Interactive Learning of Continuous Actions from Corrective Advice Communicated by Humans

Carlos Celemin(✉) and Javier Ruiz-del-Solar

Department of Electrical Engineering and Advanced Mining Technology Center, Universidad de Chile, Santiago, Chile
{carlos.celemin, jruizd}@ing.uchile.cl

Abstract. An interactive learning framework that allows non-expert humans to shape a policy through corrective advice, using a binary signal in the action domain of the robot/agent, is proposed. One of the most innovative features of COACH (COrrective Advice Communicated by Humans), the proposed framework, is a mechanism for adaptively adjusting the amount of human feedback that a given action receives, taking into consideration past feedback. The performance of COACH is compared with the one of TAMER (Teaching an Agent Manually via Evaluative Reinforcement), ACTAMER (Actor-Critic TAMER), and an autonomous agent trained using SARSA(λ) in two reinforcement learning problems: ball dribbling and Cart-Pole balancing. COACH outperforms the other learning frameworks in the reported experiments. In addition, results show that COACH is able to transfer successfully human knowledge to agents with continuous actions, being a complementary approach to TAMER, which is appropriate for teaching in discrete action domains.

Keywords: Robot learning · Interactive learning · Human teachers · Human feedback in action domains · Ball dribbling · Robot soccer

1 Introduction

The use of computational/machine learning techniques such as Reinforcement Learning (RL) allows robots, and agents in general, to address complex decision-making tasks. However, one of the main limitations of the use of learning approaches in real-world problems is the large number of learning trials required to learn complex behaviors. This can make prohibitive the use of many learning approaches in problems such as autonomous driving, x-copter flight control, or soccer robotics, where the implementation of learning trials with real robots, in the real world, may have a high cost.

This drawback can be addressed by using human feedback during learning, i.e., the learning process can be assisted by a human teacher who supervises the agent-environment interaction. There are two main schemes for using human feedback in order to modify the policy of a learning agent: a first one in which the trainer indicates to the agent what to do, i.e., human feedback in the actions domain [1–6]; and a second one in which the trainer evaluates the actions through rewards and punishments, i.e., human feedback in the evaluative domain [7–14].

© Springer International Publishing Switzerland 2015
L. Almeida et al. (Eds.): RoboCup 2015, LNAI 9513, pp. 16–27, 2015.
DOI: 10.1007/978-3-319-29339-4_2

In this context, the main goal of this article is to propose COACH (COrrective Advice Communicated by Humans), a new interactive learning framework that borrows elements from both schemes. COACH is inspired by the Shaping paradigm [16], which allows to interactively train an agent through signals of positive and negative reinforcement. But, as in the *Advice Operators* paradigm [6], the human feedback indicates the agent how the action has to be modified (increased or decreased).

Thus, COACH is based on the TAMER framework [15], but instead of using human feedback for evaluating the results of the action as TAMER does, human feedback is given in the action domains as in the Advice Operators paradigm [6], but without using an off-line and data-driven based supervised learning process. COACH manages the human feedback and the interactive update of the policy in a similar way that TAMER. One of the most innovative features of COACH is a mechanism for adaptively adjusting the amount of human feedback that a given action receives, taking into consideration past feedback. An additional feature of COACH is the possibility of providing human-feedback using either a keyboard or hand-gesture interface during training.

The proposed learning framework is validated and compared with TAMER, ACTAMER [17], and a classical SARSA(λ) method, in two problems: (i) ball dribbling with humanoid robots [18], and (ii) the very well-known Cart-Pole problem [19]. COACH outperforms all other learning frameworks in the reported experiments.

The paper is organized as follows. In Sect. 2 the COACH learning framework is presented, and in Sect. 3 the hand-gesture visual interface is described. The experimental validation and conclusions are given in Sects. 4 and 5, respectively.

2 The COACH Learning Framework

The COACH learning framework uses human binary feedback as a correction in the action domain, in order to update the current policy for the state wherein that action was executed. The trainer has to provide its feedback immediately after the execution of the action to be learnt.

