Social Robotics Mini Project: Learning from Human Feedback

BEKIROGLU Begum
3702569
ZAKARIA Belgroune
3704013

Abstract

In this project we studied existing methods of interactive machine learning. This is was to ensure a better learning result by integrating human guidance to the learning process. This guidance can take place in three main ways : Descriptive (where the teacher provides a sequence of actions for multiple tuples of states), evaluative (where the teacher gives intermediate evaluations for the actions) and imperative (where the teacher tells the agent to do an action for a specific state). There are also different ways for the human to communicate with the agent such as keyboard, mouse, with a text etc. (1). Our aim with this project is to improve an existing algorithm in order to provide a better human-machine interaction and to obtain a better result.

Related work

We studied several approaches of human intervention on machine learning. Each approach combines different feedback types with different communication modalities and implement their algorithm in different environments. Studying these algorithms gave us the opportunity to see different possibilities of integrating human guidance into machine learning. (2) (3) (4) (5) (6)
.

Algorithm	Modality	Feedback Type	Environnement	Result
The Rational Speech Acts Framework (2)	Language understanding	Descriptive/ Evaluative	The Forest of Fungi	Instructive teaching gives better results when it is used for low-autonomy settings however descriptions work better for when the agents has to act independently. The best result (0.94 future rewards and 9.55 regret) is obtained for their Latent H listener.
Bayesian Goal Interference (3)	Language understanding	Descriptive	Fetch block-stacking	They used two kinds of learners and teachers in this research: Naive/Pedagogical Teacher and Literal/Pragmatic Learner. The best result obtained was with the combination of pedagogical + pragmatic. With this combination the algorithm reached its goal in less than 20000 instructions.
TAMER (4)	Button press or language understanding	Evaluative	Tetris/ Mountain car	The performance reached an approximate peak of 65.89 lines cleared per game after only 3 games for Tetris. This shows that the TAMER agent learns better and faster than other incrementally updating algorithms tested in the article. When it comes to Mountain car environment, TAMER outperformed Sarsa-3 et Sarsa-20 agents when it was trained by a good teacher .
Hybrid A3C/IL Hybrid SARSA/IL (5)	Online feedback, but the feedback type is not specified.	Evaluative	Cart-pole/ Mountain car	For both methods, results show that hybrid methods are more efficient than the stand alone reinforcement learning in both environnements. In cart-pole environnement Hybrid A3C/IL has 14.2% more data efficacy than SARSA/IL and it has approximately # 200 cumulative reward. For the mountain car, on the other hand, both of the hybrid methods reach # -100 cumulative reward.
Human-Robot Interaction (6)	Mouse	Evaluative	Sophie's Kitchen	Evaluations show that this method improves the speed of task learning and the efficiency of state exploration. They also observed a significant drop in the number of failed trials encountered during learning.

FIGURE 1 – State of art

The Choice of the Project

Once we studied all of the articles mentioned above, we choose the TAMER algorithm to replicate and to modify certain aspects with the aim of obtaining a better result. The reason of this choice is the abundance of articles and codes using this algorithm and the facility of the implementation. The TAMER project that we consider works with an evaluative feedback given by the human user by using the keyboard. We used numpy, sklearn, pygame and gym libraries and platforms to recreate this project.

The TAMER algorithm is originally used in Tetris and mountain car environments. We have worked on Mountain car environment to replicate the article. This is an environment where a car is placed at the bottom of a sinusoidal valley. The aim is to find a way to accelerate the car in order to reach to the flag that is placed on top of the right hill. The problem of classical RL methods with this environment is that since the goal is not obvious to reach and each action that doesn't make the car reach the goal brings a negative feedback, the car can never learn how to reach the goal. So we need to either give the car instructions about what to do for some strategic states or to give some intermediate rewards to guide it. TAMER algorithm uses this second method.

