihoods of 0.5 and 1.0. Yet, their focus is on the advantage of their algorithm and not on the necessary amount of interaction between the learner and teacher. All in all, a number of algorithms for interactive learning exists, which explore the benefit of different kinds of interaction. Yet, the amount of feedback needed to improve the learning speed and performance is still an open issue.

3. CONCEPT

In this paper, we intend to quantify the benefits of an autonomous interactive learning algorithm. We intend to determine to which degree RL benefits from interactive feedback outside the reward-function in the context of a reaching task. In a reaching task, we need not only to learn to map the goal position from workspace to joint space, also known as *inverse kinematics*, but also to learn the change needed to navigate the arm from its current position to the goal position.

It is possible to train this change in a supervised fashion, but this is only possible in cases in which there is a direct one-to-one mapping between work- and joint space, i.e. there are no multiple solutions, to the inverse kinematics. Once there are multiple solutions however, e.g. the point p can be reached with two configurations c and c' as depicted in Figure 3, the learner would reach through the middle of both configurations and therefore not reach the desired point p. This is particularly true if both configurations are sampled equally often.

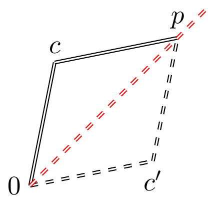

Fig. 3. Example for multiple solutions of the inverse kinematics. The configuration c' shows an alternative to the configuration c to reach the point p from the origin 0. The dotted red line shows the average of the two configurations.

RL, however, converges better on single solutions, because once the learner has found that configuration c will let it reach p, it will use this configuration more often than any other configuration and hence forth learn to reach the point p with configuration c instead of c'. This is also the case if more than two solutions for the inverse kinematics exist, e.g. by using three joints in a plane.

Our task with a variable goal specified as part of the actor's input is of interest to hierarchical RL [Rasmussen and Eliasmith, 2014]. A popular example task in hierarchical RL is the "taxi domain", which requires a taxi to navigate to varying passenger and destination locations. Transferred to a continuous domain, it resembles a pick-and-place scenario of a robotic arm. In hierarchical RL, a higher level RL module may specify a current goal location for the lower-level module as part of the lower-level's input. This would not be possible if the lower-level module only learns one fixed goal. This is a further reason to include a variable goal into our task set-up.

In our implementation of an interactive algorithm for learning to reach [Stahlhut, 2014, Stahlhut et al., 2015], we borrow the concept of undoable actions by Thomaz and Breazeal [2008] and the idea of likelihood of receiving feedback presented by Griffith et al. [2013]. For programmatic simplicity, however, we did not model it as a likelihood of interruption with feedback for the learner but as likelihood of the learner to engage the teacher. Specifically, we allow the learner to ask the teacher for feedback at every time step with a likelihood $\mathcal{L} < 1$; we speak of a fully interactive learner if $\mathcal{L} = \max \mathcal{L}$. The teacher then judges the last action by the change in the Euclidean distance between the hand and the goal. The feedback received by the learner is binary, based on the idea that a bad action increases the distance while a good action does not, i.e. the teacher gives a positive feedback if the distance did not increase. This set-up will help the learner to explore action sequences without non-beneficial states and thus speeds up the learning process.

Learning to reach is a task with both continuous input and output spaces. Of all the mentioned IRL algorithms, except for LfD, none of them works in continuous spaces because guiding the learner with a reduced exploration space requires the teacher to know in advance which actions are appropriate at each time step. Undoing the action after it has been done does not require the teacher to look into the internal decision-making process of the learner. It only needs to observe the actions physically taken by the learner, which is a realistic assumption.

To use IRL in continuous spaces, we first need a continuous spaces RL algorithm and then implement the modifications presented in Figure 2 and include the ability to undo action and ask the teacher for help. We choose the Continuous actor-critic learning automaton (Cacla) [van Hasselt and Wiering, 2007], because in contrast to the episodic natural actor-critic (eNAC) [Peters et al., 2005] does it not depend on the natural gradient and, therefore, can update the actor and critic at every time step.

We used multilayer perceptrons (MLPs) for the actor and the critic, because of their wide availability and ease of use. The output of the actor's MLP denotes the action $a(s)$ for the actor to perform given the state input s. Cacla lets the actor explore new policies by a small random variation $a'(s) = a(s) + \mathcal{N}(0, \sigma)$ of every action, where σ is typically a standard deviation of Gaussian noise as suggested by Williams [1992]. If and only if the variation leads to an improvement of the expected value, i.e. $\delta > 0$, then the new output will be learnt. Hence, the error $a' - a(s)$ causes learning of the actor MLP via back-propagation [Rumelhart et al., 1986].