COACH lets the trainer to shape the policy of an agent through occasional feedback. The method updates a policy model based on a supervised learning strategy supported by four main modules: *Human Feedback Modeling*, which characterizes the sequence of pieces of human advice, and determines how much feedback must be added to the executed action; *Policy Supervised Learner* and *Human Feedback Supervised Learner*, which updates the parameters of the policy model and the human feedback model, respectively; and *Credit Assigner*, which handles the time delay of human feedback (see Fig. 1).

In COACH, when the *Policy* module observes the state vector $\vec{s}$, it executes a continuous action $P(\vec{s})$ according to the policy model $P : S \rightarrow \mathbb{R}$ (this is a difference regarding the TAMER modeling, which bases the policy on a human trainer's reinforcement function from the state-action space $H_{TAMER} : S \times A \rightarrow \mathbb{R}$). Then, the human trainer observes the effect of the action in the environment, and gives an advice h. The signal h is the binary feedback (+1 or −1), which states how the current executed action has to be modified for that state $\vec{s}$ (increase or decrease its value, respectively).

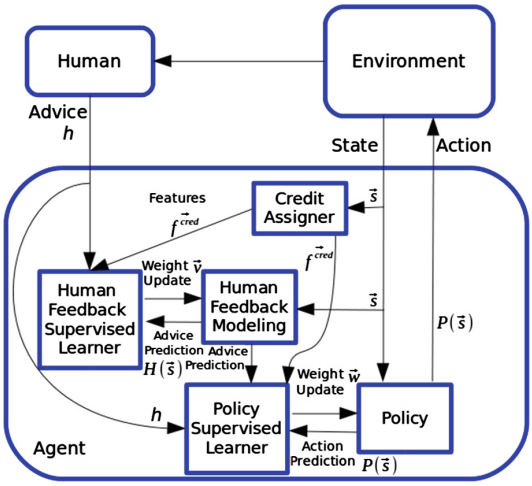

Fig. 1. Block diagram of the COACH learning framework.

The state $\vec{s}$, the executed action $P(\vec{s})$, and the human feedback h, are taken by the learning modules for updating the parameters of P (weights vector $\vec{w}$). Then, in the next time step, P has a new parameters set. When the trainer does not provide any feedback signal, h is taken as zero. The trainer is only allowed to give a binary correction, because COACH works under the assumption that a person cannot estimate the exact value of an appropriate correction, human just provide a trend of the modification (e.g. more/less force, velocity, energy, etc.).

Policy: The policy model P can be implemented using any function approximator. COACH uses a linear model of Gaussian features as TAMER and continuous SARSA (in general the Radial Basis Functions (RBF) features are the most used for function approximation in RL [19, 20]). The expression for $P(\vec{s})$ is the inner product between the $\vec{w}$ vector and the features vector $\vec{f}$ mapped by the Gaussian Kernels from the state space described by the state vector $\vec{s}$:

$$P(\vec{s}) = \vec{w}^T \cdot \vec{f} \tag{1}$$

The $\vec{w}$ vector is updated using a gradient descend approach as:

$$\Delta w_l = \alpha \cdot error \cdot \frac{\partial P(\vec{s})}{\partial w_l} = \alpha \cdot error \cdot f_l \tag{2}$$

$$error = h \cdot e \tag{3}$$

with e a constant error magnitude, h the sign of the error, given by he human feedback signal, and l the weight's/feature's index.

Human Feedback Modeling and Supervised Learner: The trainer intentions, observed in the binary feedback signal, can be considered a source of information that

not only provides the sign (direction) of the corrections, but also its magnitude. Hence, in the COACH framework a model of the human feedback $H : S \rightarrow \mathbb{R}$ is built, which characterizes the human feedback signal over each region of the state space. As in the case of the policy model P, a linear parameterization of RBF features is used for representing the human feedback model H. Therefore, two Supervised Learner modules are required in the framework, one for P and one for H (see Fig. 1).

