Toward Supervised Reinforcement Learning with Partial States for Social HRI

Emmanuel Senft
CRNS
Plymouth University
United Kingdom

Séverin Lemaignan
CRNS
Plymouth University
United Kingdom

Paul Baxter
L-CAS
University of Lincoln
United Kingdom

Tony Belpaeme
CRNS
Plymouth University (UK)
ID Lab – imec
Ghent University (BE)

Abstract

Social interacting is a complex task for which machine learning holds particular promise. However, as no sufficiently accurate simulator of human interactions exists today, the learning of social interaction strategies has to happen online in the real world. Actions executed by the robot impact on humans, and as such have to be carefully selected, making it impossible to rely on random exploration. Additionally, no clear reward function exists for social interactions. This implies that traditional approaches used for Reinforcement Learning cannot be directly applied for learning how to interact with the social world. As such we argue that robots will profit from human expertise and guidance to learn social interactions. However, as the quantity of input a human can provide is limited, new methods have to be designed to use human input more efficiently. In this paper we describe a setup in which we combine a framework called Supervised Progressively Autonomous Robot Competencies (SPARC), which allows safer online learning with Reinforcement Learning, with the use of partial states rather than full states to accelerate generalisation and obtain a usable action policy more quickly.

Introduction

Human-Robot Interaction (HRI) studies how people and robot can co-exist in society, and how they can interact socially in different environments and contexts. Robot are expected to behave appropriately regardless of the domain of interaction. However, it is impossible to foresee all possible interaction outcomes in dynamic and open social domains, as such the robot's responses cannot be implemented in the robot before its deployment in the real world. Similarly to people, robots need to be able to learn how to complete tasks through creating and optimising action policies. This includes learning social norms and how to make sense of the social world.

For a robot, interacting in the real world often requires taking in a diverse and large range of sensory inputs, which results in a high-dimensional sensory space. The robot then is required to select actions based on its current state in sensor space. However, parts of the state can be irrelevant to the current goal, and as such should not be taken into account when selecting the current action. Often, only a limited number of features in the space are important. To interact efficiently, a robot has to learn to identify these salient features and associate them with appropriate actions.

In previous work (Senft et al. 2015; 2017), we introduced the Supervised Progressively Autonomous Robot Competencies (SPARC) as a way to teach a robot an action policy while interacting based on a human supervisor intervening and correcting actions before they were executed by the robot. In this paper, we propose to extend this approach to allow the supervisor to highlight features in the environment relevant for the selected action. During the action selection phase, the robot can compare these features, defined as *partial states*, with the current state to select an action. The selected action is presented to the human supervisor, who can either correct the proposed action or approve it for execution.

Background

Reinforcement Learning

The main framework for an agent to learn how to interact in an environment while interacting is Reinforcement Learning (RL) (Kober, Bagnell, and Peters 2013; Sutton and Barto 1998). In RL, an agent interacting in an environment receives numerical rewards in reaction to its actions. The agent subsequently learns an action policy to maximise the expected cummulative discounted reward.

In many cases where RL achieves success, the agent has access to a virtual environment where the only real cost of exploring is computational effort: the agent can interact as long as needed to gather enough information on the environment and the result of its actions on the environment in order to find a sufficiently optimal action policy. Using simulation RL can achieve impressive results, as shown in RL learning to play Backgammon in the 90s (Tesauro 1995) to the more recent success in mastering the game of Go (Silver et al. 2016). Even when a virtual environment is available, but especially when it is not, human knowledge and effort is required during the design of the algorithm or the learning phase before the learning can be successful.

Human impact on Reinforcement Learning

Initial knowledge is often crucial to allow an algorithm to learn an efficient action policy. This knowledge, originating from human expertise, can be exploited in many ways.

Design decisions: Initial knowledge has to be used in the design of the algorithm, the representation of the state and actions spaces and the reward funtion. For example, only carefully crafted features allowed Tesauro to improve its algorithm for Backgammon from a intermediate-level player to super-human level (Tesauro 1995). Similarly, the design of complex neural networks for state generalisation, the representation of actions as motor primitives rather than raw motor angles or adding additional information in the reward function have important impacts on the ability of the robot to reach a successful policy (Kober, Bagnell, and Peters 2013).