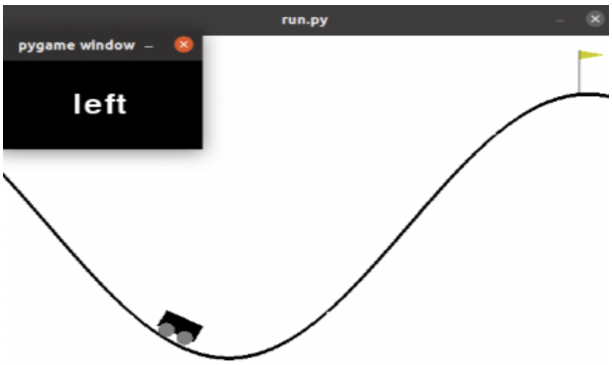

FIGURE 2 – TAMER algorithm in Mountain Climber environment

We recreated the TAMER algorithm by using the examples already existing in GitHub (7). We analyzed the code to understand how does the human feedback intervenes. The principle part concerning our project is placed the agent.py file. The training takes place in this file. In each time step, the algorithm takes an action based on epsilon greedy policy. If at the same time there is a human intervention that the code is able do detect thank to the interface.py file using the pygame library, the program is updated with the reward for the state and the action.

When we run the algorithm using Q-learning method, the agent needs at least 50 epochs to learn in average. The reward graph that we obtain shows that the agent wasn't capable of finding the flag multiple times (when the reward is equal to -200). For the best epoch, the algorithm was able to obtain an approximate episode reward of -130.

When we give intermediate rewards for actions using TAMER algorithm, the learning takes place in only 3 epochs of learning. The reward graph that we obtain with a non-optimized teacher shows the performance of this algorithm compared to Q-learning method. We can easily observe that the agent was able to find the flag for each epoch and the best episode reward is -107 with a mean reward of -128.

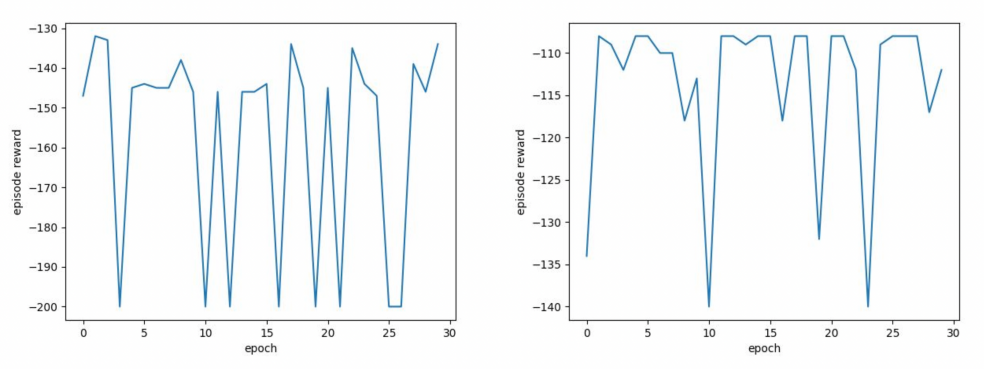

FIGURE 3 – Mountain Climber with Q-learning on the left and Mountain Climber with TAMER algorithm on the right

Alternative Approaches : Changing Modality

Even though the existing TAMER algorithm is highly efficient compared to Q-learning, we found it counter-intuitive for the teacher to use keyboard to give positive and negative rewards. Thereby, our first approach to improve the algorithm was by changing the feedback modality. To do so we have first thought of including voice commands in order have a more natural feedback form humans to interact with the agent. We used speech recognition package to realize this transformation between voice and rewards. However this package needs to have access to some online servers and not having a stable internet connection made this method unworkable for our project.

We have then decided to use a visual recognition algorithm to define the rewards. We used DeepFace library for this task. DeepFace is an attribute analyzing framework which is capable of differentiating 7 emotions : Angry,Disgust,Fear,Happy,Sad,Surprise and Neutral (8). We, on the other hand, need only 3 states for our project : positive feedback, negative feedback, no feedback. To solve this problem we have decided to regroup Sad,

Angry, Disgust, Fear and Surprise as negative feedback and happy as positive feedback.