In the proposed modeling, sequences of feedback signals with constant sign over a specific state ($\overrightarrow{s_a}$), would mean the trainer suggest a large change of the magnitude of the associated action $P(\overrightarrow{s_a})$. On the other hand, sequences of alternating values of the sign in the human feedback would mean that the trainer is trying to provide a finer change around a given set point. Thus, using the information of H for computing an adaptive learning rate is appropriate for avoiding the dilemma of setting the magnitude of the error e either large (it does not let to perform fine adjustments) or small (it does not let to carry out large corrections quickly).

The model of the human feedback H, is built using the same features vector $\vec{f}$ of P, and a weight vector $\vec{v}$ as:

$$H(\vec{s}) = \vec{v}^T \cdot \vec{f} \tag{4}$$

Both P and H models map the same state space, and are based on the same kind of function approximator. Also, their respective *Supervised Learners* use the human feedback signal for updating the parameters. However, the H model is updated using the prediction error based on the difference of h and $H(\vec{s})$. Therefore, using a gradient descend approach, the weights associated to the H model are updated as:

$$\Delta v_l = \beta \cdot (h - H(\vec{s})) \cdot \frac{\partial H(\vec{s})}{\partial v_l} = \beta \cdot \left(h - \vec{v}^T \cdot \vec{f}\right) \cdot f_l \tag{5}$$

with β the learning rate and l the weight's/feature's index.

Then, the adaptive learning rate of the policy model learning process is computed as:

$$\alpha(\vec{s}) = |H(\vec{s})| + bias = \left|\vec{v}^T \cdot \vec{f}\right| + bias \tag{6}$$

where *bias* is the default value of the learning rate.

The magnitude of $|H(\vec{s})|$ is close to 1 when most of the last human feedback signals for a specific state $\overrightarrow{s_a}$ have the same value (either +1 or −1). On the contrary, alternating values of the feedback signal decrease the magnitude of $|H(\vec{s})|$. Hence, $\alpha(\vec{s})$ is set to a large value when feedback signals of constant value are received, and $\alpha(\vec{s})$ is set to a smaller value when feedback signals of alternating value are received.

Finally, the weights associated to the P model are now updated using $\alpha(\vec{s})$ as:

$$\Delta w_l = \alpha(\vec{s}) \cdot error \cdot \frac{\partial P(\vec{s})}{\partial w_l} = \alpha(\vec{s}) \cdot error \cdot f_l \tag{7}$$

with l the weight's/feature's index.

Credit Assigner: The corrective advice has to be given after the agent executes each action. But in decision-making problems of high frequency, a human trainer is not able to assess the effect of each action at each time step, this produce a delay between the action execution and the human response. The Credit Assigner proposed in TAMER, approaches this problem by associating the feedback not only to the last state-action pair, but to a past window of pairs. Each state-action pair is weighted with the corresponding probability that characterizes the human delay. COACH uses the TAMER's credit assigner; hence this module computes a new features vector $\overrightarrow{f^{cred}}$ for replacing the original vector $\vec{f}$ in the *H* and *P* update process.

Complete Algorithm: The *H* model is used for supporting the *Policy Supervised Learner* module, which updates the *P* model. The *Credit Assigner* module takes the states vector and computes credit assignments based on the history of past states. The Supervised Learners modules do not read directly the state vector $\vec{s}$, but instead take the features vector provided by the *Credit Assigner* module. This features vector is the average weighted sum of the past features.

Algorithm 1: Learning from Corrective Advice of a human trainer, using an adaptive learning rate for the policy update (complete framework).

