Socially Guided XCS: Using Teaching Signals to Boost Learning

Anis Najar
Institut des Systèmes Intelligents et de Robotique
CNRS UMR 7222
Université Pierre et Marie Curie
Paris, France
najar@isir.upmc.fr

Olivier Sigaud
Institut des Systèmes Intelligents et de Robotique
CNRS UMR 7222
Université Pierre et Marie Curie
Paris, France
sigaud@isir.upmc.fr

Mohamed Chetouani
Institut des Systèmes Intelligents et de Robotique
CNRS UMR 7222
Université Pierre et Marie Curie
Paris, France
chetouani@isir.upmc.fr

ABSTRACT

In this paper, we show how we can improve task learning by using social interaction to guide the learning process of a robot, in a Human-Robot Interaction scenario. We introduce a novel method that simultaneously learns a social reward function on the teaching signals provided by a human and uses it to bootstrap task learning. We propose a model we call the Socially Guided XCS, based on the XCS framework, and we evaluate it in simulation with respect to the standard XCS algorithm. We show that our model improves the learning speed of XCS.

CCS Concepts

•Computing methodologies → Reinforcement learning; •Computer systems organization → External interfaces for robotics;

Keywords

Human-Robot Interaction; Interactive Reinforcement Learning; Learning Classifier Systems

1. INTRODUCTION

In addition to the autonomous capacity of humans to adapt to their physical environment, social interaction has been demonstrated to play a major role in enhancing human learning [9]. These two different aspects of learning, autonomous and social, have been for a while a source of inspiration for many works in robotics [2]. Our work aims at studying the relationship between these two aspects in the context of Human-Robot interactions. We consider a context in which a robot is endowed with the capacity to learn autonomously to accomplish a certain task by collecting rewards from its environment. In addition, the robot can get some help from a human that would provide it with teaching signals. Initially, the meaning of these signals would be unknown to the robot, so it should learn to interpret them in order to use them afterwards to learn better and faster. Our idea is to learn a model of the human (the Social Model) that would assign reward values for each teaching signal, by using the rewards coming from the environment (the Task Model). Our research question is motivated by the fact that some human feedbacks like warning, encouragement, disapproval or consent could carry some affective values. We address the question of how these affective values could be grounded in the robot's experience and how they could contribute to the robot's learning process.

To illustrate our idea, we draw an example inspired from the child development literature. We consider an infant exploring its room. When alone, the infant is able to learn to interact with its environment only by autonomous exploration, by trying different actions in different situations and by privileging actions that yield more desirable outcomes. In some situations, a parent could intervene in order to guide him by providing advice or warnings about desirable or dangerous actions. For example, if he stands close to a power plug, his mother would warn him about this situation to prevent him from putting his fingers into the plug. Suppose in that moment, the child walks away and does not get any harmful outcome. On the one hand, he will be able to update the model of his environment (Task Model) by the information that, when he is near the power plug and decides to walk away, he does not receive any bad outcomes. On the other hand, he will also be able to incorporate the information that, when his mother warns him and he does the action of walking away from the power plug, he will not get any bad outcome (Social Model). Now suppose that by the next day, the child is facing the same situation, but this time he is alone. If this time he decides to touch the power plug, the negative outcome of this action will allow him not only to update his knowledge about his environment (Task Model) but also to get a better understanding of the meaning of his mother's warnings (Social Model), that will be then associated to danger, even if she is not present, only by remembering that she warned him the last day, when he was in the same situation. So in forthcoming situations, when the child gets warned by his mother, he will be able to exploit the meaning of his mother's warnings (Social Model) in a better way in order to interact with his environment (Task Model).

GECCO'15 Companion, July 11 - 15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739482.2768452

Starting from this intuition, we designed a system we call the Socially Guided XCS (SGXCS), based on two distinct models: one related to the task (Task Model) and another related to social interaction (Social Model). These two models influence each other in the sense that the learning of the one contributes to the learning of the other. This influence is carried out through a third model (Contingency Model), that serves as a bridge, by representing the contingency between task states and teaching signals. The Social and Task models are based on XCS, an RL system endowed with a generalization capability [11]. The choice for XCS was motivated by its interesting features: the generalization property, the possibility to interpret the learned rules which could be of big interest for social interaction analysis, and the possibility to use the classifier's fitness as a confidence criterion on the reliability of the learned rules.