Once we replaced the keyboard feedback with emotion detection, using the following GitHub to change the modality type (9), we have noticed that there is a time delay while using emotion detection related to OpenCV and DeepFace libraries. To give the trainer an adaptation time to this delay, we have extended the learning duration to 5 epoch instead of 3. Once the trainer gets used to the algorithm, we have obtained slightly better results compared to classical TAMER algorithm. For a good teacher we have obtained a mean reward of -126.

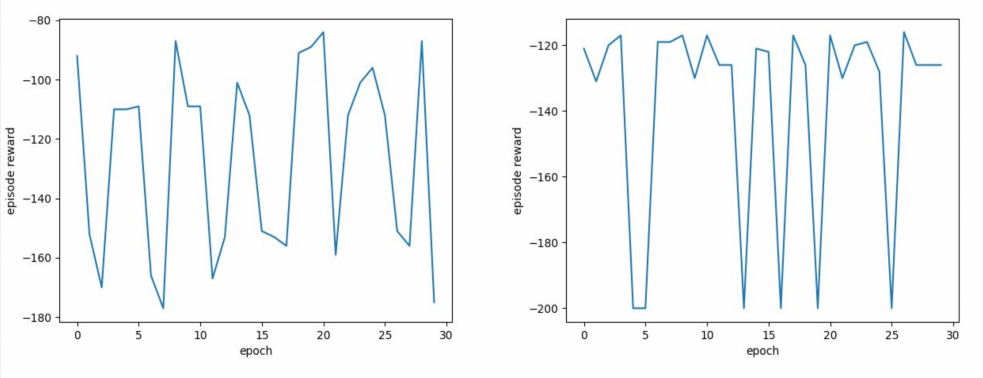

FIGURE 4 – Mountain Climber with TAMER with emotional feedback. On the left there is the result for a good trainer and on the right we see the result for an inexperienced trainer.

Alternative Approaches : Changing Feedback Type

We have also changed the feedback type from evaluative to imperative. To do so we have used our keyboard to tell the agent the action that it should take. We have then replaced the action that we have for Q-learning with our feedback. For the Mountain car environment changing feedback type did not give us an exploitable result. The algorithm preferred doing nothing most of the time. Without further digging, we can guess that the non-binary action map of mountain car prevents the imperative feedback to overcome the natural epsilon greedy choice of action.

Different Environment

We have also implemented our algorithm in Cart Pole environment. This is an environment in which we have an upright placed pendulum and the goal is to keep it in balance by applying forces towards left of right (10). The main challenge with using our algorithm with this environment is that an

epoch only takes a few seconds. Therefore, the trainer has to react rapidly and the agent has to get the reward without delay. Even though with classical TAMER algorithm we were capable of obtaining good results (mean reward of 42 in 5 epoch), our emotion detection method has not been efficient for this environment because of the delay of the feedback. Imperative method, on the other hand, for this environment we have managed to obtain the best result(mean reward of 44 in 5 epoch).

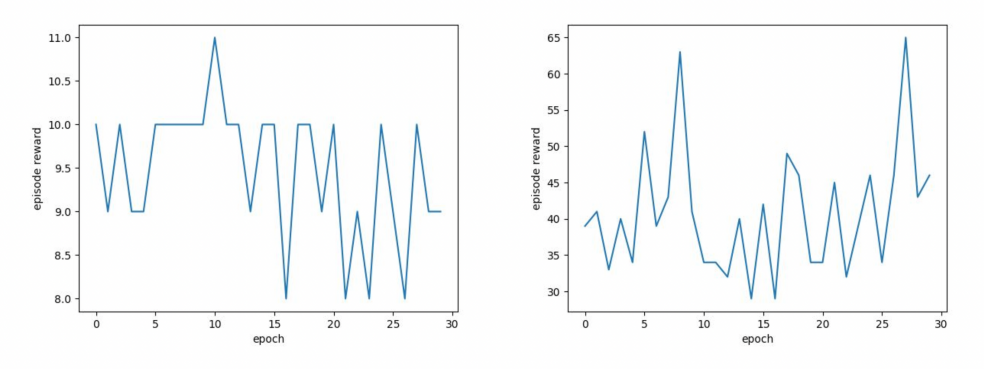

FIGURE 5 – Respectively : Cart pole with Q-learning, with classical TAMER.

