Bruno Storer Martins - RA: 11201721797
Gabriel Bicudo Moreira - RA: 11201720100
Gabriel Rios Souza - RA: 11201720940

Agente Inteligente para Evoman

Monografia apresentada à Universidade Federal do ABC como parte dos requisitos para aprovação na disciplina de Inteligência Artificial do Curso de Bacharelado em Ciência da Computação.

Professor: Fabrício Olivetti de França

Santo André, SP

2019

1. Introdução:

O Framework EvoMan é uma plataforma para desenvolver e testar algoritmos de otimização, e foi desenvolvido na Universidade Federal do ABC em Santo André - São Paulo Brasil durante a pesquisa de um projeto de mestrado. (MIRAS, 2019)

Esse trabalho tem como objetivo implementar um algoritmo de construção de agente inteligente. 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 escolhemos 4 adversários para serem utilizados durante o treino e 4 adversários que a serem utilizados apenas para testar o agente final.

2. Descrição Básica do Agente (PEAS):

- Performance:
  - maior dano no adversário;
  - maior quantidade de vida restante após o final da batalha;
  - menor tempo para vencer a batalha.

- Environment:
  - adversário;
  - projéteis (tiros)
  - campo de batalha;
  - agente inteligente.

- Actuators:
  - andar para a direita;
  - andar para esquerda;
  - pular;
  - cair;
  - atirar;

- Sensors:
  - sensor de proximidade do projétil (tiro);
  - sensor de distância do adversário;
  - sensor de direção do agente;
  - sensor de direção do adversário.

3. Descrição do Algoritmo Utilizado:

Para treinar nosso agente, utilizamos uma rede neural com 3 camadas (1 de entrada com 20 neurônios que correspondem às 20 variáveis do ambiente, 1 hidden layer com 13 neurônios e função de ativação ReLu e 1 de saída com 5 neurônios com função de ativação SoftMax que correspondem as 5 ações possíveis que o agente pode fazer..

A atualização dos pesos é feita através do Algoritmo Genético com Estratégia Evolutiva, cada geração contém 10 redes neurais (iniciadas com pesos aleatórios), a partir delas, aplicamos o processo de cruzamento, seleção e mutação do algoritmo genético.

O cruzamento é feito sempre com duas redes neurais (pais), escolhidos pela pontuação feita ao jogar o jogo (valor de fitness), a parta levada ao filho de cada pai é escolhida de maneira aleatória.

A seleção da próxima geração é feita entre todos os pais e os filhos gerados, a escolha é baseada em uma probabilidade de selecionar só os melhores pais e seus filhos ou de também selecionar alguns indivíduos que não tiveram resultados satisfatórios. Isso é feito para remover o viés que alguns indivíduos podem ter.

Por fim, com os indivíduos selecionados para próxima geração, há também uma probabilidade nos filhos de que seja alterado alguns de seus pesos por valores aleatórios, como forma de mutação no processo. A escolha dos valores e probabilidades para cada processo será discutido no próximo tópico.

4. Experimentos e Resultados Obtidos:

A fim de obter um resultado interessante alterou-se diversas vezes a ordem dos adversários aos quais nosso agente foi treinado, com isso, percebe-se facilmente que esta ordem influencia muito no desempenho final do mesmo. Notou-se que intercalando chefes aos quais atiram bem mas não pulam e nem desviam com esta mesma objetividade com chefes que possuem uma habilidade interessante em pular e desviar porém não atiram relativamente bem, o resultado foi mais satisfatório, com uma mescla dessas habilidades. Foi utilizado os chefões 7,2,3,8 respectivamente para treinamento.

Por cruzamento, são geradas cinco redes neurais filhas da seguinte maneira: o primeiro pai possui uma probabilidade de 50% de ser uma das redes neurais com maior valor de Fitness(a que chamamos de pais bons) e 50% ampla concorrência, ou seja, qualquer uma das redes neurais pais. O segundo pai também segue esta mesma regra. Deixando assim, uma prioridade aos pais bons se reproduzirem e propagarem suas características para as próximas gerações.

Para cada próxima geração há uma possibilidade de 60% de cada um dos cinco pais com menor valor fitness serem substituídos por essas redes neurais filhas geradas. Já os pais com maior pontuação possuem uma probabilidade de 10% de serem substituídos por essas mesmas redes neurais filhas. Vale ressaltar que primeiramente definiu-se que os cinco melhores pais iriam automaticamente para a próxima geração, porém o código ficaria muito enviesado desta maneira.

Dessas redes neurais que sobraram no processo, há uma probabilidade de 40% da ocorrência de mutação em qualquer rede neural filha, com o intuito de aumentar o aparecimento de novas característica ao robô. Testou-se uma possibilidade de 5% do aparecimento de um novo indivíduo

nesta população, gerado de forma completamente aleatória para o mesmo propósito, porém esta medida acabou atrapalhando a reprodução de características boas. Por isso foi retirado.