$e \leftarrow constant$ // error magnitude
$\beta \leftarrow constant$ // learning rate
$bias \leftarrow constant$ // offset for α
while *true* **do**
  $\vec{s} \leftarrow getStateVec()$
  $\overrightarrow{f_1} \leftarrow getFeatures(\vec{s})$
  $P(\vec{s}) \leftarrow \vec{w}^T \cdot \overrightarrow{f_1}$
  $TakeAction(P(\vec{s}))$
  wait for next time step
  $h \leftarrow gethumanCorrectiveAdvice()$
  for $1 \leq t \leq T$
    $c_t \leftarrow assignCredit(t)$
    $\overrightarrow{f^{cred}} = \overrightarrow{f^{cred}} + (c_t \cdot \overrightarrow{f_t})$
  end for
  if $h =! 0$
    $H(\vec{s}) = \vec{v}^T \cdot \overrightarrow{f^{cred}}$
    $\Delta v_l = \beta \cdot (h - H(\vec{s})) \cdot f_l^{cred}; l = 1, .., N_{feat}$
    $v_l = v_l + \Delta v_l$
    $\alpha(\vec{s}) = | H(\vec{s}) | + bias = | \vec{v}^T \cdot \overrightarrow{f^{cred}} | + bias$
    $error \leftarrow h \cdot e$
    $\Delta w_l = \alpha(\vec{s}) \cdot error \cdot f_l^{cred}; l = 1, .., N_{feat}$
    $w_l = w_l + \Delta w_l$
  end if
end while

Algorithm 1 presents the whole learning process. First, the error magnitude e and the learning rate β for the weights update and the *bias* of (6) are defined (lines 1–3). The loop between lines 4–22 occurs once per time step. The agent observes the new state $\vec{s}$ (line 5), and it computes the basis functions/features of the policy model $P(\vec{s})$ (line 6). Thus, the agent takes $\vec{s}$ from the state space and maps it into the feature-space for obtaining the features vector $\vec{f}$.$P(\vec{s})$ is obtained as the inner product between $\vec{f}$ and the weight's vector $\vec{w}$ (line 7). Afterwards, $P(\vec{s})$ is executed (line 8).

After action execution (lines 8–9), the human trainer sends a feedback signal h (line 10). Then the credit assigner computes the vector $\overrightarrow{f^{cred}}$ (lines 11–14). For the credit assignment, the index t is varied according with the amount of past time steps T that compound the history window (line 11); c_t obtains the credit associated with the t^{th} prior time step (line 12), which depends on the probability density function used for modeling the human delay. $\overrightarrow{f_t}$ is the features vector computed in line 6, t time steps ago, which is weighted by the respective c_t and cumulated in $\overrightarrow{f^{cred}}$ (line 13).

Afterwards, the human feedback model H (lines 16–17) is updated, and the value of the variable learning rate is calculated (line 18). Finally, the policy model P is updated (line 19–20): first the prediction *error* of P, whose magnitude/sign is given by e/h is computed (line 19), and then, the weights of $P(\vec{s})$ are updated (line 20).

3 Human-Machine Interface

Given that human feedback is a key component of the proposed learning framework, a new hand-gesture interface that allows providing feedback to the agent, is proposed for validating and contrasting the learning frameworks' performances through this interface and a keyboard based interface. The interface allows detecting 5 gestures (positive correction, negative correction, a neutral gesture used when users do not need to provide feedback, a reward, and a punishment) in dynamic setups (variable illumination, non-uniform backgrounds). It uses background subtraction to detect regions of interest (ROI), i.e. hand candidates, Kalman filtering for tracking the hand candidates, and Local Binary Patterns (LBP) and SVM for the final detection of hand-gestures (see Fig. 2). These three functionalities are described in the following paragraphs:

– *Detection of Regions of Interest (ROI):* Movement blobs are first detected using background subtraction. Then, adjacent blobs are merged and filtered using morphological filters, and the largest blob is selected as a hand candidate and feed to the tracking system.

In parallel, a second process applies background subtraction to color edges: First, a binary edge image is computed, and then, color information is incorporated into the edges. Afterwards, background subtraction and area filtering is applied in the edge's domain. Finally, the output of the area-filtering module is intersected with the color edges, in order to manage occlusions properly (see Fig. 2(b)). The output is a blob with the detected moving, color edges.