Demonstrations: Initial knowledge can be provided to the agent through demonstrations. These demonstrations can be used to create an initial policy which is sufficiently efficient to start interacting in the environment and gather information to improve over time. This initial policy, required to receive meaningful feedback from the environment can be impossible to reach by relying only on random exploration and feedback from the environment. For example the game of Go in its larger board size contains more than 10^{210} states, so exhaustive search is not possible. Silver et al. (Silver et al. 2016) started with supervised learning from Go masters' games to learn a decent enough policy and then proceeded using deep learning, self play and tree search to achieve super-human capabilities and the capacity to beat the best human players.

These demonstrations can also be used to learn a reward function. With Inverse Reinforcement Learning, the agent is not provided with a reward function, but derives it instead from a set of expert demonstrations. The agent can then explore around the demonstrated policy to optimise the reward function. With this approach, Abbeel and Ng achieved better than human control for a robotic helicopter (Abbeel and Ng 2004) based on demonstrations from experts and further exploration and autonomous learning.

Guiding the learning: Rather than solely providing initial knowledge to the agents, humans can also guide the agent during its learning, a method more resembling human teaching.

A first approach consists in sequencing the tasks the robot will face. This approach, known as "scaffolding" (Saunders, Nehaniv, and Dautenhahn 2006), progressively increases the difficulty and complexity of the task as the robot is learning to reach policies which would take prohibitively long without scaffolding.

Agents learning in environments providing rewards can also benefit from additional rewards from human teachers. Depending on the task and the environment, different ways exist to combine rewards from multiple sources, and studies show that augmenting rewards from the environment by human ones can speed up the learning and reduce the number of undesired behaviours (Griffith et al. 2013; Knox and Stone 2010; Judah et al. 2010).

When environments do not supply rewards, they can be replaced by human ones. For example, the TAMER framework (Knox and Stone 2009) derives a reward function from the human feedback and uses this function to evaluate the current behaviour. In a similar approach, MacGlashan et al. proposed COACH in (MacGlashan et al. 2017). COACH assumes that human rewards represent the advantage function, i.e. how much the current action is better than the current policy, allowing to adapt the reward function to the strategies used and the current state of the learner.

Another approach to guide the learning is to bias the action selection. In (Thomaz and Breazeal 2008), the authors propose using a human supervisor to supply an agent with both rewards and potential guidance indicating what the agent should pay attention to, or what action it should take next. They show that giving the power to the supervisor to bias the action selection can improve the learning, making it faster and safer.

In (Senft et al. 2015), we introduced the Supervised Progressively Autonomous Robot Competencies (SPARC). SPARC relies on a supervisor with the ability to control the robot's actions. This supervisor is presented with the action the robot is about to execute. He can then choose to cancel it, allow it to be executed, or can select an alternative action. Based on the supervisor's decisions, the robot learns which actions are desired and can as such refine its policy over time, thus progressively reducing the need for the supervisor to correct or select an action.

SPARC gives control of the robot's actions to an expert, who can guide the exploration in the desired part of the environment which ensures the robot's behaviour is consistently appropriate. As the exploration is guided, and all the actions are useful, the learning can be faster than autonomous learning or learning based on human feedback as illustrated in Figure 1. In Autonomous Learning, the agent has first to discover its environment to start gathering relevant feedback, leading to a low initial performance and a slow improvement in early stage of learning. With teaching based on human feedback, the teacher can quickly provide information on actions to reach an efficient policy more quickly. However, as the teacher only provides feedback and cannot prevent the agent making mistakes, the initial performance can be poor. On the other hand, SPARC, with the control of the teacher over the executed actions, can prevent the agent from making mistakes in early stages, achieving a high performance even at the start of the learning.

Compared to similar algorithms (Chernova and Veloso 2009; Walsh et al. 2010), which allow the agent to request demonstrations and/or the teacher to provide demonstrations to correct mistakes made by the robot, SPARC allows the teacher to correct any action before its execution, thus reducing importantly the risk of the agent making errors. The total blending between autonomous execution of actions and demonstrations and the control over *every* actions executed by the agent are the specificities of SPARC.