Note that δ is estimated by the critic, which is part of the learner, and so $\delta > 0$ does not necessarily mean that the action is actually better. A separate critic MLP learns the state value $V(s)$. The value prediction error δ causes learning of the critic MLP via back-propagation. If the output of the actor results in an improvement, i.e. the TD error δ is positive, then updating the actor brings its output closer to the actually taken action a'.

Algorithm (1) shows the general implementation of the Interactive Continuous actor-critic automaton (ICacla).

Algorithm 1: The Interactive Continuous actor-critic automaton (ICacla) algorithm. The red statements show the differences to Cacla.

```
Initialize $\boldsymbol{\theta}$, $\boldsymbol{\psi}$, $s$
for t $\in \{0,1,2,\ldots\}$ do
    do
        Choose $a' \sim \pi(s, \boldsymbol{\psi})$
        Perform $a$, observe $r$ and $s'$
        stateImproved = True
        if $l \sim$ uniform(0, 1) $< \mathcal{L}$ then
            if not is_acceptable($s$, $s'$) then
                undo($a$)
                stateImproved = False
            end
        end
    while not stateImproved
    $\delta = r + \gamma \mathrm{V}(s') - \mathrm{V}(s)$
    updateCritic($s, r + \gamma \mathrm{V}(s')$)
    if $\delta > 0$ then
        for $i$ 0..$\lceil \delta/\sqrt{var_t} \rceil$ do
            updateActor($s, a'$)
        end
    if $s'$ is terminal then
        Reinitialize $s'$
    else
        $s = s'$
    end
end
```

If an action is exceptionally good, i.e. the ratio of it to the running variance var : $\delta/\sqrt{var_t}$ is large, the algorithm repeatedly updates the actor, making the action even more likely to be selected in the future.

We compared multiple fully interactive ICacla learners with non-interactive Cacla learners and later learners with different likelihoods of asking. Requiring a low level of interaction to improve the learning speed and performance is highly desirable when relying on human teachers. Section 6 explains our comparison in more detail.

4. ENVIRONMENT

We compared interactive and non-interactive learners using three set-ups, one consisting of a 1-Degree of Freedom (DoF) arm, another of a 2-DoF arm and lastly a 3-DoF that moves in a plane. In all three settings the learner was required to move the arm from an arbitrary starting position to a different goal position.

The first environment was the angular space of a complete circle with an goal zone of $\sqrt{0.0012}$rad, as shown in Figure 4. In this set-up, the relevant positions were encoded by the angles between the arm and the vertical line, or the goal and the vertical line, respectively. Since the arm has a fixed length, the invariable distance between the end-effector and the center was not encoded.

Figure 5 shows the workspace of the 2-DoF arm. The shoulder had an angular range of $[-\pi/2, \pi/2]$ and the elbow of $[-\pi, 0]$. Each link had a length of 0.366m. With this set-up we obtained a one-to-one mapping between Cartesian work- and joint space. The goal zone was a circle with

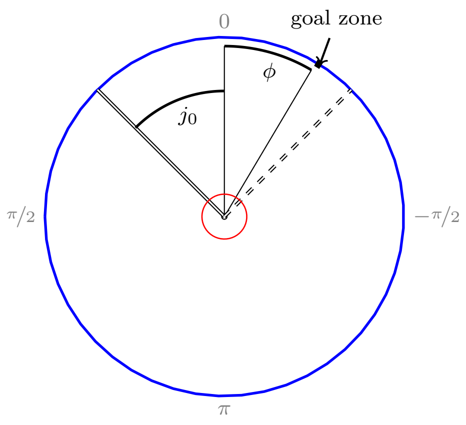

Fig. 4. The 1-DoF workspace is represented by the blue circle with the goal zone and legal value ranges of the joint indicated in red. The dashed lines show an alternative arm position. The value ϕ represents the angle of the goal zone.

a radius of 0.12m that could be placed with its centre anywhere within the workspace.