A partir disso inicia-se uma nova geração.

Percebeu-se que com um valor de fitness próximo à 84, nosso agente já conquistava vitória perante ao chefe a que estava enfrentando. Por isso definiu-se que o robô só enfrentaria o próximo adversário se sua população obtivesse uma somatória de fitness maior que 800, valor previsto para que pelo menos um dos indivíduos da população vencesse o chefe, e os outros indivíduos também obtivessem um valor interessante de fitness(com alguns também podendo vencer o adversário).

Somente a título de observação, o código do projeto roda por uma hora por questões física do computador. Após isso,o jogo é fechado e os controles do estado atual bem como o chefão a que estava enfrentando são salvos em arquivos, e o código pode ser iniciado novamente utilizando esses parâmetros ou sendo iniciado do zero. O agente foi fruto de várias execuções deste código.

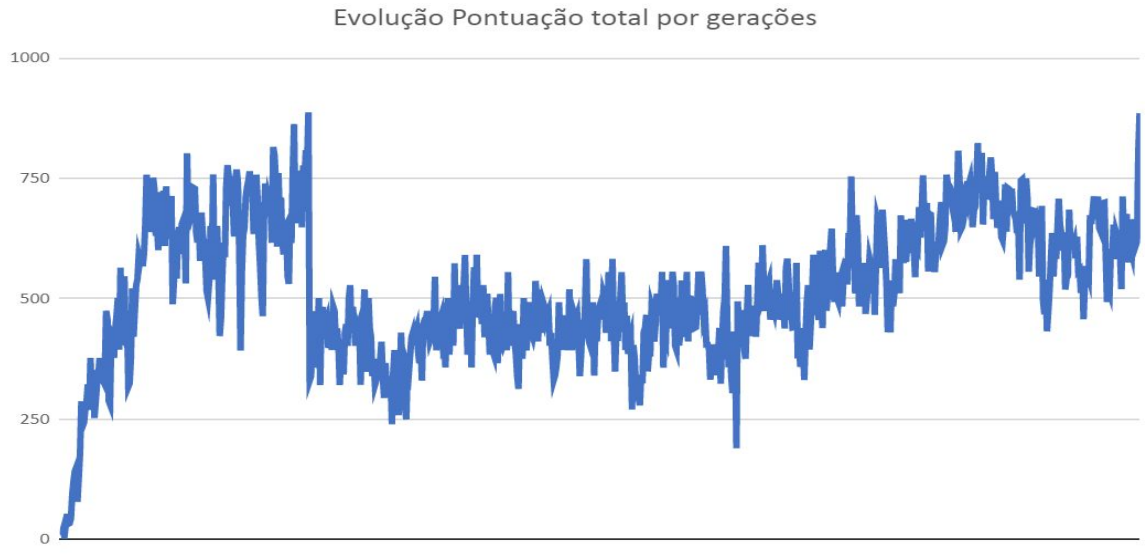
Arquivos do projeto:

-Projetofinal.py: possui o código principal;

-Funçõesprojeto.py: armazena as funções e a classe controler;

-Teste.py: Utiliza o melhor controle gerado na última vez que o código foi rodado, usando-o contra todos os chefões em um loop infinito;

Para a posterior análise dos resultados, segue o gráfico das pontuações das gerações no decorrer do tempo.



5. Análise dos Resultados e Conclusão:

Percebe-se pelo gráfico acima que a pontuação obtida cresce com o tempo, principalmente no início, ainda que em alguns momentos a mesma fique estagnada, mostrando que o aprendizado é lento nesses períodos.

Vale ressaltar que como a função fitness é diretamente proporcional ao logaritmo do tempo de sobrevivência do agente, no início este tempo de sobrevivência tem uma influência maior no cálculo da mesma.

Essas grandes quedas que acontecem repentinamente se devem à passagem para outro chefão, pois como o cenário é mudado, é necessário um novo aprendizado.

Como ponto positivo, o robô conseguiu, além de atirar contra os chefões com certa eficiência, adquirir a característica de andar em direção ao chefe, o que possivelmente aumenta as

chances em acertar seus tiros, consegue também, pulando, desviar de muitos ataques de seus adversários.

Porém muitas características adquiridas foram perdidas com o tempo, mostrando que o agente possui uma memória de curto prazo. Também não consegue ter um bom desempenho contra a maioria dos chefes. Por isso, como ponto de melhoria, se propõe a criação de uma rede neural mais complexa, com mais Hidden Layers, possibilitando assim um armazenamento maior de características específicas adquiridas.

6. Referências:

- Miras, Karine. EvoMan - Framework 1.0. August 27, 2019.
- Russell, Stuart J. Artificial Intelligence, A Modern Approach; PRENTICE HALL 2010.