In the next section, we give an overview of related work. Then, in Section 3, we present our model. In Section 4, we show some experimental results about the performance of our system in simulation. In Section 5, we discuss the mechanisms of our model. Finally, we conclude this paper by providing some ideas for future work.

2. RELATED WORK

Reinforcement Learning (RL) [13] offers a variety of techniques that allow a computational agent to autonomously learn to achieve a task by interacting with its environment. Compared with supervised learning, an RL agent does not require the presence of a supervisor to provide it with the right answers, but tries to optimize its performance by trial-and-error while exploring its environment. However, when applied to a real world problem, the autonomous exploration aspect of RL raises some issues, such as slow convergence rate and unsafe exploration [7].

In order to deal with these challenges, Interactive Learning has been proposed as a new paradigm for guiding the learning process of an agent by employing human teaching signals such as instructions [4], demonstrations [1], advice [3] or feedbacks [6, 15]. In [15], an RL-based software agent is guided by reward signals delivered by a human through a clicking interface. The values of these rewards are fixed in advance and added to the task rewards. [14] takes a similar approach by associating verbal feedbacks to predetermined reward values and by adding them to task rewards. Other approaches [5, 10, 12], however, consider only the human rewards and not task rewards. In [6], human generated rewards are used to learn a model of the the task reward function.

All these works have in common that they consider predetermined scalar values for human rewards, for example 1 and -1, and do not consider the question of learning these values. In contrast, in our work, only task rewards are predetermined while human rewards are learned. Furthermore, excepting [14] which considers verbally delivered feedbacks, all these works rely on an artificial reward interface such as a keyboard or mouse clicks and do not consider real teaching signals. In our work, we focus on the social interaction aspects by considering raw, nonverbal teaching signals. We find a similar approach in [4], where the system learns to solve the task by using unlabeled teaching signals. However,in [4], teaching signals are not considered as rewards but are rather used to infer the task reward function by Inverse Reinforcement Learning.

An important feature of these Interactive Learning systems is the autonomy of the learning agent with respect to the human. In [4], the learning agent is entirely dependent upon the human teaching signals so it is able to learn the task only when the human is present. It is also the case in [5, 10, 12], where the agent has no access to any rewards but those provided by the human. In contrast, in [15] and [14], human reward signals are added to task rewards, so the robot is able to learn autonomously beyond the human additional rewards.

3. METHOD

Our idea is to learn a model of the human rewards as a social reward function, that we distinguish from the task reward function. Formally, the distinction between the two functions lies only in the input space description. While the task reward function $R^T(s,a)$ is defined on task states $s \in S^T$, the social reward function $R^S(s,a)$ is defined on human teaching signals $s \in S^S$.

For this purpose, we propose a model we call Socially Guided XCS (SGXCS), based on three main components: a Task Model, a Social Model, and a Contingency Model. The Task and Social models are represented by two different Markov Decision Processes (MDP) and implemented by two different XCSs. The Task MDP serves to learn the task, while the Social MDP is used to learn the social reward function. The Contingency Model constitutes a mapping function between task states and social states (teaching signals), that allows to transfer the reward values from one model to another. The social reward function is learned by associating the rewards coming from the Task MDP to the corresponding teaching signals. It is then used for bootstrapping the learning process of the Task Model.

In the remainder of this section, we present our scenario, we give a detailed description of the three components of our model. Then, we present our algorithm.

3.1 Scenario

We consider a scenario (Figure 1) where a robot has to learn to press buttons. The experimental set-up is composed of a humanoid robot facing a table on top of which there is a set of buttons with different shapes and colors. At each moment, a screen displays the color or the shape of the button that the robot has to press. The robot is able to perform two kinds of action: gazing to one of the different buttons or pressing the one it is facing. So the task is a multi-step problem, meaning that in order to press the right button, the robot has to look for it first, and then to perform the action of pressing. While the robot is learning to perform the task, a human can sit in front of it in order to help it, as an adult would do with a child who is trying to solve a puzzle, by giving him some indications or feedbacks about his actions.

3.2 Model

In this section, we present the different components of SGXCS: the Task Model, the Social Model and the Contingency Model.

3.2.1 Task Model

The Task Model relies on two different principles, the use of the gaze as an intention device, and the use of affordances in action selection.