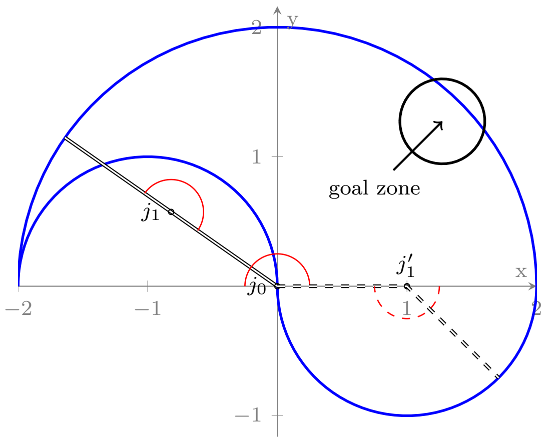

Fig. 5. The 2-DoF workspace is the space within the blue line. An example goal zone is visible at the upper right border or the workspace and the legal value ranges for the joints are depicted in red. The center of the goal zone can be placed anywhere within the workspace, even if this means that parts of it lie outside and thus are unreachable. The Cartesian coordinates are used to specify the goal's center for the learner to aim for.

The 3-DoF workspace is an extension to the 2-DoF workspace. We added an additional link of 0.2m length, see Figure 6. The range of motion for this new joint was $[-\pi/2, \pi/2]$. In this set-up there is no longer a direct one-to-one mapping between work- and joint space. Except for the workspace border, every position within the workspace can be reached with at least two joint configurations. The goal zone was the same as for the 2-DoF arm.

For all three scenarios, every joint was only allowed to move within the interval $[-\pi/4, \pi/4]$ per time step.

5. CONTROLLER SET-UP

We used varying starting and goal positions which required us to not only give the actor and critic the position

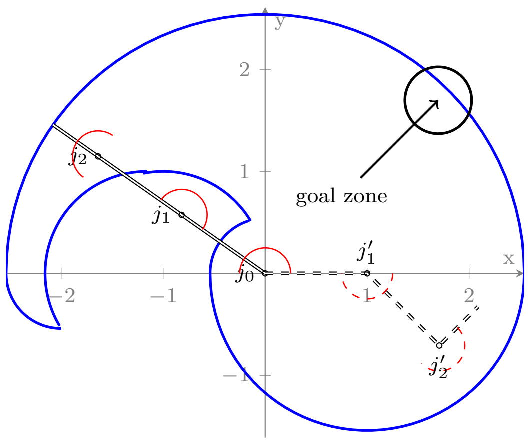

Fig. 6. The 3-DoF workspace is the space within the blue line, an example goal zone is shown in black and the legal value ranges for the joints are shown in red.

of the goal but also the current position of the arm itself. We added the goal's position in angular space for the 1 dimensional task and in Cartesian coordinates for the 2 dimensional tasks as input, as well as the arms proprioception in radians.

We determined the network's topologies by selecting the network with the smallest overall square error in learning to reach for the goal in a single step. This task can easily be learned with supervised learning by selecting the goal as a random position in joint space, which means the action is just the difference between the arm's and goal's position in joint space, and then calculating the Cartesian coordinates through forward kinematics. We selected networks with one and two hidden layers and 5, 10, and 20 hidden nodes each. We used the average distance between the hand and the goal after training the networks with 100 samples for 500 epochs as guide for the goal zone's size. We did not repeat the procedure for the 3-DoF arm but used the same number of hidden layers and nodes for it as for the 2-DoF arm.

The actor for the 1-DoF arm had a 2-dimensional input space and a 1-dimensional output space. For this set-up we only used a single hidden layer of 10 nodes, as shown in Figure 7.

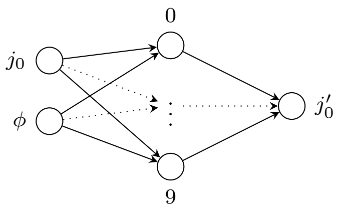

Fig. 7. The layout of the MLP for the 1-DoF actor. The input j_0 represents the arm's proprioception and the input ϕ the position of the goal's centre.

In both 2 dimensional tasks we used the Cartesian coordinates of the target (x, y) and the angular positions of every joint j_i. The 2-DoF arm used two input nodes for the proprioception and two as outputs, while the 3-DoF used three input and output nodes, as shown in Figure 8. Both networks used two hidden layers with 20 nodes each.

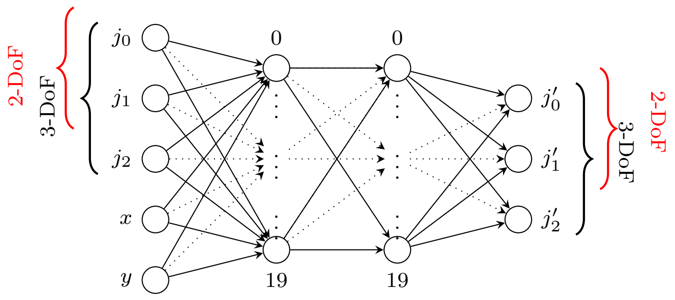

Fig. 8. The layout of the MLP for the 2- and 3-DoF arm actors. x and y represent the coordinates of the target position, j_i represent the current joint value and j'_i represents the chosen joint displacement.