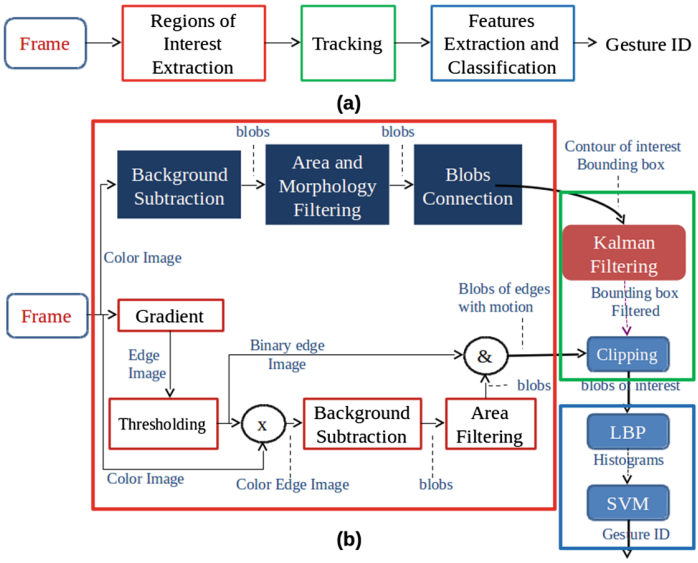

Fig. 2. Hand Gesture Recognition system. (a) General scheme, (b) detailed scheme.

- *Tracking:* The parameters of the bounding box given by the prior module are used as observations by a Kalman filter, which estimates the final hand candidates, based on the fusion of the current ROI information and the prior ones. Afterwards, the blob with the moving edges is intersected with the Kalman-filtered bounding box.
- *Features Extraction and Classification:* The image window given by the *Tracking* module is analyzed in order to classify the captured gesture. Histograms of LBP features are computed inside the image window. This feature vector feeds five SVM classifiers, one trained for each gesture, where the gestures are detected.

The described system is able to detect gestures with a mean detection rate of 91 %. A more detailed description of this system is given in [21].

4 Experiments and Results

The performance of the COACH framework is validated and compared with the ones of TAMER [15], ACTAMER [17] (an Actor Critic approach based on TAMER) which are pure interactive machine learning algorithms like COACH, and an autonomous agent trained using SARSA(λ). Two learning problems are used for the comparisons: ball dribbling with Humanoid robots [18] and the Cart-Pole [19]. In the first problem, 10 subjects interacted with each interactive learning framework (COACH, TAMER and ACTAMER) only through a keyboard interface, while in the second problem 15 subjects trained an agent using each framework, first with the keyboard interface, and then with the proposed hand-gesture interface. In this second experiment, the goal is to evaluate the use of these two alternative interfaces.

For each problem to be solved, the subjects interacted with each learning framework four times: the first and second trials are used for practicing (human operator

training), while the other two for the interactive training of the agent. In the case of the SARSA(λ) method, 50 training runs were executed.

Although, only the SARSA(λ) agent uses an environmental reward function during learning, the cumulative environmental reward and the average environmental reward per episode were computed and used as performance metrics.

4.1 Ball Dribbling with Humanoid Robots

In the context of humanoid robotics soccer, ball dribbling is a relevant problem in which a robot walks to a target as fast as possible by keeping the ball possession. Keeping the ball possession means having the ball close to the robot's feet while walking (see a description of this behavior in [18]). In a recent work this problem has been modeled as two simpler tasks: ball pushing, and alignment to the ball and target. The first task reduces the problem to dribbling on a straight line (one dimension), and it is tackled using autonomous RL [18]. An episode is completed when the robot goes across the complete soccer field with the ball. For dribbling the ball in a straight line the robot estimates the distance to the ball ρ, and decides its speed. Therefore this problem has a very small state space but with high level of uncertainty, due to the fact that the feet motion is not observed by the decision-making system of the robot.

This work proposes a modification of the reward function used in [18], consisting on incorporating a parameter that defines a security robot-ball distance ρ_{max} that must not be exceeded:

$$r = \begin{cases} 100 + v_x, & \rho \leq \rho_{max} \\ -100 - (\rho - \rho_{max}) + v_x, & \rho > \rho_{max} \end{cases} \tag{8}$$

This reward function re defines the ball-dribbling task: the goal is to walk as fast as possible, without exceeding the distance ρ_{max}.

The robot speed v_x is set between 0 and 100 mm/s. For the algorithms with discretization of the action space (SARSA and TAMER) ten different magnitudes of speed were defined; the state space was divided uniformly in thirty features between 0 and 3 m, and the ρ_{max} parameter was set to 300 mm.