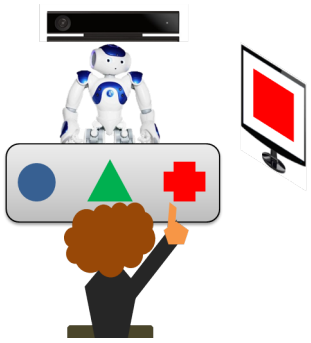


Figure 1: Scenario: The robot must press the button corresponding to the the information displayed on the screen. A human can help it by providing it with teaching signals.

Gazing as an intention device: Before taking an action, the robot looks at the object upon which it intends to act. We think that this could be a solution for the problem of random exploration by avoiding worthless and unsafe exploration.

Affordances in action selection: Instead of considering all the actions of the robot at the same level, we adopt a hierarchical model for actions by using the principle of affordances. The robot acts only on the objects it sees, and the set of possible actions is determined by this object. This could also contribute to limiting the drawbacks of exploration by reducing the space of possible actions in each state.

We represent the task as a Markov Decision Process $MDP^T =< S^T, A^T, T^T, R^T >$ where

- $S^T = C \times S \times G$ is the set of possible task states which are represented by three features C, S and G, where $C \in \{1...N\}$ and $S \in \{1...N\}$ are respectively the color and the shape of the button that the robot has to press and $G \in \{1...N\}$ is the button the robot is looking at.
- $A^T \in \{G_1, ..., G_N, P\}$ is the set of possible actions in each state where G_i is the action to gaze the button i and P is the action of pressing the button that the robot is gazing.
- $T^T : S^T \times A^T \rightarrow Pr[S^T]$ is the task transition function.
- $R^T : S^T \times A^T \rightarrow \Re$ is the task reward function.

In this preliminary work, we only consider a deterministic environment, meaning that T^T and R^T are deterministic. The action of gazing does not provide any reward. However, the robot receives a positive reward if it presses the right button and a negative reward if it presses a wrong button. The problem ends when the robot presses any button and receives a non null reward.

3.2.2 *Social Model*

In each state of the task problem, the human can provide to the robot an indication about the action to perform. We choose to use pointing and head movements as state features for the teaching signals. The human points to the button that the robot must press. If the robot is gazing to the right object, the human provides it with a head nod and if the robot is not gazing to the right object, the human provides it with a head shake. We model this problem as single-step, meaning that the problem ends after the robot performs any action, regardless of the value of the received reward.

The Social Model is a Markov Decision Process $MDP^S =< S^S, A^S, T^S, R^S >$ where

- $S^S = H \times P$ is the set of possible human states which are represented by two features H and P, where $H \in \{nod, shake\}$ is the information about the human head movements, and $P \in \{1...N\}$ is the information about the human pointing.
- $A^S \in \{G_1, ..., G_N, P\}$ is the set of possible actions in each state where G_i is the action to gaze at the button i and P is the action of pressing the button that the robot is gazing. Note that A^T and A^S are the same, meaning that we consider the same action set for both Task and Social models. So, in the remainder of this paper, we note it A.
- $T^S : S^S \times A^S \rightarrow Pr[S^S]$ is the social transition function. As the problem here is single-step, we do not consider transitions between human states.
- $R^S : S^S \times A^S \rightarrow \Re$ is the social reward function that we aim to learn.

3.2.3 *Contingency Model*

The contingency model represents the probability of observing a teaching signal in a certain task state. The model continuously receives states from both Task and Social models and updates their joint probabilities. When the robot receives a teaching signal s while being in a task state s', the Contingency Model updates the joint probability $P(s \cap s')$. So, even if the human is not present, the robot can still use the Social Model by getting the most likely teaching signal from the Contingency Model:

$$\hat{s} = \arg\max_{s \in S^S} P(s|s'); s' \in S^T$$

3.3 Algorithm

Algorithm 1 shows the different steps of our method. First, the robot evaluates the task state (line 1). If it receives teaching signals from the human, it uses them with the task state to update the Contingency Model (lines 2 to 4). Then, the social state is retrieved from the Contingency Model as the most likely teaching signal for the task state (line 5). Second (lines 7 to 11), if the social state is not empty, the Task Model is updated using the predictions of the Social Model, which are considered as social rewards. In this step, we rely on a minor modification of XCS, where the prediction array is used also to store the prediction of the most reliable classifier (with highest fitness). We need to do this because the prediction, computed as a weighted average of the predictions of all the classifiers belonging to an action set, can be distorted by newly created classifiers with low fitness but with high and wrong predictions; and so it can mislead the update of the Task Model. Then (lines 13 to 15), the robot performs an action, receives a reward from the task and uses it for updating the Task Model, like in the standard XCS algorithm. Finally, if the social state returned by the Contingency Model is not empty, the task reward is also used to update the Social Model (lines 17 and 18).