6. METHODOLOGY

We selected a sequence of random initial arm and goal positions which all learners used. We initially compared non-interactive with fully interactive learners. We then selected the learners with the largest difference in the learning speed to ascertain how much interaction is necessary to gain an advantage.

We investigated the necessary amount of feedback by comparing a non-interactive learner against increasingly interactive ones. We effectively varied the likelihood of asking $\mathcal{L}$ in the range $[0, 99]$. We did not use the maximum likelihood of asking of 1, because there are special cases in which it is necessary to first increase the distance slightly to reach the goal. For instance in Figure 5, if the end-effector is at the position $(-2, 0)$ it can take no action without increasing the distance to a goal at position $(1, -1)$. In this situation, our teacher, which follows a simple heuristics "closer to goal is better", behaves counter-productively. Always obeying it would let the learner never discover the goal. Once a likelihood of asking was selected, it remained constant throughout the training.

For both set-ups we performed a testing episode every 10th episode. During a testing episode the learner was not further trained, action exploration and asking the teacher were turned off. We repeated every experiment with 10 differently initialized controllers to reduce fluctuations, but these controllers were initialised once and then cloned for each hyperparameter value. We selected hyperparameters randomly from the ranges shown in Table 1.

Tab. 1. The intervals for the hyperparameters.

Paramter		Interval
Critics's learning rate	α	[0.05, 0.2[
Actor's learning rate	β	[0.05, 0.2[
Exploration rate	σ	[0.2, 0.5[
Discount factor	γ	[0.75, 1[

To assess the effectiveness of learning, we need a metric that integrates over the entire learning episode of one, or several, learners. We developed a metric based on the number of steps it took the learner to reach the goal. Since the initial position of the arm and goal were set randomly, the minimal required number of steps to reach the goal varied considerably. Therefore, we normalised the number of steps τ_i for the i^{th} episode by dividing them through the initial distance in action space, which in our case is the joint space, to obtain the sequence $\mathcal{T}$ of normalised steps. With it, we calculate the learning speed

$$\mathcal{LS} = |(\tau_i \,|\tau_i < 4, \tau_i \in \mathcal{T})_{i=0}^{\infty}| + 1 \qquad (1)$$

as the number of times, the normalised numbers of steps τ is below 4, plus 1 to avoid the learning speed being 0. If the learner reached the goal quickly and frequently, the learning speed was high, if it reached the goal tardily and infrequently, the learning speed was low.

To determine whether the learner actually learned anything we compared its behaviour against a dummy which did not learn and stayed with the initial, random behaviour. The dummy performed the behaviour commanded by a randomly initialized MLP controller.

7. RESULTS

We randomly selected 22 different hyperparameter values uniformly distributed from Table 1 to compare interactive with non-interactive learning in the task with the one DoF arm and compared them against a non-learning dummy, see Figure 9. In this section we use the blue colour for all results from non-interactive learners and we use the red colour for all results from interactive learners. Overall, both learners learned equally well and better than the dummy. In a few cases has the non-interactive learner a higher learning speed, as defined in equation (1), and learned more reliably than the interactive one. The fastest learner, denoted by the label "f" and the yellow background, is also non-interactive. The difference between both learners is generally small, except for the instance with the largest difference which we marked with an "l".

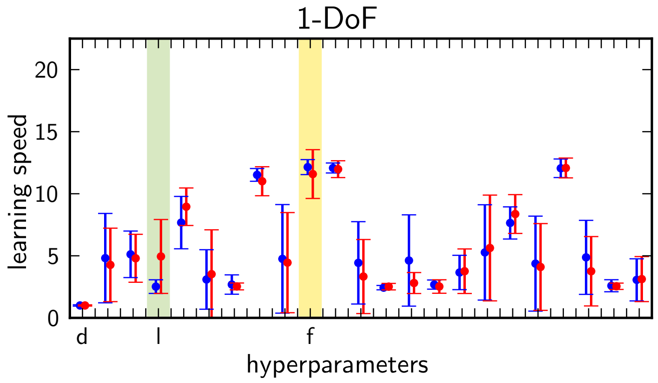

Fig. 9. There is no large difference between the interactive and non-interactive learners in the set-up with the 1-DoF arm. The indexes highlight the dummy "d", fastest learner "f" and largest difference between the interactive and non-interactive learner "l". The error bars show the standard deviation of the 10 repetitions. The red dots show the interactive and the blue dots the non-interactive learner.