In this problem the gesture recognition interface was not used due to its computational burden. The obtained results are shown in Fig. 3. As it can be observed in this

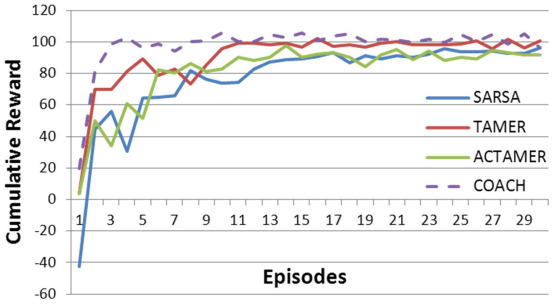

Fig. 3. Average cumulative reward for the ball dribbling problem.

figure, the three interactive learning algorithms converged faster than the non-interactive one (SARSA), and COACH achieves the fastest convergence (in three episodes), and the highest average performance. The second best convergence was obtained by TAMER.

4.2 Cart-Pole

This well known problem in RL literature is an episodic task with the goal of keeping balanced a pole on top of a cart. The actions are the forces applied by the cart; the state space has four dimensions defined by the cart's position and velocity, and the pole's angle and angular velocity [19]. An episode is finished (a failure occurs) when the pole falls to a given angle regarding the vertical axis, or if the cart exceeds the bounds of the scene. In our modeling, the continuous, 4-dimensional state space is approximated and divided uniformly using 256 Gaussian RBF features as in [17]. This problem has been approached by TAMER and ACTAMER in their original papers. All learning frameworks were tested under the same conditions.

In the experiments with the hand gesture recognition system, with COACH are used: neutral, positive, and negative correction gestures; for TAMER and ACTAMER are used: neutral, reward and punishment gestures. In Fig. 4 are shown the obtained learning curves of this problem trained with both the keyboard and the hand-gesture interfaces. The experiments with interactive agents were finished in the episode 150 with a maximum of allowed time steps of 5,000, considered here as the optimum performance. The TAMER and ACTAMER algorithms reach the lowest performances, although these algorithms achieve faster learning than SARSA in the early episodes (by the episode 20). The autonomous SARSA agent converges by the episode 500 with higher final performance than the ones achieved by TAMER and ACTAMER. COACH outperforms the other algorithms since the first episode, and it achieves a performance four and three times higher than the one of SARSA's (the second highest performance) with the keyboard and the hand-gesture interfaces, respectively. In addition COACH achieves the fastest convergence among all the agents; the users only need almost 25 episodes for teaching the policies.

The use of the keyboard interface allowed obtaining better results. Despite the hand-gesture interface is a more natural way of communication for human-machine interaction, the keyboard interface was easier for most of the users because it allowed them to alternate the signals given to the agent at a higher speed.

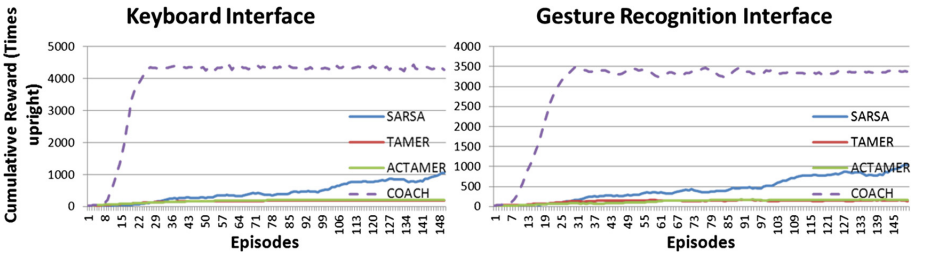

Fig. 4. Average cumulative reward for the cart-pole problem.

Table 1 summarizes the obtained results. It can be observed that COACH allows obtaining a faster convergence than all other methods, and also a higher performance. In the case of the Cart-Pole problem, the final performance obtained by COACH is much higher than the ones obtained by TAMER or ACTAMER, which is very low even compared to the one obtained when using SARSA.