Algorithm 1 Socially Guided XCS

```
1: t_state ← task_environment.getState()
2: teaching_signal ← social_environment.getState()
3: if teaching_signal ≠ null then
4:     contingency_model.update(t_state,teaching_signal)
5: s_state ← contingency_model.getSocialState(t_state)
6:
7: if s_state ≠ null then
8:     social_pred ← social_xcs.getPredictionArray(s_state)
9:     for all actions a in A do
10:        social_reward ← social_pred[a].max_fit_prediction
11:        task_xcs.update(t_state,a,social_reward)
12:
13: action ← task_xcs.perform(state,exploration)
14: task_reward ← t_environment.perform(action)
15: task_xcs.evaluate(t_state,action,task_reward)
16:
17: if s_state ≠ null then
18:     social_xcs.update(s_state,action,task_reward)
```

4. EXPERIMENTS

In order to evaluate the contribution of the Social Model on task performance, we conducted experiments in simulation by using a modified version of XCSLib [8] that allows employing two different instances of XCS, one for the Task Model and another for the Social Model. For these experiments, the human behavior was implemented in a simple manner, through a function in the Environment class of the Social Model that takes as input a task state and returns the corresponding teaching signal deterministically. The task was instantiated with 5 buttons.

In this section, we describe the settings of the experiments and then, we discuss the results.

4.1 Settings

In order to investigate how the Social Model contributes to the task performance, we performed the experiments in two different settings. In the first setting, we compare SGXCS with the standard XCS. As a second setting, we modified the Social Model in SGXCS so that, instead of returning a teaching signal for a task state, it returns the task state itself. So, here, the social MDP and the task MDP become equivalent, aside the fact that in the first one, we consider a single-step problem, while in the second the problem is multi-step. This setting would be equivalent to a model-based XCS that learns the reward function on the task states and uses it to update its model as in Algorithm 1. We refer to this model as RBXCS.

For both settings, we run 10 experiments of 20000 problems, by alternating learning and test trials. We use the same XCS parameters for all models (see Table 1).

For performance evaluation, we use the following criteria: the evolution over the test trials of: the number of steps, the accumulated rewards and the population size at the end of the trial. For each criterion, we report the median, the minimal and the maximal values over the 10 experiments. The performance curves are smoothed by computing a moving average over a window of 500 trials.

Table 1: Parameters used for XCS, Task Model and Social Model within SGXCS and RBXCS

Parameter	Notation	Value
Population size	N	2000
Learning rate	β	0.2
Discount factor	γ	0.7
GA threshold	θ_{GA}	25
Crossover probability	χ	0.8
Mutation probability	μ	0.04
Accuracy criterion	ϵ_0	1
Accuracy falloff rate	α	0.1
Accuracy exponent	v	5
Prediction initial	p^0	10
Error initial	ϵ^0	0
Fitness initial	f^0	0.01
Deletion threshold	θ_{Del}	20
GA subsumption threshold	$\theta_{GA_{sub}}$	20
Hash probability	$P_\#$	0.3
Maximum steps	N_s	10

4.2 Results

Figures 3 and 4 report respectively the evolution of the number of steps and the evolution of accumulated rewards for XCS, RBXCS and SGXCS. We can observe that SGXCS outperforms both XCS and RBXCS in the number of steps and the accumulated rewards. Figure 5 reports the evolution of the population size for the three compared models. We can see that SGXCS and RBXCS equally outperform XCS, finding much more compact generalizations.

Figure 2 provides statistics about the performance of the three models over the 10 runs, in terms of number of trials until convergence. The convergence of the algorithm is considered when it reaches the optimal values in both the number of steps and the accumulated rewards criteria. A Mann-Whitney U test shows that the difference in performance between XCS and RBXCS is significant for a confidence level of 95%, while the difference between SGXCS and the two other models is significant for a confidence level of $99,9\%$.