Since both learners are comparatively equal in their learning speed and reliability, we take a closer look only at the hyperparameter value with the largest difference. Figure 10 shows the number of normalised steps, averaged through the repetitions, it took the agent to reach the goal throughout the test episodes.

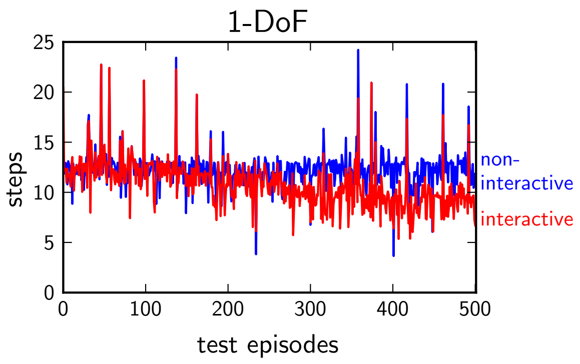

Fig. 10. Even with a large difference in the learning speed, highlighted in green in Figure 9, did interactive learning a little better than non-interactive learning with the 1-DoF arm. The red line shows the progress of the interactive and the blue line of the non-interactive learner.

For the task with the 2-DoF arm, we also randomly selected 22 different hyperparameter values from Table 1 and compared the interactive and non-interactive learner with each other against a dummy. Figure 11 shows that in this set-up interactive learning is always faster than non-interactive learning. The hyperparameter configuration with the fastest interactive learner also brought the fastest non-interactive learner.

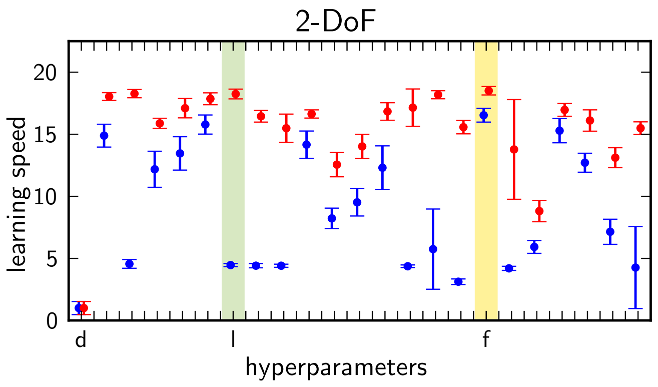

Fig. 11. Interactive learning is always faster than non-interactive learning in the task with the 2-DoF arm. The visualisation follows the same pattern as the previous one in Figure 9.

In a hyperparameter configuration with a large difference between interactive and non-interactive learning is the advantage of interaction clearly visible, as Figure 12 illustrates. The non-interactive learner learned very little, if anything at all, but the interactive learner performed distinctively better with the same hyperparameter values and initial controllers.

Figure 13 shows the learning performance of the fastest learners, marked with an "f" and a yellow background in Figure 11, in the 2-DoF task. Here, both learners learned very well and there is no real difference between them.

Similarly, as the for the previous two set-ups did we also select 22 different, random hyperparameter values from Table 1, for the 3-DoF arm. We then compared the interactive and the non-interactive learner with each other

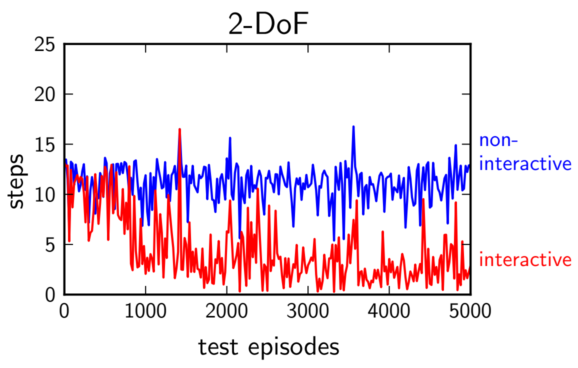

Fig. 12. The interactive learner requires less steps than the non-interactive learner to reach the test target positions in the task with the 2-DoF arm. The figure shows only every 20th episode for readability. The colour code is the same as in Figure 10.