In (Senft et al. 2017), we presented a way to combine SPARC and RL, by assigning a positive reward to every action executed by the robot, making the assumption that every action has been explicitly or implicitely approved by the supervisor. However this method directly mapped a single (state, action) pair to a reward without making use of any kind of generalisation. As such it was not applicable to

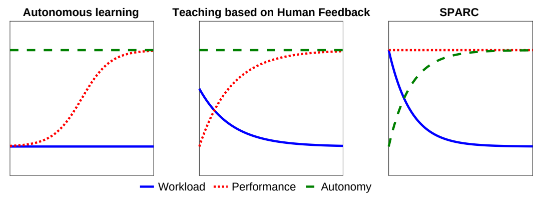

Figure 1: An illustration of the evolution over time of the performance, autonomy and human workload for an autonomous learner, an approach using human feedback, and SPARC.

environments with a continuous or high dimensional space or non-deterministic transitions from one state to another, elements which are typically present in social interactions. Similarly to TAMER or COACH, this approach allows to reproduce an action policy from a teacher even in the absence of a reward function, but as the teacher has control over the robot actions, the learning can be faster and safer.

Partial State Supervised Reinforcement Learning

To make RL applicable in high dimensional or continuous states, the algorithm requires a way to generalise knowledge to unseen states. A classic approach is to use a feed-forward neural network or deep learning to learn a the value function generalising to unseen states. Neural networks rely on having a large number of datapoints to converge toward a good function approximator. Alternatively, Cobo et al. proposed in (Cobo et al. 2011) to abstract features automatically from demonstrations to learn faster. But even in this case, the number of datapoints required is still high (around 1000 samples per participant). However, in many applications and especially in HRI, these amounts of data are not available or obtainable due to practical constraints. Furthermore, human responses are often noisy and lack consistency, so methods are needed which can generalise from a low number of noisy data points.

In this paper we propose to use a human user to highlight the relevant features of the environment to reduce the state dimension of the points only to relevant information. We introduce the concept of a *partial state*, a sliced version of the state defined only on a subset of the dimensions of the state. This shifts the (state, action) pair paradigm to (partial state, action) and in the case of RL, the tuple (state, action, reward) to (partial state, action, reward). This allows a comparison of the current state and the datapoints only on relevant features for action selection. With this instance-based method (Aha, Kibler, and Albert 1991), the algorithm can have a state abstraction allowing it to generalise even with a few datapoints instead of the large numbers normally required to abstract features from examples.

Learning algorithm

An expert supervises the agent actions, and can assign rewards to actions and highlight the parts of the states which are relevant to assigning this reward to this partial state.

As the supervisor can estimate the future impacts of an action, the problem of credit assignment for delayed rewards can be ignored which allows us to consider only a myopic approach in a fashion similar to TAMER (Knox and Stone 2009).

For this paper, we will reuse the formalism of rewardless Markov Decision Process to identify the different elements of our system. The agent has access to a set of actions $\mathcal{A}$ and a state $\mathcal{S} \in [0,1]^n$. We also define the partial states $\mathcal{S}' \in [0,1]^{n'}$ with $n' \leq n$ as a slice of $\mathcal{S}$, a subset of $\mathcal{S}$ where some dimensions have been removed.

When the agent executes an action a in state s, it receives the reward r associated with the partial state s'. For example, a state s could be defined in 4 dimensions such as $s = [1, 0.2, 0, 0.5]$, and s' in two dimensions with $s' = [-, 0.2, 0, -]$ with symbol '$-$' reprensenting the dimensions removed. For the learning algorithm, this means that the action a has been evaluated r in the partial state s'.

To each action $a \in \mathcal{A}$ we can associate a set $\mathcal{C}_a$ of pairs (s', r) representing the rating done by the supervisor to action a with features highlighted for the multiple s'. When adding a new pair (s', r), we can discard potential previous pairs with an identical s' to represent the evolution of the policy evaluation by the supervisor.