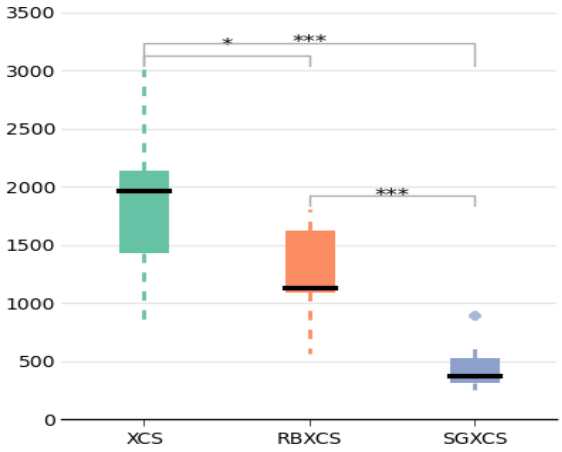
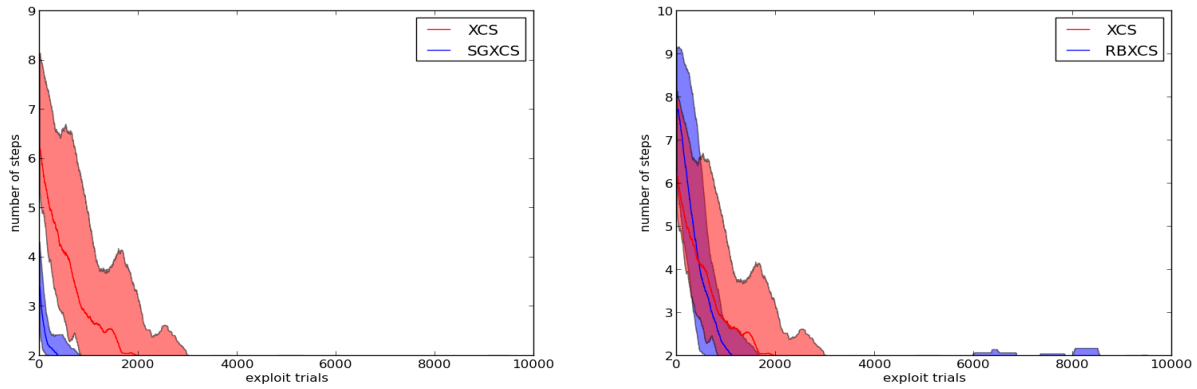
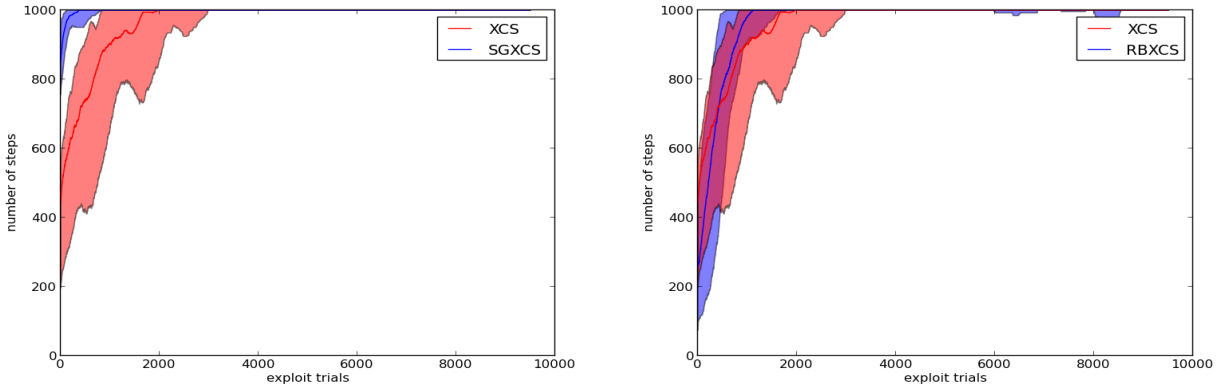


Figure 2: Number of trials to convergence for 10 runs. Mann-Whitney U test : $p < 0.05(*)$, $p < 0.001(***)$