Table 1. Algorithms comparison. EC: Number of episodes for convergence. AP: Average of final performance of the last 5 episodes.

	Ball dribbling		Cart-pole (keyboard)		Cart-pole (hand-gestures)	
	EC	AP	EC	AP	EC	AP
TAMER	11	96.2	89	188	40	150
ACTAMER	11	93.8	68	196	79	166
COACH	3	104.8	25	4,317	22	3,383
SARSA	21	91.6	461	1,125	461	1,125

It is interesting to note that in [17] it was possible to attain higher performance for the Cart-Pole problem only when using hybrid methods that combine the use of type-TAMER algorithms and autonomous RL. In a first stage, those hybrid methods learn suboptimal policies from humans. Then the acquired knowledge is transferred for supporting agents with autonomous RL processes. In this work, similar high performances were obtained with faster convergence, directly by COACH agents, which are based on a pure interactive machine learning approach that uses the feedback in a more intuitive sense.

5 Conclusions

Humans assisting agents in the process of shaping a policy is a strategy very useful in problems where the environment-agent interaction could be observed by the human senses. Interactive learning algorithms have the ability to allow the transfer of knowledge from humans to machines in this kind of problems. COACH lets to a human trainer with no expertise, to shape a policy with a binary signal. COACH sets the feedback in the domain of the actions space as in Learning from Demonstration approaches, which is more natural and intuitive than the evaluative feedback domain. Due to the above and the obtained results, it is stated that COACH is an easy-to-use framework for teaching an agent that have to perform continuous actions.

Moreover, COACH models the human feedback which allows extracting more information from the human trained. Thus allows advancing on understanding how humans teach autonomous agents.

With the reported experiments we have validated the hypothesis that interactive learning frameworks for shaping the policies of learning agents may be the solution in applications where a human is able to observe the agent-environment interaction and to advice it. This is especially important because in many applications the a priori definition of the reward function is not easy to obtain for an autonomous RL agent.

Results showed that human participation in learning process is a powerful strategy that succeeds and outperforms the classical autonomous RL, depending of the kind of feedback given by humans and the problem's nature. Although TAMER algorithms have more general applications (discrete and also continuous action problems), COACH is a complementary approach to TAMER, while TAMER is appropriate for teaching in discrete domains, COACH is more suitable for continuous domain applications.

It is important to remark, that the experimental procedure allowed us to see that due to people are not experts with the tasks being solved and also no experts in teaching agents, sometimes they do mistakes giving accidentally incorrect signals (e.g. punishing actions they think are the correct one, or increasing a force they do not plan to change). Then, turning back to the prior policy is a challenge easier to achieve with COACH compared to TAMER, because with TAMER the users change the policy over a specific state, punishing the action, but they cannot control or forecast which is the new action executed by the policy in that state, whereas with COACH, the human has an insight about the changes in the policy after feeding back. The aforementioned fact has impact in the time reduction obtained in the learning process with COACH.

The difficult level of teaching a task to a machine is due to two important facts: the problem's nature, and the abilities of humans using the human-machine interface for teaching the task. In this work the difficulty associated to the interface was analyzed: a keyboard and a hand-gesture interface were used as input devices. The hand-gesture interface is more intuitive for users, but it constraints them, because when using hand-gestures it is not easy to provide different orders (i.e. different gestures) in short periods of time, as the case when using a keyboard. Nevertheless, results show that COACH allowed users to reach optimal policies too, in this last case. This leads to the conclusion that COACH is robust and reliable to be used in difficult problems where either the nature of the agent-environment interaction is complex or where the human-machine interface is complex to use.

Acknowledgements. This work was partially funded by FONDECYT Project 1130153, CONICYT-PCHA/Doctorado Nacional/2015-21151488, and the Department of Electrical Engineering, University of Chile.

