Final Report on Solitaire Using a Deep Q Reinforcement Learning Network

Author: Grant Deljevic

Introduction

This project aimed to create a Deep Q-Network (DQN) model to autonomously play 3-card Draw Solitaire. The focus was on developing a system capable of learning an effective strategy for playing Solitaire using reinforcement learning techniques.

Methodology

The methodology employed Deep Q Networks, known for their effectiveness in handling complex state spaces, to enable strategic decision-making in the game of Solitaire. I custom-built a Solitaire environment to simulate the actual game, incorporating a deck of cards and a table structure with various columns and piles. The DQN model was designed with an input of encoded choices and game state, utilizing dense linear layers and the ReLU Activation Function. It included a policy network for decision-making and a target network to ensure stability during training. Training involved initializing replay memory, batch learning, loss calculation, and frequent updates of the model, using an Epsilon-Greedy Strategy for action selection.

Challenges and Solutions

Several challenges were encountered and addressed. The complexity of representing the Solitaire game state was overcome by developing a numerical encoding function. We also created a system to identify and encode all possible actions, enabling the network to effectively narrow down its choices. Additionally, we devised a method to represent the consequences of actions, storing future states and probabilities in an action state tree.

Results

The model's training was monitored through metrics like loss and rewards. Observations showed that the loss converged, but the reward stayed stagnant, indicating the model was not actually making satisfactory changes. This would hopefully be fixed by the change in encoding the action state and other future improvements.

Conclusion

The project achieved its goal of developing a functioning DQN, but failed to create one that was well suited to Solitaire or learned the game. This was due to the nature of the environment needing to be set up. Key learnings include insights into the complexities of applying reinforcement learning to even a seemingly simple game, and the many different factors which could stop a model from learning.

Future Work and Improvements

Plans for future improvements include hyperparameter tuning, finalizing the encoding of actions and integrating it with the state representation, creating the action space tree, and optimizing functions for full GPU utilization.

Citations

1. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533. https://doi.org/10.1038/nature14236

2. Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press. http://incompleteideas.net/book/the-book-2nd.html

3. PyTorch. (n.d.). Documentation. PyTorch. https://pytorch.org/docs/stable/index.html

4. Python Software Foundation. (n.d.). Python 3.9.1 documentation. Python.org. https://docs.python.org/3/

5. TensorFlow. (n.d.). TensorBoard: TensorFlow's visualization toolkit. https://www.tensorflow.org/tensorboard

6. Python Software Foundation. (n.d.). Unittest — Unit testing framework. Python 3.9.1 documentation. https://docs.python.org/3/library/unittest.html

7. Parts of language used for this paper - OpenAI. (2023). Chat with ChatGPT. Retrieved from https://chat.openai.com/share/78cf5e7e-9787-4fea-b461-f809c5fc4f43