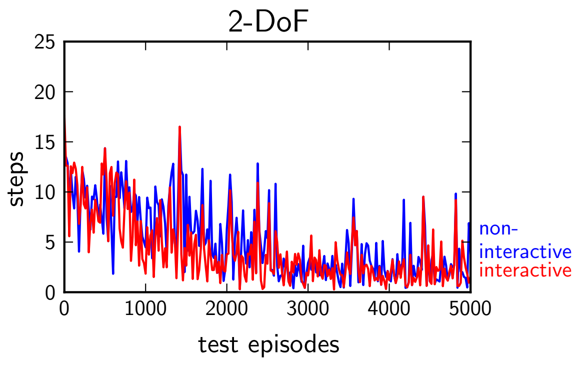

Fig. 13. The fastest interactive and non-interactive learners learned equally well in the task with the 2-DoF arm. As in Figure 12, this figure only shows every 20th testing episode.

against a dummy, see Figure 14. As in the case of the 2-DoF arm, interaction also facilitated faster learning with poorly tuned hyperparameter values. The 3rd DoF, therefore did not change the robustness of the interactive learner.

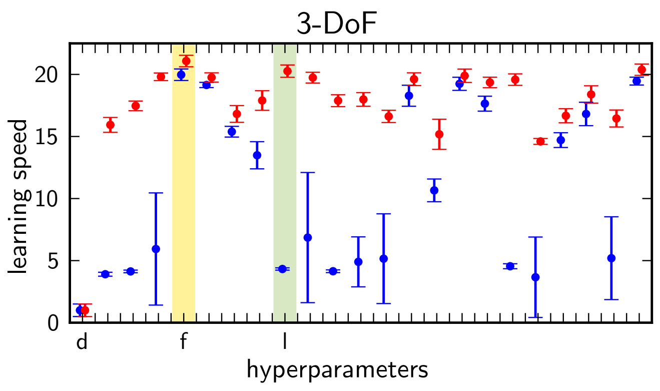

Fig. 14. Interactive learning is always faster than non-interactive in the task with the 3-DoF arm. The visualisation follows the same pattern as the previous ones in Figure 9 and Figure 11.

A closer look at the learners with the largest difference reiterates the advantage of interactive learning, as Figure 15 shows. The difference between interactive and non-interactive learning with well-chosen hyperparameter values is again as small as for the task with the 2-DoF arm.

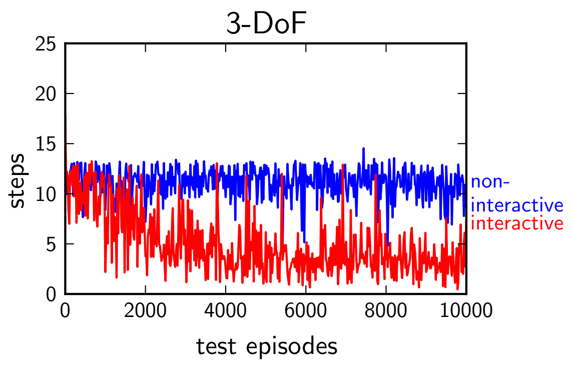

Fig. 15. The interactive learner learned much better than the non-interactive with the hyperparameters marked with an l in Figure 14 in a task with 3-DoF. The figure shows only every 20th episode for readability.

Figure 16 summarizes the development of learning speed for different likelihoods of asking for the three environments. We chose the hyperparameter values with the largest difference in the learning speed to ascertain how much interaction is necessary to have such a large difference. In contrast to the previous results in Figure 9, interaction did not speed up learning but increased the reliability, which can be seen by the reduced errors in Figure 16. In the task with 2-DoF, however, a likelihood of asking of only 10% already shows an improvement in the learning speed. The largest changes are visible from 0% to 30% of the likelihood, after which the learning speed converges. In the case of the 3-DoF the learner needed more interaction in comparison to the 2-DoF task and generally showed a less stable behaviour, until using the highest possible level of interaction.

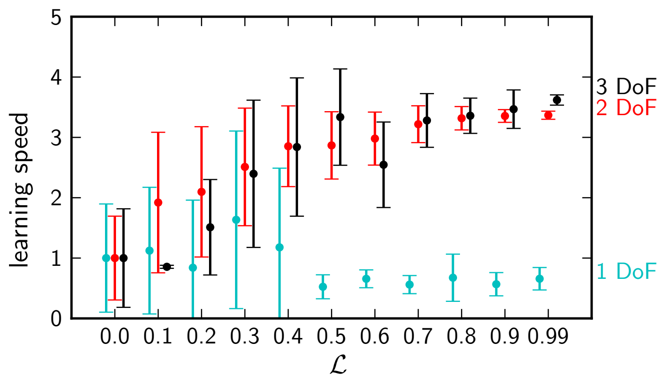

Fig. 16. The learning speed increases asymptotically with the likelihood of asking $\mathcal{L}$ in the task with the 2-DoF arm (red) and 3-DoF arm (black); the interactive feedback has no significant effect on the task with the 1-DoF arm (cyan).