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Rob. Auton. Syst. **57**(5), 469–483 (2009)
2. Breazeal, C., Scassellati, B.: Robots that imitate humans. Trends Cogn. Sci. **6**(11), 481–487 (2002)
3. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the Twenty-First International Conference on Machine Learning. ACM (2004)
4. Meriçli, C., Veloso, M., Akin, H.L.: Complementary humanoid behavior shaping using corrective demonstration. In: 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 334–339. IEEE (2010)

5. Meriçli, Ç., Veloso, M., Akin, H.L.: Task refinement for autonomous robots using complementary corrective human feedback. Int. J. Adv. Rob. Syst. **8**(2), 68 (2011)
6. Argall, B.D., Browning, B., Veloso, M.: Learning robot motion control with demonstration and advice-operators. In: IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2008, pp. 399–404. IEEE (2008)
7. Mitsunaga, N., Smith, C., Kanda, T., Ishiguro, H., Hagita, N.: Adapting robot behavior for human–robot interaction. IEEE Trans. Rob. **24**(4), 911–916 (2008)
8. Tenorio-Gonzalez, A.C., Villaseñor-Pineda, L., Morales, E.F.: Dynamic reward shaping: training a robot by voice. In: Kuri-Morales, A., Simari, G.R. (eds.) IBERAMIA 2010. LNCS, vol. 6433, pp. 483–492. Springer, Heidelberg (2010)
9. León, A., Morales, E.F., Altamirano, L., Ruiz, J.R.: Teaching a robot to perform task through imitation and on-line feedback. In: San Martin, C., Kim, S.-W. (eds.) CIARP 2011. LNCS, vol. 7042, pp. 549–556. Springer, Heidelberg (2011)
10. Suay, H.B., Chernova, S.: Effect of human guidance and state space size on interactive reinforcement learning. In: RO-MAN 2011, pp. 1–6. IEEE (2011)
11. Pilarski, P.M., Dawson, M.R., Degris, T., Fahimi, F., Carey, J.P., Sutton, R.S. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In: IEEE International Conference on Rehabilitation Robotics (ICORR), pp. 1–7. IEEE (2011)
12. Yanik, P.M., Manganelli, J., Merino, J., Threatt, A.L., Brooks, J.O., Green, K.E., Walker, I.D.: A gesture learning interface for simulated robot path shaping with a human teacher. IEEE Trans. Hum.-Mach. Syst. **44**, 41–54 (2014)
13. Thomaz, A.L., Hoffman, G., Breazeal, C.: Reinforcement learning with human teachers: understanding how people want to teach robots. In: The 15th IEEE International Symposium on Robot and Human Interactive Communication, ROMAN 2006, pp. 352–357. IEEE (2006)
14. Thomaz, A.L., Breazeal, C.: Asymmetric interpretations of positive and negative human feedback for a social learning agent. In: The 16th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2007, pp. 720–725. IEEE (2007)
15. Knox, W.B., Stone, P.: TAMER: training an agent manually via evaluative reinforcement. In: 7th IEEE International Conference on Development and Learning, ICDL 2008, pp. 292–297. IEEE (2008)
16. Knox, W.B., Stone, P.: Interactively shaping agents via human reinforcement: the TAMER framework. In: Proceedings of the Fifth International Conference on Knowledge Capture, pp. 9–16. ACM (2009)
17. Vien, N.A., Ertel, W., Chung, T.C.: Learning via human feedback in continuous state and action spaces. Appl. Intell. **39**(2), 267–278 (2013)
18. Leottau, L., Ruiz-del-Solar, J., Celemin, C.: Ball dribbling for humanoid biped robots: a reinforcement learning and fuzzy control approach. In: Bianchi, R.A., Akin, H., Ramamoorthy, S., Sugiura, K. (eds.) RoboCup 2014. LNCS, vol. 8992, pp. 549–561. Springer, Heidelberg (2015)
19. Sutton, R.S., Barto, A.G.: Reinforcement Learning: an Introduction. MIT Press, Cambridge (1998)
20. Busoniu, L., Babuska, R., De Schutter, B., Ernst, D.: Reinforcement Learning and Dynamic Programming Using Function Approximators. CRC Press, Boca Raton (2010)
21. Celemin, C.: A hand-gesture interface for interactive learning. Internal report, Advanced Mining Technology Center, Universidad de Chile (2014). (in Spanish)