Mini Project

Learning from Human Feedback

Nicola Abeni - Gianmarco Coccoli - Abdelghani Msaad - Federico Tisi

Abstract

The human feedback in reinforcement learning intends to improve the performance of a simple baseline algorithm. The aim of our work is to find and modify an existing project, trying to improve its behaviour. Starting from a little benchmark of existing approaches we decided to begin with TAMER (Training an Agent Manually via Evaluative Reinforcement). After making it work and replicating the results of the authors, we tried different *OpenAI-Gym* environments and we opted to work with *CartPole-v2*, managing also to implement two different approaches for the human feedback to improve its performance:

- Learning from Human Preferences (*https://arxiv.org/pdf/1706.03741.pdf*)
- Learning from Demonstration (`https://arxiv.org/pdf/1704.03732.pdf`)

For each different approach the starting baseline used is the standard DQL algorithm.
Here the repository where to find all the material
https://github.com/nico-abe6/AbeniCoccoliMsaadTisi-LfHF.git

1. Benchmark of similar existing projects

Benchmark				
Project	Algorithm	Environment	Feedback	Author's Score
TAMER[1]	DQL modified	MountainCar	Keyboard	~120
IRL Minigrid [2]	IRL	Gym-Minigrid	Keybord	~ 0.9613
HCR Atari[3]	DQL HCR	Montezuma	Human Replay	~ 379.1
LFHP[4]	DQL	Mujoco ant	Preferences	~ 300

[1] *https://github.com/benibienz/TAMER*
[2] *https://github.com/francidellungo/Minigrid_HCI-project*
[3] *https://github.com/ionelhosu/atari-human-checkpoint-replay*
[4] *https://openai.com/blog/deep-reinforcement-learning-from-human-preferences/*

Usually, an agent learns autonomously via environmental interactions. With the human feedback we help the algorithm teach the model better. Here's a list of the most common strategies adopted to effectively have this kind of feedback.

- **Feedback**, also known as *evaluative feedback*, is considered to communicate about the performance of a robot's action, from time to time. TAMER is an example of this;
- **Demonstration**, where a demonstration is produced by a human demonstrator (expert) with the intention of showing a state-action sequence to the robot. The paradigm assumes that the human expert shows the optimal action a_t^* for each state. *Learning from Demonstration* (*LfD*) and *Learning from Imitations* and *Behavioral Cloning* are the methodologies used. This approach could be formulated as *showing*;
- **Instructions**, that are produced by the humans with the intention of communicating the action to be performed in a given task state. This could be formulated as *telling*.

Since most of the benchmark projects are using a DQL algorithm, we decided to better understand it in order to keep it as the baseline throughout all the projects.
Its peculiarity is that it replaces the general Q-table with a neural network called Deep Q-Network. The general algorithm works this way:

- *Initialize your Main and Target neural network*
  DQL learning process uses two neural networks with the same architecture but different weights. Every N steps, the weights from the main network are copied to the target network. Using both of these networks leads to more stability in the learning process and helps the algorithm to learn more effectively;

- *Choose an action using the Epsilon-Greedy exploration strategy*
  Epsilon-Greedy method balance exploration and exploitation by choosing between them in a pseudo-random way;

- *Update your network weights using the Bellman equation*
  After choosing an action, it's time for the agent to perform the action and update the Main and Target networks according to the Bellman equation. Deep Q-Learning agents use Experience Replay, a technique used for storing the agent's experiences at each time-step.
  Here is presented the Bellman Equation:

$$\underbrace{\text{New}Q(s,a)}_{\text{New Q-Value}} = Q(s,a) + \underbrace{\alpha}_{\text{Learning rate}}[\underbrace{R(s,a)}_{\text{Reward}} + \underbrace{\gamma}_{\text{Discount rate}}\overbrace{\max Q'(s',a')}^{\text{Maximum predicted reward, given new state and all possible actions}} - Q(s,a)]$$

.

2. Replicate the Results

Following the benchmark table, the first project we managed to run was the one with the TAMER approach, that worked with the MountainCar environment. This approach allows a human to train a learning agent to perform a common class of complex tasks simply by giving scalar reward signals in response to the agent's observed actions. Based on his or her evaluation of the agent's recent performance, the trainer can choose to give reward in any form of expression that can be mapped to a scalar value (in this case through keyboard). Given the current state description, the agent's goal is to choose the action that will receive the most reward from the human. To do this, the agent models the human's "reward function" and greedily chooses actions that it expects to earn the most reward. After learning an accurate model of the human's reward, the agent can continue to perform the task in the absence of the human, choosing actions that are predicted to maximize the received reward if the human was present.

After a little bit of familiarization, the first results we have obtained were:

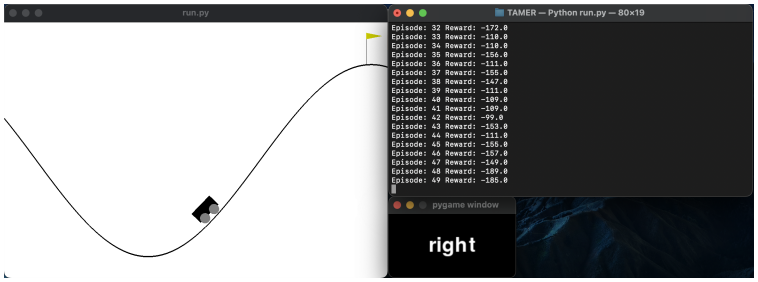

Figure 1: Results TAMER-MountainCar

The image shows the results obtained by us during one of the games we played; results that are often even better than the ones declared by the author, reported in the benchmark table. Despite this, since the MountainCar reward scheme is not helping the agent to good understand how it has played when it fails (it gives -200 every times it fails), we decided to try with other environments from the same package. Indeed, we found in CartPole a good and challenging environment: a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The pendulum is placed upright on the cart and the goal is to balance the pole by applying forces in the left and right direction of the cart.
The difference between this environment and the previous one is that here the action space is composed with only two elements: push the cart to the left or to the right. Furthermore, the reward increases itself of one point every step that the pole stays upright. The episodes can end in three different ways:

- Pole angle is grater than ± 12°;
- Center of the cart reaches the edges of the display;
- Episode length is greater than 500 (win case);

Playing with this environment was pretty challenging. In fact, despite the implementation of the human feedback, the results obtained with the simple DQL baseline were sometimes outperforming ours. Taking that into account, instead of changing the way the feedback was given (ex.: through voice or images or gestures), we decided to completely change the approach, trying with two different methods.

3. Alternative Approaches for the Human Feedback

Based on the documentation we have found online, we have decided to pursue the path of Learning from Human Preferences (*https://arxiv.org/pdf/1706.03741.pdf*) and the one of Learning from Demonstration (`https://www.cs.utah.edu/~dsbrown/readings/bco.pdf`).

Learning from Human Preferences

The aim of this approach is to solve modern RL environments using just small amounts of human feedback.
The agent starts by acting randomly in the environment and periodically two video clips of its behavior are given to the user, who decides which one of the two clips is closest to fulfilling its goal. The AI gradually builds a model of the goal of the task by finding the reward function that best explains the human's judgments. It then uses RL to learn how to achieve that goal. As its behavior improves, it continues to ask for human feedback on trajectory pairs where it's most uncertain about which is better, and further refines its understanding of the goal. The approach is well described both in the paper cited above and in this OpenAI blog (`https://openai.com/blog/deep-reinforcement-learning-from-human-preferences/`).
With CartPole it works pretty well, even managing sometimes to outperform the regular DQL. In our case, we can decide whether and when to ask for the human help. For the proposed result the algorithm asks the user's help once every five trials.

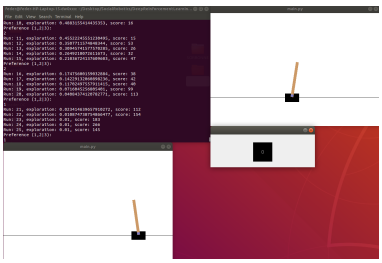

Figure 2: LfHP interface

Learning from Demonstration

Taking inspiration from the presented Minigrid project (`https://github.com/francidellungo/Minigrid_HCI-project`), we also managed to implement this interesting approach.
With regular Reinforcement Learning, we just have access to the data of the system being operated by its previous controller, but we do not have an accurate simulator of the system. The Learning from Demonstration approach overcomes this problem by giving to the agent demonstration data (games played) provided by a human pre-training. Once this pre-training has been provided, the algorithm takes it and extracts states and actions, thus being helped by the human, in order to train the agent in the best possible way. This algorithm then works like an usual Deep Q Learning, but with batches of replay that come from the human demonstration.

4. Comparison with Baseline Algorithm

We are now able to compare the learning curve of different approaches in the CartPole environment. This is possible and efficient due to the fact that the two human feedback methodologies implemented start now from the same DQL baseline.