8. DISCUSSION

The advantage of interactive learning seems to apply only to complex settings, which is clearly visible for the task with the 2-DoF arm as well as for the task with the 3-DoF arm. The task with the 1-DoF arm showed almost no difference between interactive and non-interactive learning. The advantage of interaction is the same with the task with the 3-DoF arm as it is for the task with the 2-DoF arm. This, however, may be attributed to the small sample size of 22 hyperparameter settings and 11 different values for the likelihood of asking, or the small difference in complexity. We needed ten times as many episodes for the 2-DoF task than for the 1-DoF task, yet the 3-DoF task needs only twice as many episodes compared to the 2-DoF task. Although the learning speed's increase was small from the 2-DoF to the 3-DoF task, the interactive learner learned more robustly than the non-interactive learner. This effect can be observed in the reported learning speeds and particularly in the frequency of large deviations from the mean for non-interactive learners when compared to interactive learners, see Figure 11 and Figure 14. Another possible reason is the low amount of feedback used, i.e. only indicating whether the action was good or bad without providing any hint towards a better action. At least the latter point needs further exploration before judging the advantages of interactive learning in complex environments. Repeating every experiment 10 times may also not be enough to accurately demonstrate the effect of interactive feedback.

Our experiments also show that interaction makes learning more robust, especially with poorly tuned hyperparameters, as shown in Figure 11 and Figure 14, where the interactive learner outperforms the non-interactive learner with every hyperparameter setting (except for the dummy). This is useful not only for simple problems as shown here, but also for more interesting complex systems where, even with expert knowledge, it is difficult to find adequate hyperparameter values.

We left it for further study to investigate the effect of feedback over time as learning progresses. Our expectation is that interaction has a greater impact during early learning when the learner knows very little, thus requiring less and less feedback as the learner gains more experience. A simple improvement on the interactivity could be to avoid asking the teacher repeatedly in similar states, thus avoiding feedback that would not help because the learner already knows a good action. In cases of more complicated evaluations of the states quality, a teacher with an internal model of the learners current knowledge could avoid unnecessary interaction [Wood et al., 1976]. More sophisticated improvements may incorporate elements of social cooperation such as trust, competence, cooperativeness, believability, emotions and intention recognition [Kulms et al., 2015].

9. CONCLUSION

We showed that interactive learning, at least in its simplest form, does not provide a clear advantage when working with simple problems such as learning to reach with a 1-DoF arm. However, we showed a measurable benefit in the case of a 2-DoF and a 3-DoF arm. We hypothesise that the benefit of interactive feedback may be more relevant or even indispensable for more complex tasks in a higher dimensional Cartesian space.

The simplistic feedback based on the Euclidean distance may be imprecise if judged by human teachers, but it is easy to automate, albeit sometimes leading to wrong feedback. This was the case even in our simple 2-DoF scenario, where "closer to goal is better" is incorrect in a few situations. Erroneous feedback may even be delivered by a domain expert, who may be consistently wrong in certain situations, thus learners cannot depend on it completely. Hence, we had to limit the likelihood of asking to $\mathcal{L} < 1$ and allow additional exploration. But as our results show, already a small likelihood of asking speeds up learning in our set-ups.

We also found that in the case of well-tuned parameters the advantage of interaction becomes small. However, this effect may be attributed to the simplistic feedback used or may just be a coincidence. The role of the quality of the feedback should also be further explored to obtain a better evaluation of the benefits and types of scenarios where interactive learning is useful.

REFERENCES

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(5):834–846, 1983. ISSN 0018-9472. doi: 10.1109/TSMC.1983.6313077.

M. Duguleana, F. G. Barbuceanu, A. Teirelbar, and G. Mogan. Obstacle Avoidance of Redundant Manipulators Using Neural Networks Based Reinforcement Learning. *Robotics and Computer-Integrated Manufacturing*, 28(2): 132–146, 2012. ISSN 0736-5845. doi: 10.1016/j.rcim.2011.07.004.

I. Farkaš, T. Malík, and K. Rebrová. Grounding the Meanings in Sensorimotor Behavior Using Reinforcement Learning. *Frontiers in Neurorobotics*, 6(1), 2012. doi: 10.3389/fnbot.2012.00001.