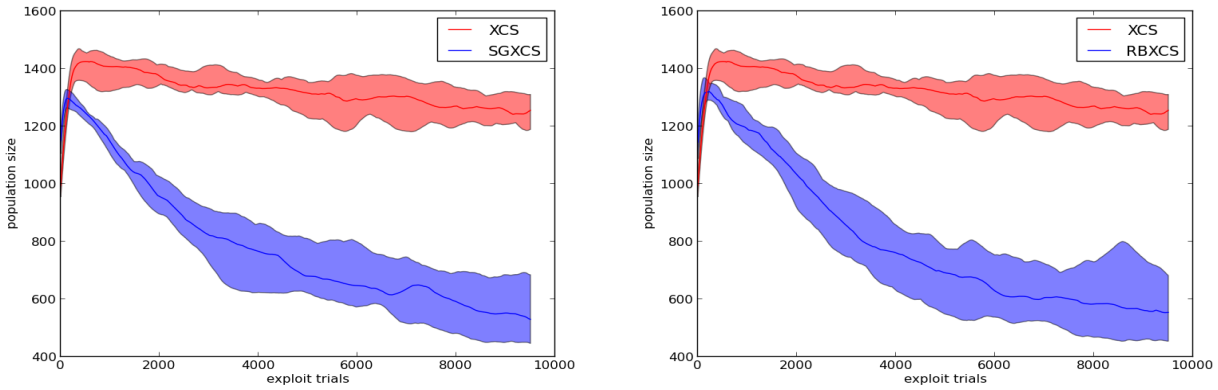
(a) (b)

Figure 3: Number of steps: (a) XCS vs. SGXCS. (b) XCS vs. RBXCS.



(a) (b)

Figure 4: Accumulated rewards: (a) XCS vs. SGXCS. (b) XCS vs. RBXCS.



(a) (b)

Figure 5: Population size: (a) XCS vs. SGXCS. (b) XCS vs. RBXCS.

Table 2 shows the learned rules in the Social Model. We can see that the model found correct generalizations on the teaching signals. Lines 1 to 5 predict with maximum accuracy a null reward on all the gazing actions, whatever the teaching signal. Lines 6 and 9 predict with maximum accuracy a reward of -1000 for pressing a button while the teaching signal contains a head shake (01 in the first two bits), whatever the pointing information (the last five bits). In the same way, lines 7 and 8 predict with maximum accuracy a reward of 1000 for pressing a button while the teaching signal contains a head nod (10 in the first two bits), whatever the pointing information.

Table 2: Learned classifiers in the Social Model: the first two bits represent head movement information. The remaining bits represent the pointing information.

Cond	Act	Pred	Err	Fit	Exp	Num
#######	1	0	0	0.99	10312	293
#######	2	0	0	0.98	10353	283
#######	4	0	0	0.99	9962	281
#######	5	0	0	0.99	10345	278
#######	3	0	0	0.99	10290	259
0######	0	-1000	0	0.77	6945	215
#0#####	0	1000	0	0.76	10736	215
1######	0	1000	0	0.23	10900	67
#1#####	0	-1000	0	0.21	6811	60

5. DISCUSSION

In this section, we provide some insights about the mechanisms of our model and their implications for task learning. First, we evaluate the cost of our model in terms of interaction load for the user. Then we discuss its complexity. Finally, we review the rules learned by the Social Model.

5.1 Interaction load

First of all, we consider the cost of our method in terms of human load measured as the number of interactions needed from the human in order to reach optimal performance. In the experiments that we performed in simulation, the Environment class that models the human behavior in the Social Model is designed to interact systematically with the algorithm by providing teaching signals in each task state. So it assumes that the human is able to interact with the robot thousands of times, which is not reasonable. However, the Contingency Model allows to relax this assumption by memorizing the corresponding teaching signal to each task state. Consequently, to achieve optimal performance, the human needs to interact with the robot only the number of possible task states, by giving teaching signals only in newly encountered situations. This may be done for example through a hard-coded behavior that asks for help whenever the Contingency Model provides an empty response for an unknown situation.

So, the system does not need to receive a teaching signal from the human in order to use it for updating the Task Model. The Contingency Model allows to update the Task Model with the rewards coming from the Social Model only through the teaching signals that have been previously provided by the human. So, even if the human is absent or not engaged with the robot, the Social Model still continues to boost the learning process of the Task Model (lines 7 to 11 of Algorithm 1). This remark is also valid in the opposite way, meaning that the Social Model can always be updated with an effective task reward, without the need for an effective teaching signal as it can get one from the Contingency Model (lines 5, 17 and 18 of Algorithm 1).

Furthermore, if the human does not interact with the robot, it will not worsen its performance with respect to the standard XCS algorithm. Indeed, if neither the human nor the Contingency Model provide a teaching signal in a given situation, the algorithm will behave normally as the standard XCS, by updating the Task Model only through effective task rewards (lines 13 to 15 of Algorithm 1). However, the more the human provides teaching signals, the more he improves the robot's performance. So the robot's learning process will be more or less improved depending on the level of the human engagement in teaching.

