Relatório do Projecto de Inteligência Artificial

110005 Bruno Silva e 110976 Diogo Pereira

Rede neuronal:

Durante a inicialização da rede neuronal, gerámos os valores iniciais dos pesos e bias com valores aleatórios entre -1 e 1, *new Random().nextDouble(-1,1)*. Optámos pela função sigmoide para a camada interna e de output, pois era a que produzia melhores resultados.

A função **nextMove()** que decide qual movimento tomar tem como base devolver o índice do neurónio de output com maior valor, sendo que cada output corresponde a um movimento diferente.

Adicionamos também uma função para calcular a fitness da rede neuronal. Está é calculada com base na média de 3 jogos para sementes diferentes, sendo esses jogos enviados para **threads** diferentes para aumentar a velocidade do cálculo da função fitness. Implementamos ainda a função **getFitness** para devolver o valor do fitness, sendo que é útil para a seleção no algoritmo genético.

Algoritmo Genético:

A função de seleção é uma seleção elitista, em que esta ordena o array da geração anterior e retirar deste os 20 melhores, seguindo estes para a próxima geração.

A função **beginSearch()** é usada para a criar uma nova população. É usada uma percentagem dos melhores indivíduos dessa geração, selecionados na seleção. Outra percentagem de indivíduos resulta do crossover e a restante da mutação. Os indivíduos a serem mutados são os melhores dessa geração, tentando assim criar as melhores soluções para o problema. Se eles melhorarem passa para a geração seguinte, caso contrário são descartados.

Além disso a nossa rede é lhe dada um estímulo em que a seed é mudada de x em x gerações na qual. A rede neuronal tem que se adaptar ao novo jogo. Assim sendo apenas sobreviveram os indivíduos que realmente sabem jogar o jogo.

Inicialmente também treinámos o algoritmo apenas para 3 sementes diferentes, sem haver variação das sementes ao logo da run, o que resultou em resultados bastante altos para estas sementes e tempos de treino menores, mas quando testávamos numa semente diferente a Rede Neuronal não tinha resultados comparáveis. Como achamos esta solução menos correta optámos pelo descrito no paragrafo acima, embora seja mais difícil e demore muito mais gerações a treinar.

Reduzimos a rede neuronal para 25 neurónios no Breakout e 50 no Pacman.

Decidimos realizar uma **Mutação** que escolhe aleatoriamente um gene e soma-lhe um valor aleatório entre -1 e 1.

Embora tenhamos implementado uma função crossover, obtemos melhores resultados sem usar esta, tendo posto a percentagem de seleção a 20% e a de mutação a 80%, que pela nossa implementação a função de crossover não realiza nenhuma operação.

Na função **crossover** presente no código optamos pelo one point crossover por uma questão de simplicidade na implementação.

A nossa função depende um pouco da sorte pois para alguns runs pode atingir um máximo local, não conseguindo sair dele como também poderá atingir um máximo local que consideramos ótimo para jogar atingindo cerca de 1.8 milhões de pontos no Breakout e de 580.000 pontos no Pacman.

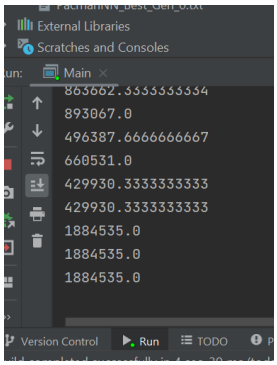

Figura 1- Fitness Breakout

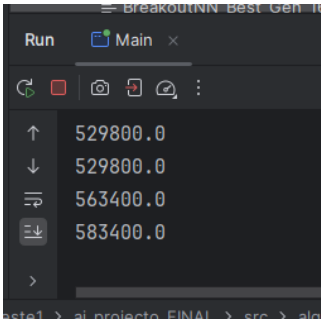

Figura 2- Fitness Pacman

Após testes adicionais carregando redes neuronais geradas anteriormente e testando com seeds diferentes, aferimos que a rede apesar de saber jogar bem seeds mais recentes, esta obtia pontuações baixas nas seeds iniciais, sendo provavelmente jogar este jogo para qualquer semente um problema de dimensão demasiado elevada para este tipo de modelo de aprendizagem.

Adicionalmente:

Durante o desenvolvimento deste projeto testamos diversas implementações e soluções diferentes, entre estas:

- Na tentativa de superar o máximo local, experimentamos aumentar a mutação para valores mais altos durante x gerações;
- Mexemos nos valores de inicialização da rede neuronal;
- Experimentamos varias funções de ativação para a camada interna e output (relu, leakyRelu, tanh(tangente hiperbólica),sigmoid, swish) ;
- Testamos diferentes tipos de mutação: bit flip e bit swap;
- No crossover testamos o k point crossover;
- Testámos uma seleção por torneio com indivíduos aleatórios;
- Usamos a função **best()** neste algoritmo permite escolher a melhor rede neuronal(indivíduo) dessa geração. Sendo este depois usado para realizar uma mutação especial,em que apenas era mutado o melhor indivíduo

Breakout:

- Valores de mutação.
- Testámos também só com uma seed e obtemos resultados na ordem dos 1 900 000 pontos, sendo considerado overfitting, não conseguindo jogar outros jogos.

Pacman:

- Aumentamos o numero de neurónios, testando nos 50, 100, 250 e 800(sendo que a performance descia significativamente nos 800, não sendo prático treinar a rede);
- Trocamos a fitness para tentar recompensar os pontos e penalizar o número de movimentos.