Checkpoint 2

Jason Jabbour
jasonjabbour@g.harvard.edu

3/14/2023

1 Problem 1

1.1 Implementing Baselines

The title of the paper I chose to re-implement can be found using this information:

- **Paper Title**: Analysis of Explainable Goal-Driven Reinforcement Learning in a Continuous Simulated Environment
- **Paper URL**: https://www.mdpi.com/1999-4893/15/3/91.

1.1.1 Guidelines for picking the baseline

I was particularly motivated to choose this paper because it proposes an algorithm that enables reinforcement learning agents to explain their decision-making process in continuous-based simulated environments. The algorithm proposed in this paper is robust and effective in building trust in the behavior of reinforcement learning agents through its explainability. Given the relevance of this paper to my project, I believe it will be a valuable resource in my semester project.

The algorithm proposed in the paper cited here [5] is rather complex and relies on the mathematical underpinnings of Deep Q Learning, as seen on page five. It is noteworthy that the paper did not provide publicly available code, which required me to completely re-implement the algorithm based on the information presented in the paper. To do so, I had to enhance my proficiency in implementing deep reinforcement learning algorithms from scratch. In order to accomplish this, I referenced the Medium DQN Article to learn how to implement a DQN algorithm from scratch. It is worth mentioning that implementing a DQN from scratch was necessary for the paper I selected, as the algorithm introduces changes to the DQN algorithm.

As this is a reinforcement learning paper, there is no publicly available data. However, the paper used a publicly available OpenAI Gym environment to generate reproducible data for training the RL agent. Therefore, I was able to utilize this environment to train my own RL agent and produce the data in the same manner as presented in the paper.

1.1.2 Guidelines for implementing the baseline

All the code I created is organized in a methodical manner. The code is organized in two Python scripts called *run.py* and *helper.py*. For information about how to run my code and what packages are required to install in order for the code to run properly, please see the README within my folder. All the code within the *run.py* script was fully written by myself to match the descriptions made in the paper. I would like to note that the code I implemented did also make use of the *gymnasium* and *keras* packages. The gym package was necessary to launch and train the RL agent with the same environment used in the paper. Keras was necessary since I was implementing the DQN algorithm from scratch and needed a way to build a neural network with the same architecture as described in the paper. Since I implemented this code fully from scratch, I did not need to heavily rely on external resources. Therefore, my *helper.py* script is rather short.

1.2 Attempting to Reproduce Results

1.2.1 A. State Clearly what is the experiment that you tried to reproduce.

I attempted to replicate the results of Algorithm 1 proposed by the authors. The main objective of Algorithm 1 is to estimate the probability of success of an action taken by an agent through a learning-based method. The authors suggest that understanding the confidence behind an agent's decision is crucial in scenarios where humans interact with the agent, such as in autonomous driving. The proposed algorithm attempts to

provide explainability to the decision-making process of the agent, which can help to establish trust with humans. To test the effectiveness of their algorithm, the authors trained a DQN-based agent to navigate a 2D race track in the CarRacing environment available in the Open AI Gym library [1]. The agent receives a 96 x 96 x 3 matrix of pixels as input to select appropriate actions, including steering, acceleration, and braking. The authors then used their algorithm to learn the probability of success for actions taken by the agent at each timestep. I attempted to reproduce the experiment to verify if the algorithm is able to predict the probability of success of an agent accurately. The authors reported that their algorithm was able to learn actions with a probability of success of .5 during the first 75 episodes and .75 after 100 episodes.