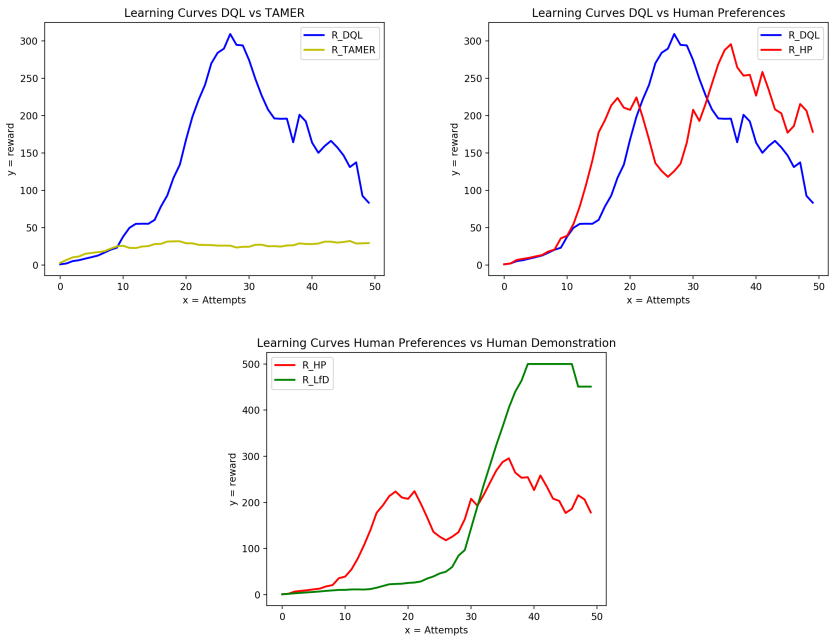

Figure 3: Different comparison

DQL vs. TAMER

The TAMER approach in this environment (CartPole) does not perform well, being beaten by the simple DQL baseline. Therefore it does not reflect the main purpose of the human feedback, which is to help the agent in the learning process.
From our point of view, the reasons of this can be found in the counterintuitiveness of the commands in this particular environment. In fact, for an innate reason, the human tents to anticipate the reward, already thinking at the future states of the agent, and this is incoherent with the algorithm dynamic. This dynamic ,infact, only takes into account the state history until the present one, being the DQN an off-policy.

DQL vs. DQL from Human Preferences

The approach with human preferences increase a little the performance, but when the user select the wrong example the learning curve is affected, obviously in a negative way.
This is exactly what is shown in the graph.
Also, sometimes the two examples showed are really close to each other and indistinguishable for human, so it becomes difficult to understand if our choices are robust or not.

DQL from Human Preferences vs. DQL from Human Demonstration

The approach with human demonstration has resulted to be the best at all, with very high performance managing to widly reach the maximum score of 500 points. This is due to the fact that, giving

initial demos, the algorithm really understands and extracts the correct data he needs to improve its training phase, starting from a reasonable policy (given with the demos).
Of course, the results are affected by how well the game is played by the human, and to simplify this we add a small delay during the playing phase.

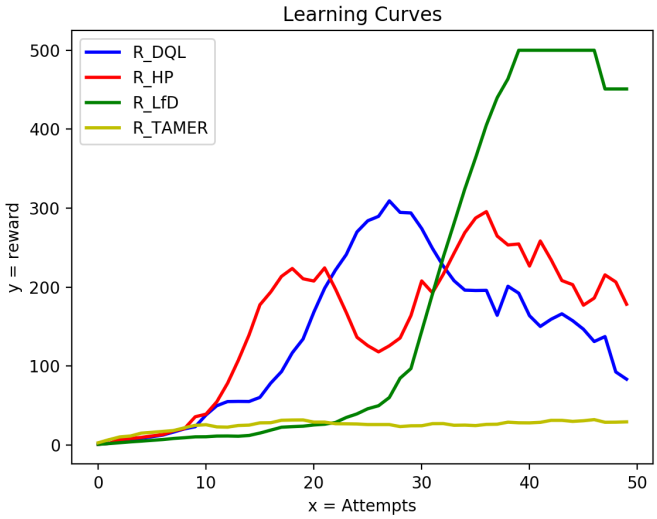

Figure 4: General comparison

5. Conclusion

In conclusion, the Learning from Demonstration approach has resulted to be the one with the best performance so far. The other two are affected by different kind of limitations, that lower the maximum score.
As said before, TAMER is counterintuitive for the *CartPole* environment, while the Learning from Human Preferences approach gives sometimes problem with the options.

Future Works

Having more time and more confidence about this subject, we should manage to:

- Applying the Learning from Demonstration also to the *Atari games* Environments (we already managed to create the key-map and to store the records of our games). It is just needed to be implemented in a RL algorithm;
- Investigate the possibility to converge the two different approaches used;
- Understand how to properly tune the hyperparameters of the baseline and maybe find another one that can perform even better and remove the problems encountered with TAMER and LfHP (Learning from Human Preferences).