Deep Reinforcement Learning From Human Preferences

Zoheir El Houari

Abstract:

This project is an attempt combine two ideas that can be very useful in RL problems, first is to explore goals defined in terms of (non-expert) human preferences between pairs of trajectory segments which would be later used as a reward function for the RL model. Secondly, presenting a general framework for exploiting the capacity of deep neural networks to approximate complex reward functions in the context of solving Reinforcement Learning problem.

Since human attention is a scares resource therefor very expensive, the main goal of the project is to maximize the agent's performance while trying to minimize the human feedback, in an attempt to reduce the cost of human oversight so that it becomes practical in modern RL systems.

Introduction:

The experiment of the project aim to leverage easily provided human feedback, as mentioned in the **DeepMind** paper.

- Experiment Description:

  My approach started by having a grid based environment with a continuous actions space, then our agent maintain a policy that interacts with the environment to produce sets of trajectories of fixed length in order for the agent to have consistency. The trajectories are later sent to human for comparison where the human should choose which trajectories is better by selecting either (1, or 2 as feedback), or if the trajectories are equally irrelevant the human can chose (0) as feedback which makes the distribution of the preferences equal. The human feedback is then mapped to a reward function estimate. We chose the reward estimate in order to minimize the cross-entropy between the agent's prediction and the actual human labels.

- The method used in the experiment goes as follows:

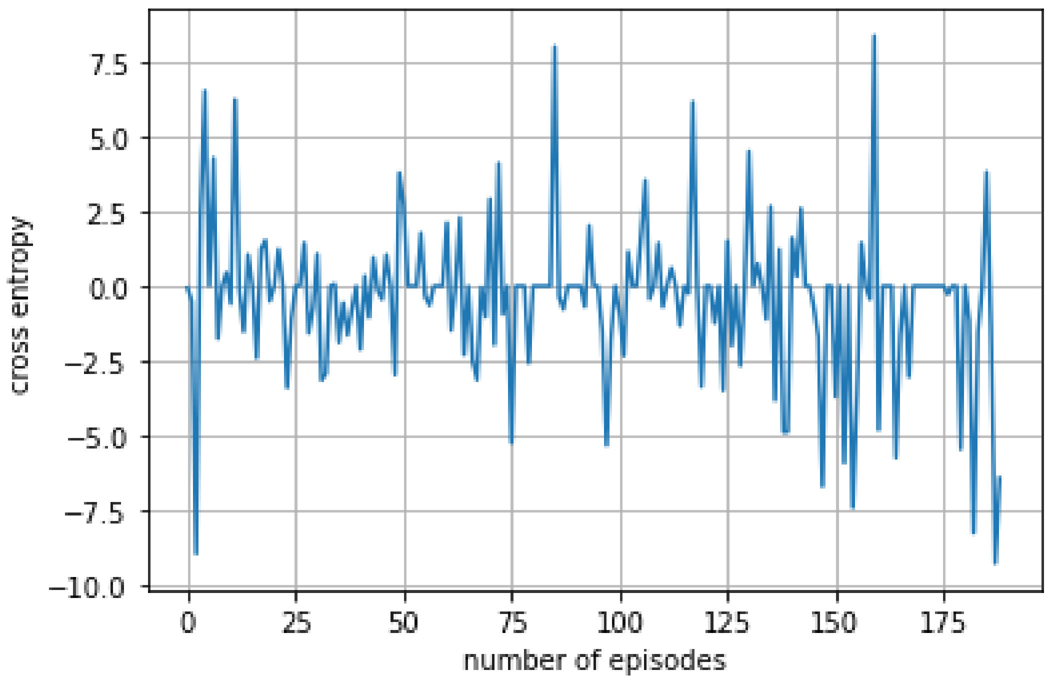
  At each point in time we maintains a policy **P** and a reward function estimate **ˆR**, each parameterized by deep neural networks.

These networks are updated by three processes:

1. The policy **P** interacts with the environment to produce a set of trajectories of length (n=14). The parameters of **P** are updated with the gradients of the expected utility in an episode, in order to maximize the sum of the predicted rewards.
2. We select two trajectories and from step 1 and render them to the human for comparison.
3. The parameters of the mapping **ˆR** are optimized via supervised learning to fit the comparisons collected from the human so far.

- Optimizing the policy:
  We use The REINFORCE algorithm to which is part of a special class of reinforcement learning algorithms called **Policy Gradient algorithms,** our agent creates a policy model, which takes an environment state as input and generates the probability of taking an action as output

Results:



Plot of cross-entropy per episode

Implementation:

Environment.py: the file contain the code of for all the functionalities of the environment from reset(), step(), render(). The environment requires the library pygame

Network.py: the file contain the two following neural networks:

- PolicyGradientNetwork: The network is used to maintain a policy that interacts with the environment to produce a set of actions that has a higher sum of rewards.
- Rewarder: The network is used to estimate the reward function of a given state, the parameters are updated by fitting the human preferences using the formula provided in the DeepMind paper

Agent.py: the file contain an implementation of the reinforce algorithm which it is used here to update policy parameters with the gradient of the expected utility in an episode.

Main.py: The file contain a main class where i run the experiment, the policy goes and create two sets of trajectories which are presented for the human comparison, the parameters of the reward function estimate are optimized to minimize the cross-entropy loss between the agent predictions and the actual human

Requirements:

- pip install tensorflow== 2.9.1
- pip install pygame
- pip install tensorflow-probability== 0.17.0
- pip install numpy pip install keras==2.9.0
- Run python main.py

Resources:

- **https://www.tensorflow.org/tutorials/reinforcement_learning/actor_critic**
- **https://github.com/ModMaamari/reinforcement-learning-using-python/blob/master/Reinforcement%20Learning%20(RL)%20Using%20Python.ipynb**

- **https://keras.io/api/layers/**
- **https://www.youtube.com/watch?v=LawaN3BdI00&ab_channel=MachineLearningwithPhil**
- **https://www.analyticsvidhya.com/blog/2020/11/reinforce-algorithm-taking-baby-steps-in-reinforcement-learning/#:~:text=REINFORCE%20belongs%20to%20a%20special,taking%20an%20action%20as%20output.**
- **https://github.com/kvsnoufal/reinforce**
- **https://proceedings.neurips.cc/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf**