Inteligência Artificial - Projeto EvoMan

Prof. Fabrício Olivetti de França, Prof. Denis Fantinato

3º Quadrimestre de 2019

EvoMan

Introdução

O objetivo desse projeto é entender na prática os conceitos de Inteligência Artificial aprendidos em sala de aula, principalmente com foco na construção de agentes inteligentes. Para tanto, nesse projeto será utilizada uma plataforma de aprendizado de agentes para video games feita em Python chamada EvoMan. O projeto deve ser feito em grupos de até 03 (três) pessoas. A entrega do projeto deve ser feita via Github Classroom, um repositório por grupo, contendo todo o código-fonte necessário para o aprendizado e a execução do agente (já deve ser fornecido o agente final) e um relatório de 03 (três) páginas descrevendo o algoritmo utilizado e os resultados obtidos.

Prazo de entrega

O prazo final para a entrega será no dia 04 de dezembro, quarta-feira, até às 23:00. Esse prazo **não é negociável**, trabalhos entregues com atraso receberão desconto de 0.5 ponto por hora.

Instalação e Configuração

O código inicial, manual de uso e todos os recursos extras necessários estão disponíveis no Github:

- `git clone https://github.com/karinemiras/evoman_framework.git`
- siga as instruções de instalação no arquivo *evoman1.0-doc.pdf*
- execute o script de demonstração *controller_specialist_demo.py* para testar a instalação
- jogue o jogo utilizando o teclado para entender a dificuldade do problema. Utilize o script *human_demo.py*.

Tarefa

Implemente um algoritmo de construção de agente inteligente entre os algoritmos vistos em aula, aqueles que estão no livro oficial do curso ou provenientes de algum artigo científico.

O agente deve ser treinado utilizando o modo *Individual Evolution* da plataforma EvoMan com o objetivo de obter um agente que vença o maior número de adversários. Para isso deverão ser escolhidos 4 adversários que poderão ser utilizados durante o treino e 4 adversários que serão utilizados apenas para testar o Agente final.

Entrega

A entrega deverá ser feita via Github Classroom (link a ser definido) contendo todo o código necessário para treinar o agente, o arquivo contendo o agente final e o código para executá-lo. Além disso deve ser entregue um relatório em PDF seguindo o formato IEEE de conferências (link a ser definido) contendo a descrição PEAS do agente, descrição do algoritmo utilizado e um resumo e análise dos resultados contendo quais adversários foram escolhidos para treino, quais para testes, quais adversários o agente consegue vencer, a energia final do agente, a energia final do adversário (caso o agente perca) e o tempo de duração da luta.

Observações

Não serão permitidos:

- o uso do código de computação evolutiva e neuroevolução contida no repositório.
- qualquer alteração ao *core* da plataforma EvoMan (pasta *evoman*).
- caso o algoritmo utilizado dependa de valores aleatórios, os experimentos devem ser repetidos 10 vezes e a média dos resultados deverá ser reportada.
- O parâmetro `level` deverá ser setado como 2 e o parâmetro `contacthurt` setado como `player` (ambos valores padrões).
- Os outros parâmetros estão liberados para serem alterados, contanto que sejam reportados no relatório final.

Relatório

A estrutura do relatório deve conter:

- Uma breve introdução do problema (máximo 2 parágrafos)
- Descrição básica do agente (PEAS)
- Descrição do algoritmo utilizado (1 página)
- Experimentos e resultados obtidos (1 página)
- Análise dos resultados e conclusão

Não se esqueçam de citar os trabalhos da literatura que foram utilizados no projeto. Não citem os slides de aula, mas os artigos originais dos algoritmos.

Notas

O projeto completo e correto contará como 5 ptos a serem distribuídos nas notas de prova de forma a maximizar a média harmônica. Além disso, os três primeiros colocados recebrão bônus de 1 pto para o 1o. colocado, 0.6 pto para o segundo colocado e 0.3 pto para o terceiro colocado.

Competição inter-universidades

O melhor agente da turma competirá com o melhor agente da turma de Computação Evolutiva da Vrije Universiteit Amsterdam. Se nosso melhor agente ganhar, toda a turma ganha 0.6 pto extra!

Dicas gerais

- O arquivo de demonstração mais limpo é o *dummy_demo.py*, esse pode ser um bom ponto de partida para implementar seus algoritmos.
- Utilize fontes grandes quando plotar os gráficos de resultados, dessa forma eles continuam legíveis mesmo com a restrição de espaço do formato escolhido.
- Teste seu algoritmo com poucas iterações e/ou apenas um adversário para ter certeza que ele está funcionando corretamente. Lembrem-se que o custo computacional de aprendizado para esse problema é bastante alto.
- Controle o tempo de seus experimentos para planejar a execução do experimento final.
- Implemente funções para salvar e recuperar resultados intermediários. Não subestime a lei de Murphy!
- Implemente um código que leia o agente final e mostre a luta contra todos os adversários.