The genetic generalization mechanism provided by XCS allows to go one step further. In fact, a teaching signal can contribute to updating a task state even if they have never been contingent, and even to updating states that have never received a teaching signal. Consider two different task states s_1^T and s_2^T that have in common a set of matching classifiers M^T (cf. Figure 6). Assume that s_1^T has a corresponding teaching signal s_1^S in the Contingency Model and s_2^T has no corresponding teaching signal. During the update of s_1^T by s_1^S, the classifiers in M^T are updated by the social reward given by s_1^S. So, s_2^T is indirectly affected by s_1^S even if it has no associated teaching signal in the Contingency Model. This property is also reciprocal, meaning that the influence of a task state on a teaching signal through a task reward (line 18 of Algorithm 1) can be carried out indirectly through the generalization mechanism in the Social Model, even if the task state and the teaching signal have never been contingent.

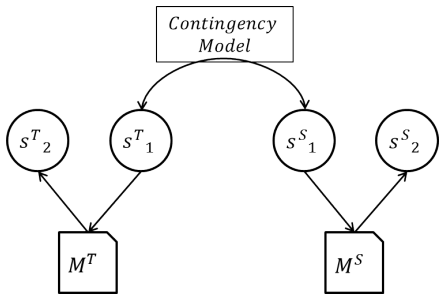


Figure 6: Extended influence: The classifiers M^T of the task state s_2^T are updated with the social reward coming from the teaching signal s_1^S, through the contingency of s_1^S and s_1^T. Similarly, the classifiers M^S of the teaching signal s_2^S are updated with the task reward coming from the task state s_1^S, through the contingency of s_1^S and s_1^T.

So, we can say that genetic generalization provides an extended influence to both Task and Social models, so that two states from both sides can influence each other even if they have never been encountered at the same moment. It also gives another dimension to the Contingency Model that acts beyond the state level, by reaching states at features level, even if it is not explicitly designed in this way. Consequently, the minimal number of interactions needed by the system to reach optimal performance could be less than the number

of task states, depending on the extent to which the Task Model and the Social Model are generalizable.

5.2 Complexity analysis

The extended effect of rewards allowed by genetic generalization provides a better understanding of the contribution of the teaching signals in SGXCS with respect to task states in RBXCS. One could think that the same thing happens in RBXCS as in SGXCS, meaning that there is an extended propagation of the rewards in both directions because of generalization. So, let us consider the same example for RBXCS as previously, with two states s_1^T and s_2^T from the Task Model that share a common matching classifiers M^T and a teaching signal s_1^S from the Social Model that is associated with s_1^T through the Contingency Model. The state s_2^T is indirectly influenced by the social rewards of s_1^S through M^T, as in the previous example. But knowing that in RBXCS, $s_1^S = s_1^T$, the influence of s_1^T on s_2^T is already carried out by genetic generalization within the Task Model. So, with RBXCS, there is no additional information that could be provided by the extended influence of rewards, apart from the fact that, compared to XCS, the Task Model is updated several times in one step using the previously encountered rewards, whereas in the standard XCS the update is performed only once in each step for the current situation and the effective reward. So, we can say that teaching signals provide more information than those provided by the extended influence of rewards.

Actually, beyond the effect of genetic generalization, the Social Model in SGXCS provides a generalization over the state-space of the Task Model. There are two main factors to that: the state-space complexity of each model and the mapping function between task states and teaching signals.

On the one hand, state-space complexity in the Task Model is in $o(n^3)$, where n is the number of buttons. However, the state-space complexity of the Social Model in SGXCS is in $o(n)$. So, the state-space of the Social Model is smaller than the state-space of the Task Model, which means that learning in the former is faster than in the latter.

On the other hand, the mapping function between task states and teaching signals contributes further to the reduction of the complexity of the Task Model. In fact, one single teaching signal can correspond to many different task states. This is due to the fact that teaching signals represent a global evaluation of the robot's state and provide an indication on the action to perform, beyond the details of the robot's state. For example, when the robot gazes at the wrong button, the human will always provide it with the same information not to press the button (through a head shake) and the same information about the button to press (through pointing), whatever the button that the robot is gazing at.