S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. L. Thomaz. Policy Shaping: Integrating Human Feedback with Reinforcement Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2625–2633, Lake Tahoe, NV, USA, 2013. Curran Associates, Inc.

P. Kulms, N. Mattar, and S. Kopp. Modeling Decision Making in Cognitive Systems for Social Cooperation Games. In *4. Interdisziplinärer Workshop Kognitive Systeme: Mensch, Teams, Systeme und Automaten*, pages 151–157, Bielefeld, Germany, 2015.

N. Navarro-Guerrero, C. Weber, P. Schroeter, and S. Wermter. Real-World Reinforcement Learning for Autonomous Humanoid Robot Docking. *Robotics and Autonomous Systems*, 60(11):1400–1407, 2012. ISSN 0921-8890. doi: 10.1016/j.robot.2012.05.019.

J. Peters, S. Vijayakumar, and S. Schaal. Natural Actor-Critic. In *Proceedings of the European Conference on Machine Learning (ECML)*, volume 3720 of *LNCS*, pages 280–291, Porto, Portugal, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-29243-2 978-3-540-31692-3. doi: 10.1007/11564096_29.

D. Rasmussen and C. Eliasmith. A Neural Model of Hierarchical Reinforcement Learning. In *Proceedings of the Annual Conference of the Cognitive Science Society*, pages 1252–1257, Québec City, Canada, 2014. Cognitive Science Society. ISBN 978-0-9911967-0-8.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.

W. Schultz. Getting Formal with Dopamine and Reward. *Neuron*, 36(2):241–263, 2002. ISSN 0896-6273. doi: 10.1016/S0896-6273(02)00967-4.

D. R. Shanks, R. J. Tunney, and J. D. McCarthy. A Re-Examination of Probability Matching and Rational Choice. *Journal of Behavioral Decision Making*, 15(3): 233–250, 2002. ISSN 1099-0771. doi: 10.1002/bdm.413.

C. Stahlhut. *Learning to Reach with Interactive Reinforcement Learning*. MSc, Universität Hamburg, Hamburg, Germany, 2014.

C. Stahlhut, N. Navarro-Guerrero, C. Weber, and S. Wermter. Interaction Is More Beneficial in Complex Reinforcement Learning Problems Than in Simple Ones. In *4. Interdisziplinärer Workshop Kognitive Systeme: Mensch, Teams, Systeme und Automaten*, pages 142–150, Bielefeld, Germany, 2015.

H. B. Suay and S. Chernova. Effect of Human Guidance and State Space Size on Interactive Reinforcement Learning. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1–6, Atlanta, GA, USA, 2011. IEEE. ISBN 978-1-4577-1571-6 978-1-4577-1572-3. doi: 10.1109/ROMAN.2011.6005223.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive computation and machine learning. A Bradford Book/The MIT Press, Cambridge, MA, USA, first edition, 1998. ISBN 0-262-19398-1.

E. Theodorou, J. Buchli, and S. Schaal. Reinforcement Learning of Motor Skills in High Dimensions: A Path Integral Approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2397–2403, Anchorage, AK, USA, 2010. IEEE. ISBN 978-1-4244-5040-4, 978-1-4244-5038-1, 1050-4729. doi: 10.1109/ROBOT.2010.5509336.

A. L. Thomaz. *Socially Guided Machine Learning*. Phd, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006.

A. L. Thomaz and C. Breazeal. Teachable Robots: Understanding Human Teaching Behavior to Build More Effective Robot Learners. *Artificial Intelligence*, 172(6-7):716–737, 2008. ISSN 0004-3702. doi: 10.1016/j.artint.2007.09.009.

E. L. Thorndike. *Animal Intelligence: Experimental Studies*. The Macmillan Company, New York, NY, USA, 1911.

H. van Hasselt and M. A. Wiering. Reinforcement Learning in Continuous Action Spaces. In *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 272–279, Honolulu, HI, USA, 2007. IEEE. ISBN 1-4244-0706-0. doi: 10.1109/ADPRL.2007.368199.

C. J. C. H. Watkins. *Learning from Delayed Rewards*. Phd, Cambridge University, 1989.

R. J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992. ISSN 0885-6125, 1573-0565. doi: 10.1007/BF00992696.

D. Wood, J. S. Bruner, and G. Ross. The Role of Tutoring in Problem Solving. *Journal of Child Psychology and Psychiatry*, 17(2):89–100, 1976. doi: 10.1111/j.1469-7610.1976.tb00381.x.