IART Segundo Projeto - Aprendizagem Por Reforço

Diogo Oliveira Reis, up201405015
João Pedro Oliveira Lírio,
up201705254
FEUP

I. Introdução

O objetivo do segundo projeto da disciplina de Inteligência Artificial foi implementar um agente de aprendizagem por reforço a um jogo de tabuleiro chamado "Parquet".

O jogo "Parquet" trata-se de um simples jogo de tabuleiro cujo objetivo é conseguir colocar três peças no canto do tabuleiro do adversário. Cada jogador só pode movimentar as suas peças para a frente, e sempre que realiza um movimento pode mexer umas das quatro peças especiais situadas no meio do tabuleiro quando o jogo é iniciado. Cada jogada corresponde a dois movimentos, o primeiro para a movimentação da peça da cor do primeiro jogador e o segundo movimento corresponde à movimentação das peças "Void".

Devido à complexidade deste jogo e ao facto de nunca termos tido contacto com o programa "Unity", decidimos simplificar as regras de maneira a tornar os resultados da análise dos algoritmos de aprendizagem por reforço melhores. Esta simplificação passou por reduzir o número de peças e o número de movimentos que cada jogador pode realizar. Nesta implementação o objetivo continua a ser chegar ao canto do adversário, e a existências das peças "Void" manteve-se.

II. Descrição do problema

Para treinar um agente utilizando aprendizagem por reforço podemos aplicar vários algoritmos como Q-Learning, SARSA, Proximal Policy Optimization (PPO), "Soft Actor-Critic" (SAC), "Generative Adversarial Imitation Learning" (GAIL), entre outros.

Optamos por nos focar em PPO e SAC, visto que são suportados nativamente pela ferramenta "MLAgents".

Fugindo um bocado à aprendizagem por reforço implementamos um outro algoritmo suportado pelo "MLAgents" denominado por "Generative Adversarial Imitation Learning" ou GAI. Este algoritmo revelou-se muito interessante porque o Agente aprende com base num exemplo dado pelo utilizador.

No programa "Unity" gravamos uma demonstração do utilizador a jogar o jogo, ou seja, este algoritmo leva em consideração a abordagem de um humano.

O tipo de treino especificado no ficheiro continua a ser PPO, a diferença é que este algoritmo leva em consideração uma demonstração quando começa o treinamento, pelo que achamos pertinente incluir isto no relatório.

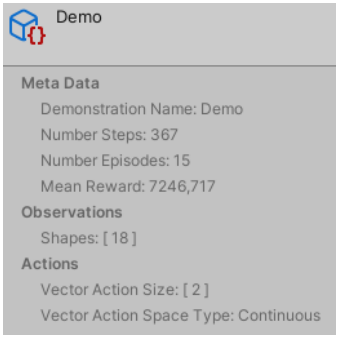

Figura 1: Ficheiro de demonstração criado no Unity

III. Abordagem

Para aplicar algoritmos de aprendizagem por reforço a um agente primeiro foi necessário desenvolver o jogo em si, recorrendo ao programa "Unity".

Utilizamos a ferramenta "Canvas" em "Unity" para tratar de toda a representação gráfica do projeto e criamos vários "Scripts" em C# para tratar da jogabilidade. Estes "Scripts" estão localizados na pasta "Assets".

A principal razão para a utilização do programa "Unity" foi para conseguirmos usar uma ferramenta "Open Source" de treino de agentes chamada "MLAgents". Esta ferramenta revelou-se crucial na implementação dos agentes.

Para medir o treino dos agentes utilizados outra ferramenta chamada "TensorBoard", que gera automaticamente gráficos de recompensa ao longo do tempo, entre outras informações interessantes para este projeto.

IV. Resultados experimentais

Foram realizados muitos testes com vários parâmetros e algoritmos.

O valor da recompensa que os algoritmos usam é a distância ao canto do adversário, quando mais perto estiverem do canto, maior será este valor. O valor da recompensa é descontado sempre que a peça sai do tabuleiro ou atinge uma peça "Void" ou a peça adversária.

Estes testes estão localizados na pasta "MLAgents\summaries".