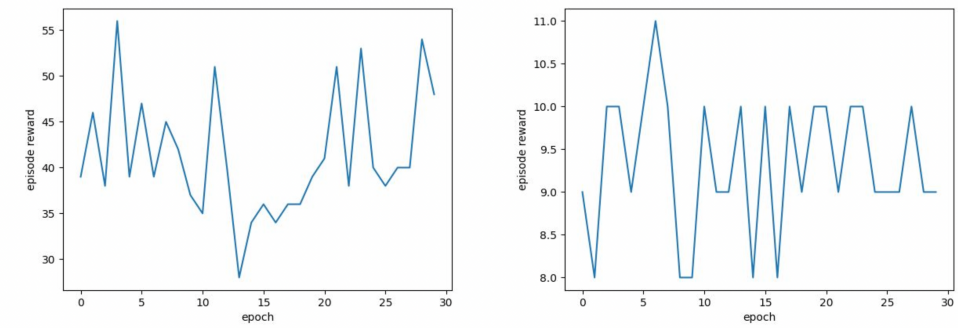

FIGURE 6 – Respectively : imperative feedback, TAMER with emotional feedback.

Conclusion

In this project we have changed the feedback modality of TAMER algorithm. We tried to provide the human trainer with an easier teaching process. To do so, we have implemented an emotion detection algorithm in TAMER to give positive and negative feedback via our facial expressions. Since we use OpenCV to captivate real time emotion, we always have a delay that we can not ignore. This delay can be compensated when we work in an environment that does not need to be particularly precise as in Mountain car. However, environments like Cart Pole that require a fast reaction and decision, the emotion detection algorithm is not officiant.

We have also implemented an imperative feedback to see how the performance of the agent will change. Training the agent using clear orders trough keyboard provided us with best results for the Cart pole environment. However we could not obtain exploitable results for mountain car environment.

Références

[1] T. R. Sumers, M. K. Ho, R. D. Hawkins, K. Narasimhan, and T. L. Griffiths, "Learning rewards from linguistic feedback," 2017. https://arxiv.org/pdf/2009.14715.pdf.

[2] T. R. Sumers, R. D. Hawkins, M. K. Ho, T. L. Griffiths, and D. Hadfield-Menell, "How to talk so your robot will learn : Instructions, descriptions, and pragmatics," 2022. https://arxiv.org/pdf/2206.07870.pdf.

[3] H. Caselles-Dupré, O. Sigaud, and M. Chetouani, "Overcoming referential ambiguity in language-guided goal-conditioned reinforcement learning," 2022. https://arxiv.org/pdf/2209.12758.pdf.

[4] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement," https://www.cs.utexas.edu/~sniekum/classes/RLFD-F16/papers/Knox09.pdf.

[5] N. Navidi, "Human ai interaction loop training : New approach for interactive reinforcement learning," 2020. https://arxiv.org/pdf/2003.04203.pdf.

[6] A. L. Thomaz, C. Breazeal, *et al.*, "Reinforcement learning with human teachers : Evidence of feedback and guidance with implications for learning performance," 2020. https://faculty.cc.gatech.edu/~athomaz/papers/Sophie-Guidance.pdf.

[7] T. B. Brown, "Rl-teacher." https://github.com/nottombrown/rl-teacher#rl-teacher.

[8] V. Srivastava, "Détection d'émotion à l'aide de python." https://geekyhumans.com/fr/detection-demotion-a-laide-de-python/.

[9] A. Lansiaux, "Human in the loop." https://github.com/SkyaX/Human-in-the-loop.

[10] G. Documentation, "Cart pole." https://www.gymlibrary.dev/environments/classic_control/cart_pole/.