1.2.2 B. Describe your experimental setup. Please be clear and include as many details as possible.

To implement the experiment mentioned above, I first replicated the DQN architecture presented in the paper for a continuous state environment. To achieve this, I built a DQN agent from scratch, replicating the neural network architecture within the paper using Keras. The network architecture consisted of a series of Conv2D and MaxPool layers followed by a flatten layer and two dense layers, with the final dense layer having a sigmoid activation function. The specific order of these layers are as follows: Conv2D layer, Conv2D layer, MaxPool layer, Con2D layer, MaxPool layer, flatten layer, dense layer, and finally a dense layer with a sigmoid activation function. The input observation had an input shape of 96x96x3, and the output of the network was equivalent to the number of actions an agent can choose within the environment. The authors of the paper mentioned that they modified the gym environment to allow the agent to make a combination of base actions available within the environment, leading to a total of 12 different actions. However, since the authors did not provide sufficient details about how the physics of the racing environment was modified to accommodate these 12 actions, I chose to limit my agent to only outputting 5 actions. These 5 actions consist of steer left, steer right, gas, brake, and do nothing. It is important to note that the objective of this experimental setup was not to create a perfect physics simulator, but rather to replicate the DQN architecture and algorithm presented in the paper. The environment I used was the same CarRacing OpenAI gym environment used in the paper. The objective of this environment is to autonomously drive a vehicle around a race track in 2D. The observations at each time step is a 96x96x3 matrix of pixels which convey the top down view of the vehicle in the environment. To compile my model, I used an Adam optimizer and categorical cross-entropy loss. I used the hyperparameters specified by the authors, including an initial epsilon set to 1 with a .9999 decay rate, and a learning rate of .001. However, the authors did not specify the batch size, tau, gamma, or episode length, so I used commonly used values of 4, .125, .95, and 500, respectively. To replicate the algorithm presented in the paper, I modified the DQN agent by adding a Q-model and a P-model, which are used to learn the q-values and probability values. The neural network architecture for the P-model was changed to use a Dense layer with a linear activation function to follow the authors' description. I updated the approximate target values for the Q and P functions as described in the algorithm, and modified the DQN learning process by including function objectives for both Q and P. To validate my implementation with that of the paper, I plotted the probability of success of the timesteps I trained my agent on, as seen in Figure 6 of the paper. Since training this algorithm takes many hours and a lot of compute power, I only trained my agent for 8 episodes of 500 timesteps each. This was done due to obvious resource restrictions of mine. In a perfect scenario, I would train the agent for all 500 episodes as described in the paper. In order to create a plot of the probability success learned throughout the training process, I saved the action taken, reward, and success probability at every timestep after each episode was completed. The agents were also saved at the end of each episode. This was done more often than what was specified in the paper's Algorithm 1 in order to create more frequent backups for the models I trained.

1.2.3 C. Were you able to reproduce the result in the paper?

Through the implementation of the experiment I described above, I was able to replicate the Algorithm 1 but was unable to verify that the Algorithm reaches a success probability of around .75 after 100 episodes. Within this paragraph, I will describe my limitations and constraints that prohibited me from fully reproducing the results in the paper. I will also hypothesize other reasons that restricted the reproduction of the results mentioned within this paper. The biggest restriction I faced was not having all the values of the hyperparameters used within the author's implementation of this algorithm. There were several important hyperparameter values that were not mentioned by the authors. These hyperparameters include the seed set, the minimum epsilon value that is allowed after decay, the number of timesteps within each episode, the batch size, the specific activation functions for each layer of the DQN neural network, how the authors modified the physics of the vehicle to allow for a 12 dimension action space, the maximum queue length for storing policy memory, and the loss function for the neural networks. Tuning the values of each of these parameters is crucial for an agent to perform in a consistent manner. There has been lots of previous research done in the field of reinforcement learning that show how minor changes in an RL agent implementation can cause different results [2]. The change in parameter settings within the context of reinforcement learning has been described as a lottery. Meaning, the values of the parameters described above that were not properly documented by the authors of this paper are necessary for reproducing the results of their paper. Another limitation I faced was training the RL agent for the same number of episodes as described in the paper. Training the agent for only 8 episodes took my laptop several hours. I would need to train for all 500 episodes or at least several dozen episodes to verify the results of the paper. Training for a short number of episodes such as 8 episodes is very restrictive in reinforcement learning since performance can be extremely variable at the beginning stages of the training process. This is due to the exploration/exploitation trade-off paradigm that RL researchers face. I tried my best to train my agent multiple times across a few different values for the unknown parameters. Specifically, I tried a longer number of timesteps per episode (1000 timesteps per episode), larger batch sizes such as 8, 16, and 32 instead of 4, and a tau value of .05 instead of .125. Since some of these hyperparameters slowed down the training process even more, I was still unable to train for a large number of episodes. Lastly, it seems as if the algorithm the authors are proposing is computationally inefficient. While I was training the algorithm, my laptop would occasionally stumble across segmentation faults. This shows how more memory is being requested than what is available on my hardware. It is important to note that this paper did not provide an appendix and did not provide their source code. Because of these limitations, it is hard to understand every implementation detail chosen by the others. I would like to state that I don't believe there was anything particularly erroneous in the author's implementation. However, the lack of details provided by the authors make replicating their work difficult.

1.3 Critiquing Research Papers

1.3.1 List 2 or more contributions of the paper that differentiate it from prior work.

The paper made a significant contribution by adapting an algorithm that predicts the probability of success behind the actions of a reinforcement learning agent to a continuous state space. This is in contrast to previous work, which primarily focused on explainability in discrete state space environments [4]. The paper presented a novel approach that can explain an agent's decision-making process through a probability regardless of the state-space type of the environment, thereby expanding the applicability of the explainability algorithm to a wider range of environments. Another significant contribution of this paper that sets it apart from previous work is its novel methods for estimating the probability of success using learning-based and introspection-based approaches, rather than relying solely on memory-based methods [3]. The new estimation techniques presented in this paper were shown to be more efficient in terms of memory and compute usage, which is a critical consideration in real-world applications where computational resources are often

limited. This advancement makes the approach more practical and scalable for a wider range of applications. The contributions of this paper are not only novel but also provide significant real-world benefits. By utilizing learning-based and introspection-based approaches, the paper presents a more efficient method for estimating the probability of success, which enables reinforcement learning agents to incorporate an element of explainability in a wider range of environments. This is a crucial aspect of modern AI research as it promotes transparency and trust in AI systems, making them more accessible and useful for a diverse range of stakeholders.

1.3.2 Comment on the clarity of writing

The authors did a commendable job of introducing the problem and highlighting the need for RL agents to have an element of explainability. They provided a good background by discussing the use of RL in human-agent interaction and how trust is important in such scenarios. This helped to establish the importance of explainability in building trust. The authors then proceeded to describe the two proposed architectures for predicting success probability, which was helpful in understanding the neural network architecture. However, the pseudo code of Algorithm 1 and Algorithm 2 could have been clearer. Some important elements were not clearly described in the writing around the pseudo code. For instance, a better description of the difference between the Q and P functions, as well as how these functions differ from the Q and P objective functions, would have improved the clarity of the paper. Furthermore, to ground the difference between their proposed algorithm and a baseline DQN algorithm, it would have been helpful for the authors to provide a quick background on how a default DQN algorithm works. This would have helped readers to understand the comparison between the default algorithm and the proposed algorithm. The authors' experimental setup in the CarRacing environment was generally comprehensible. However, the paper would have benefited from a better description of the hyperparameters used to produce the experimental results. This would have improved the clarity of the paper and allowed readers to better understand how the authors arrived at their conclusions. Overall, the paper was well written, but there were a few areas where clarity could have been improved. By providing better descriptions of the differences between algorithms and providing more context to the experimental setup, the authors could have made the paper more accessible to a wider range of readers.

1.3.3 Were all the proofs, algorithms, and other technical aspects of the paper correct?

Overall, the technical aspects of the paper appear to be sound. However, there are a few areas where the authors could improve their argument and analysis. Firstly, the authors argue that their proposed explainability methods are computationally more efficient than previous memory-based methods. However, the authors chose a significantly inefficient algorithm, which raises questions about the validity of their claim. It would have been interesting if the authors had compared the performance of their algorithm with other state-of-the-art algorithms such as Soft Actor-Critic, to further bolster their argument. I should mention that during the training process, the algorithm would occasionally request more memory than what is available on my system causing a segmentation fault. Therefore, I am a bit hesitant to make any conclusions about the computational efficiency of their algorithm. Secondly, while the authors showed success in a continuous state space environment, their results were only visible in one environment. It would have been stronger if the authors had tested their algorithm on multiple different continuous state space environments to demonstrate its robustness and applicability to various real-world scenarios. Finally, the authors did not provide any intuition or qualitative reasoning behind why their algorithms work so well. This could have been strengthened by providing the reader with a clear understanding of the theory behind their algorithm and how it relates to the problem being addressed.

1.3.4 Did the experiments justify the claims in the paper?

The experiments conducted by the authors serve as a strong baseline to demonstrate the functionality of their algorithm. However, to further justify the claims made in the paper and establish the algorithm's suitability for broader use, there are a few areas where the authors could have tested their algorithm more rigorously. Firstly, the authors should evaluate their algorithm on several different environments, not just the CarRacing environment. This would help demonstrate the algorithm's robustness and applicability to a range of scenarios. Secondly, it would be beneficial for the authors to evaluate the performance of their algorithm when the agent is initialized at various seeds. This would help ensure the replicability of their findings within the community and help establish the algorithm's generalizability. Finally, given that the maximum average success probability that their algorithm could predict is .75, the authors should address what improvements could be made to improve this probability in future work. This would help clarify the limitations of the current algorithm and guide future research in this area.

1.3.5 Did you find it easy to implement and reproduce the results in this research paper?

The authors did a good job explaining the general concept of their algorithm and the experiments conducted to produce the reported results. However, there are some areas where the paper could be improved to enhance the reproducibility of the work. The paper lacks clarity on both the algorithmic and experimental side, making it difficult to implement and reproduce their work. From the algorithmic perspective, several important details were left out of the pseudocode, which are crucial to understanding and implementing the algorithm correctly. From the experimental side, the authors did not provide all values of hyperparameters used in their experiments. To enhance the reproducibility of their work, it would be beneficial for the authors to provide a baseline implementation in their Github repository or attach additional descriptions in the appendix. Providing a baseline implementation or additional descriptions of the hyperparameters would help researchers to more easily reproduce the results and build on the authors' work. This would also make it easier for other researchers to understand and implement the algorithm correctly, potentially leading to more widespread use and further development in the field.

1.3.6 Rating for Fun

Due to the novelty and exciting preliminary results of this paper, I would assign this paper a score of 4 out of 5. Again, the paper did a wonderful job motivating the need for explainability in reinforcement learning systems and produced some promising experimental results. However, the paper was lacking in a details that would make their work easier to reproduce and integrate into real-world RL systems.

References

[1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[2] S. C. Chan, S. Fishman, J. Canny, A. Korattikara, and S. Guadarrama. Measuring the reliability of reinforcement learning algorithms. *arXiv preprint arXiv:1912.05663*, 2019.

[3] F. Cruz, R. Dazeley, and P. Vamplew. Memory-based explainable reinforcement learning. In *AI 2019: Advances in Artificial Intelligence: 32nd Australasian Joint Conference, Adelaide, SA, Australia, December 2–5, 2019, Proceedings 32*, pages 66–77. Springer, 2019.

[4] F. Cruz, R. Dazeley, P. Vamplew, and I. Moreira. Explainable robotic systems: Understanding goal-driven actions in a reinforcement learning scenario. *Neural Computing and Applications*, pages 1–18, 2021.

[5] E. Portugal, F. Cruz, A. Ayala, and B. Fernandes. Analysis of explainable goal-driven reinforcement learning in a continuous simulated environment. *Algorithms*, 15(3):91, 2022.