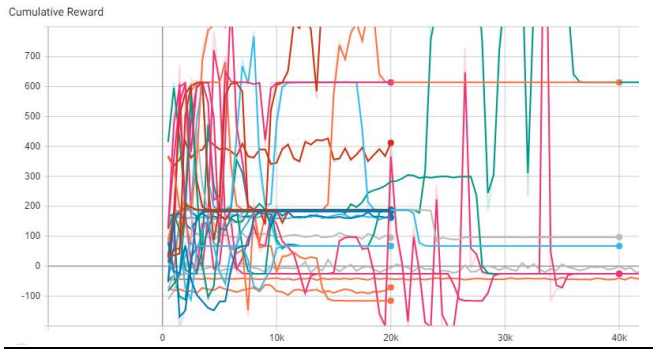

Figura 2: Gráfico de recompensa de todos os testes realizados

De entre todos os testes realizados destacam-se os seguintes.

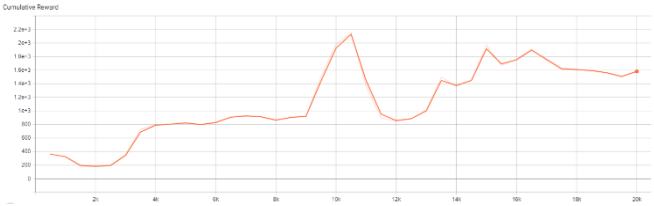

Figura 3: Gráfico de recompensa do melhor teste usando PPO

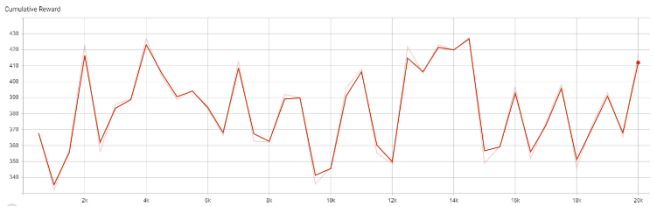

Figura 4: Gráfico de recompensa do melhor teste usando SAC

A figura 5 é interessante porque representa os testes realizados usando a rede neural do melhor teste usando o algoritmo PPO.

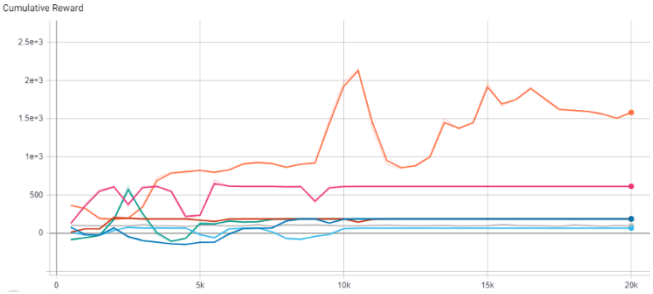

Figura 5: Gráfico de recompensa usando a rede neural do melhor teste usando PPO

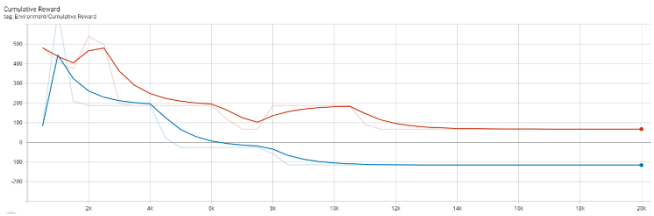

Figura 6: Gráfico de recompensa dos melhores testes usando GAI

Como o algoritmo GAI é menos relevante foram corridos signifivamente menos testes. Os resultados esperados seriam os mesmo que no PPO mas em teoria surgiriam mais cedo devido ao uso das demonstrações.

V. Conclusão

Foi possível implementar múltiplos agentes capazes de realizar aprendizagem por reforço sobre uma versão simplificada do jogo "Parquet".

Optamos por simplificar as regras do jogo devido ao facto de nunca termos utilizado a ferramenta "Unity", pelo que a implementação do jogo em si se revelou mais demorada que o previsto.

No entanto, o principal objetivo deste projeto era implementar diversos algoritmos de aprendizagem por reforço e explorar os seus resultados.

Os resultados da implementação destes algoritmos foram bons, o agente aprendeu por reforço a jogar esta versão simplificada do jogo "Parquet"

Com a implementação dos algoritmos "Proximal Policy Optimization" (PPO), "Soft Actor-Critic" (SAC) e "Generative Adversarial Imitation Learning" (GAIL) assim como a criação de vários ambientes de teste com diferentes parâmetros, realização e análise de gráficos de aprendizagem, o objetivo foi cumprido.

Referencias

Juliani, A., Berges, V., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D. (2020). Unity: A General Platform for Intelligent Agents. *arXiv preprint* arXiv:1809.02627. https://github.com/Unity-Technologies/ml-agents.

Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in Neural Information Processing System*