Algorithm 1: Algorithm for selecting an action based on the previous (partial state, action, reward) tuples and the current state.

```
inputs : Current state $s$, set of $(a, s', r)$
output: selected action $\pi(s)$
foreach $a \in \mathcal{A}$ do
    foreach $p = (s', r) \in \mathcal{C}_a$ do
        compute similarity $\Delta$ between $s$ and $s'$:
        $\Delta(p) = 1 - \frac{\sum_i^{n'} (s'(i) - s(i))^2}{n'}$
    find closest pair $\hat{p}$:
    $\hat{p} = arg\,max_p \Delta(p)$
    compute expected reward $\hat{r}(a)$ for taking $a$ in $s$:
    $\hat{r}(a) = \Delta(\hat{p}) \cdot r(\hat{p})$
    with $r(p)$ the reward $r$ of the pair $p = (s', r)$
Select the action with the maximum expected reward:
  $\pi(s) = arg\,max_a \hat{r}(a)$
```

When facing a new state s where an action has to be selected, the agent can select an action following Algorithm 1 in a instance-based learning fashion. For each action $a \in \mathcal{A}$, we take the pair (s', r) with the closest s' to the current state (as defined by the average quadratic distance over the normalised dimensions where s' is defined). That way, each action a can be associated to an expected reward defined by the product between the similarity of the closest partial state known for a and the reward obtained for executing a in that partial state. Finally, the action with the highest expected reward can be selected.

The normalisation of each dimension of the state allows distances to be comparable as values on all dimensions have the same range. Additionally taking the average quadratic distance over each defined dimension allows to compare

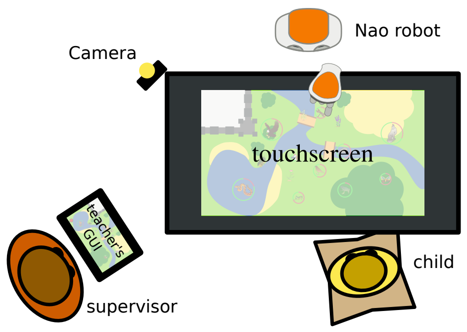

Figure 2: Interaction setup: the child and the robot are interacting on the touchscreen and a supervisor can control the robot using a GUI on a tablet.

similarities even when states are defined on a different number of dimensions.

Combination with SPARC

SPARC has been shown to be compatible with RL in (Senft et al. 2017), and can also be easily combined with the approach presented in this paper using partial states. For example, when selecting an action for the robot to be executed, the supervisor can also select which features in the environment should be selected as the partial state, and this associates the reward to this action in this partial state. Similarly, when an action is proposed to the supervisor the features represented by the dimensions of the closest partial state for this action can be exposed to the supervisor as a way to explain why this action has been selected. Facing this, the supervisor can: (1) not react, allowing the action to be executed and associating a reward of +1 to the partial state and the action proposed, (2) change the partial state to correct the features related to this action or (3) cancel it, preventing the execution and associating a reward of -1 to the partial state identified by the robot or the supervisor.

Application scenario

An example of an application is a social robot which interacts with children in an educational scenario. The robot plays an educational game with children to teach them notions about diverse topics according to the needs of the teacher. For this example, a child and a robot are playing a game about the food web, teaching which animals eat which ones on a Sandtray (Baxter, Wood, and Belpaeme 2012). In addition to the child and the robot, an adult supervises the robot using a tablet with a Graphical User Interface (GUI) to teach the robot how to interact with the child as shown in Figure 2.

The GUI (Figure 3) is an augmented version of the game itself, which can be use to make the robot move items on the game by dragging them on the GUI or which can display actions proposed by the robot with a cancel button to refuse an action. For example in Figure 3 the robot proposes to move the eagle to the rat, and highlights (as shown by blue circles) the eagle and the rat. This indicate that features relevant to the eagle and the rat have been used to select this action.

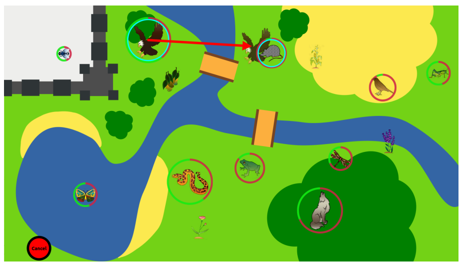

Figure 3: Interface for the supervisor with an action being proposed, moving the eagle to the rat highlighting both the eagle and the rat.

In the current implementation, the state is defined by the distance between each animal and their energy. With these features selected, the partial state transmitted to the user is the distance between the eagle and the rat, the eagle's energy and the rat's energy. Similarly, when selecting an action, the supervisor can select features in the state relevant to the action.

The main limitations of the approach reside in the difference of representation of the state and action spaces between the supervisor and the algorithm and the limit in communication. For example a user could try to move an animal close to another one, and depending on the representation of the actions on the algorithm side, the action might not be understood in the same way. Similarly, features used by the supervisor to select actions might not be represented in the state used by the algorithm. And lastly, in the case of implicit selection of features, a single case of features representation (for example highlighting two animals) might not be perceived in the same way by observers.

Future work

The system presented in the previous section will be improved and evaluated in the real world with children in the next months.

The current work could also be extended to allow the agent to continue to improve its behaviour even in the absence of a supervisor, progressively exploring around the learnt policy improving its behaviour beyond the demonstrations. This could be done by allowing the supervisor to provide rewards during the learning phase and combine these rewards with the demonstrations to learn a reward function in a fashion similar to Inverse Reinforcement Learning (Abbeel and Ng 2004) or TAMER (Knox and Stone 2009). This could also use partial states associated to these rewards to ease the generalisation of the reward function with only a low number of datapoints. However, without the supervisor, the assumption that only a myopic action selection is sufficient would not hold anymore and the problem of delayed rewards would have to be tackled.

Acknowledgments

This work was supported by the EU FP7 DREAM project (grant no. 611391) and EU H2020 Marie Sklodowska-Curie Actions project DoRoThy (grant 657227).

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*.

Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. Instance-based learning algorithms. *Machine learning* 6(1):37–66.

Baxter, P.; Wood, R.; and Belpaeme, T. 2012. A touchscreen-based sandtray to facilitate, mediate and contextualise human-robot social interaction. In *Human-Robot Interaction (HRI), 7th ACM/IEEE International Conference on*, 105–106.

Chernova, S., and Veloso, M. 2009. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research* 34(1).

Cobo, L. C.; Zang, P.; Isbell Jr, C. L.; and Thomaz, A. L. 2011. Automatic state abstraction from demonstration. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22.

Griffith, S.; Subramanian, K.; Scholz, J.; Isbell, C.; and Thomaz, A. L. 2013. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*, 2625–2633.

Judah, K.; Roy, S.; Fern, A.; and Dietterich, T. G. 2010. Reinforcement learning via practice and critique advice. In *AAAI*.

Knox, W. B., and Stone, P. 2009. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, 9–16.

Knox, W. B., and Stone, P. 2010. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 5–12.

Kober, J.; Bagnell, J. A.; and Peters, J. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32(11):1238–1274.

MacGlashan, J.; Ho, M. K.; Loftin, R.; Peng, B.; Wang, G.; Roberts, D. L.; Taylor, M. E.; and Littman, M. L. 2017. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning*.

Saunders, J.; Nehaniv, C. L.; and Dautenhahn, K. 2006. Teaching robots by moulding behavior and scaffolding the environment. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 118–125.

Senft, E.; Baxter, P.; Kennedy, J.; and Belpaeme, T. 2015. Sparc: Supervised progressively autonomous robot competencies. In *International Conference on Social Robotics*, 603–612.

Senft, E.; Baxter, P.; Kennedy, J.; Lemaignan, S.; and Belpaeme, T. 2017. Supervised autonomy for online learning in human-robot interaction. *Pattern Recognition Letters*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

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

Tesauro, G. 1995. Temporal difference learning and td-gammon. *Communications of the ACM* 38(3):58–68.

Thomaz, A. L., and Breazeal, C. 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence* 172(6):716–737.

Walsh, T. J.; Subramanian, K.; Littman, M. L.; and Diuk, C. 2010. Generalizing apprenticeship learning across hypothesis classes. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 1119–1126.