So, we can say that the state-space of the Social Model in SGXCS constitutes a certain generalization of the state-space of the Task Model, that contributes, in addition to the mechanisms that we mentioned above, to the acceleration of the learning process.

5.3 Social rules

In this paragraph, we review the rules learned by the Social Model and their contribution to the learning process.

In Table 2, we can see that the pointing features do not provide any useful information for the robot's decision making. Indeed, the "social rules" tell that, whatever the pointing, the robot has to press the button if there is a head nod and gaze to any button if there is a head shake, without specifying to which one. This is due to the fact that the Social Model learns to predict the direct rewards of the task, and so, it has a myopic vision of action outcomes. In other words, the Social Model does not learn, when the robot is gazing to the wrong button, that the pointing information could get it in a situation in which it can get a positive reward. It only tells not to press the button it is looking at, but not which button it has to look at. This is because in this model, head movements inform about the action of pressing or not, and the pointing informs about the action of gazing. As the action of pressing yields a direct reward and the action of gazing yields only long-term outcomes, the Social Model as it is designed (to predict direct rewards) has a myopic vision that does not exploit the long term information of pointing. One solution to that would be to use the Social Model to learn the state-action values instead of the reward function, so the information driven by the pointing would be more informative.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a method using the XCS framework, for guiding an RL agent with unknown teaching signals. Our model, the Socially Guided XCS (SGXCS), is based on three main components: a Task Model that is responsible for learning the task, a Social Model that learns a social reward function and uses it to bootstrap the learning process of the Task Model, and a Contingency Model that allows to associate task states with their corresponding teaching signals. We showed that our model improves the task learning speed with respect to the standard XCS algorithm.

In the experiments we performed in simulation, we considered only a deterministic environment. So, in future work, we propose to extend our model to the stochastic case. We also intend to test our model on other domains, specially domains with a sparse reward function.

Using unknown teaching signals and learning their values online provides to our model a certain flexibility for the human in choosing its own teaching strategy. For example, inverting head nods and head shakes would not affect the model performance. However, we have to study to what extent our model resists to inconsistent teaching signals. This question will also be addressed in future work.

As discussed in Section 5.1, the performance of our system could vary depending on the number of interactions with the human. So, we propose to study the evolution of the performance according to the level of engagement of the human in the teaching task. Furthermore, if we take into account the extended effect of rewards allowed by genetic generalization, another non-trivial question would be to find the minimal set of interactions required by the system to reach optimal performance.

Finally, in order to benefit from the long-term information provided by the teaching signals, we propose to investigate the possibility to extend our model to learn the state-action values instead of the reward function.

7. ACKNOWLEDGMENTS

This work is funded by the Romeo2 project.

8. REFERENCES

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

[2] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida. Cognitive developmental robotics: a survey. *Autonomous Mental Development, IEEE Transactions on*, 1(1):12–34, 2009.

[3] 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*, pages 2625–2633, 2013.

[4] J. Grizou, I. Iturrate, L. Montesano, P.-Y. Oudeyer, and M. Lopes. Interactive learning from unlabeled instructions. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, 2014.

[5] C. Isbell, C. R. Shelton, M. Kearns, S. Singh, and P. Stone. A social reinforcement learning agent. In *Proceedings of the fifth international conference on Autonomous agents*, pages 377–384. ACM, 2001.

[6] W. B. Knox and P. Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, May 2010.

[7] J. Kober, J. A. D. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, July 2013.

[8] P. L. Lanzi and D. Loiacono. Xcslib: The xcs classifier system library, 2009.

[9] A. N. Meltzoff, P. K. Kuhl, J. Movellan, and T. J. Sejnowski. Foundations for a new science of learning. *science*, 325(5938):284–288, 2009.

[10] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pages 1–7. IEEE, 2011.

[11] O. Sigaud and S. W. Wilson. Learning classifier systems: a survey. *Soft Computing*, 11(11):1065–1078, 2007.

[12] H. B. Suay and S. Chernova. Effect of human guidance and state space size on interactive reinforcement learning. In *RO-MAN, 2011 IEEE*, pages 1–6. IEEE, 2011.

[13] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[14] A. C. Tenorio-Gonzalez, E. F. Morales, and L. Villaseñor-Pineda. Dynamic reward shaping: training a robot by voice. In *Advances in Artificial Intelligence–IBERAMIA 2010*, pages 483–492. Springer, 2010.

[15] A. L. Thomaz and C. Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI*, volume 6, pages 1000–1005, 2006.