Projeto: Super Mario World - Inteligência Artificial - 2023.Q3

Prof. Fabrício Olivetti de França (folivetti@ufabc.edu.br)

Enunciado

Para esse projeto vocês devem implementar um agente inteligente capaz de jogar a fase *YoshiIsland2* do jogo Super Mario World utilizando um dos algoritmos apresentados em aula na segunda metade do curso, do livro texto ou de algum artigo científico.

Os algoritmos devem ser escritos em Python e farão uso da biblioteca Retro Gym (https://github.com/openai/retro).

Para carregar o jogo você deve necessariamente utilizar essa linha de código:

```
env = retro.make(game='SuperMarioWorld-Snes', state='YoshiIsland2', players=1)
```

Instruções de instalação:

- Instale a biblioteca Retro Gym seguindo as instruções em: https://github.com/openai/retro
- Copie a ROM do jogo para o diretório *site-packages/retro/data/stable/SuperMarioWorldSnes/* com o nome *rom.sfc* (se estiver utilizando o Anaconda, ele deve estar em *~/anaconda3/lib/python3.11/* ou diretório similar)
- Se tudo estiver funcionando corretamente, você conseguirá executar os scripts marioRule.py e marioAstar.py
- Estude os códigos marioRule.py, rominfo.py, utils.py e marioAstar.py para entender o funcionamento da biblioteca.

Entregas

05/12/2023 - Entrega final: códigos, agente treinado, um README explicando qual o algoritmo utilizado e quais bibliotecas necessárias.

IMPORTANTE: O README deve conter nome e RA do aluno.

Dentre os arquivos, o repositório deve conter:

- Um código-fonte chamado `train.py` que é utilizado para treinar o agente
- Um código-fonte chamado `play.py` que é utilizado para jogar o melhor agente atual

Ambos os códigos serão testados quanto ao seu funcionamento. O código `train.py` deve permitir continuar o treinamento desde sua última execução, ou seja, se o professor executar esse código em sua máquina local ele deve melhorar (se possível) o melhor agente atual.

Será permitido utilizar bibliotecas que implementam os algoritmos de aprendizado de agentes. O código ainda assim deve estar bem estruturado, organizado

e comentado. Os comentários servem de documentação quanto ao uso das bibliotecas utilizadas.

Avaliação

A nota será atribuída em relação a:

- P1: organização e estruturação do código ($[0, 3]$)
- P2: corretude das soluções ($[0, 4]$)
- P3: rank no progresso dentro da fase ($[0, 3]$)

O rank no tempo de execução será:

- Os 10% melhores agentes: 3 pontos
- Os 10% piores agentes: 1 ponto
- Entre esses dois: 2 pontos
- Caso o agente não ultrapasse o desempenho do código *marioRule.py*: 0 pontos

Não serão permitidas soluções *hard-coded* (e.x., rule-based agent).