Evoman

Fernando Kenji Ishikawa
Centro de Matemática, Computação e Cognição (CMCC)
Universidade Federal do ABC (UFABC)
Santo André, Brasil
E-mail: fernando.ishikawa@aluno.ufabc.edu.br

Leandro Zangirolami Trovões
Centro de Matemática, Computação e Cognição (CMCC)
Universidade Federal do ABC (UFABC)
Santo André, Brasil
E-mail: leandro.trovoes@aluno.ufabc.edu.br

Leonardo Jose do Carmo
Centro de Matemática, Computação e Cognição (CMCC)
Universidade Federal do ABC (UFABC)
Santo André, Brasil
E-mail: leonardo.carmo@aluno.ufabc.edu.br

***Resumo*—A aplicação de Inteligência Artificial em jogos eletrônicos Este artigo trata a implementação de um agente artificial**

I. Introdução

O EvoMan é um jogo inspirado no Mega Man II, desenvolvido com o propósito de experimentos em aplicações relacionadas à Inteligência Artificial, utilizando a biblioteca de jogos pygame, da linguagem de programação Python. O jogo apresenta o personagem principal e oito adversários diferentes em suas respectivas fases.

O objetivo deste trabalho é criar um agente artificial capaz de vencer o maior número de adversários. Deve-se escolher somente quatro oponentes para o treino do agente de modo que os restantes serão utilizados para o teste do agente final.

Este artigo está organizado da seguinte maneira: Seção II descreve o agente (PEAS); Seção III descreve o algoritmo utilizado; Seção IV apresenta os experimentos e resultados obtidos

II. Descrição Básica do Agente (PEAS)

Agente: Personagem Principal

Performance: Derrotar adversários, Energia do personagem, Energia do adversário, Menor tempo de execução

Environment: Plataformas, ambiente marinho

Actuators: Locomover-se à direita, esquerda, pular e atirar

Sensors: 20 sensores, sendo: 16 para distância do personagem ao projétil, 2 para distância do personagem ao adversário e 2 indicando a direção em que ambos estão direcionados

III. Descrição do Algoritmo Utilizado

A. Redes Neurais

Redes Neurais Artificiais são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. Uma grande rede neural artificial pode ter centenas ou milhares de unidades de processamento. Uma rede neural artificial é composta por várias unidades de processamento, cujo funcionamento é bastante simples. Essas unidades, geralmente são conectadas por canais de comunicação que estão associados a determinado peso. As unidades fazem operações apenas sobre seus dados locais, que são entradas recebidas pelas suas conexões. O comportamento inteligente de uma Rede Neural Artificial vem das interações entre as unidades de processamento da rede. [1]

- Sinais são apresentados à entrada;
- Cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;
- É feita a soma ponderada dos sinais que produz um nível de atividade;
- Aplica-se uma função de ativação ao resultado
- No caso da saída, se este nível de atividade exceder um certo limite (threshold) a unidade produz uma determinada resposta.

B. Algoritmos Genéticos

Toda tarefa de busca e otimização possui vários componentes, entre eles: o espaço de busca, onde são consideradas todas as possibilidades de solução de um determinado problema e a função de avaliação (ou função de custo), uma maneira de avaliar os membros do espaço de busca. Existem muitos métodos de busca e funções de avaliação. As técnicas de busca e otimização tradicionais iniciam-se com um único candidato que, iterativamente, é manipulado utilizando algumas heurísticas (estáticas) diretamente associadas ao problema a ser solucionado. Geralmente, estes processos heurísticos não são algorítmicos e sua simulação em computadores pode ser muito complexa. Apesar destes métodos não serem suficientemente robustos, isto não implica que eles sejam inúteis. Na prática, eles são amplamente utilizados, com sucesso, em inúmeras aplicações. Por outro lado, as técnicas de computação evolucionária operam sobre uma população de candidatos em paralelo. Assim, elas podem fazer a busca em diferentes áreas do espaço de solução, alocando um número de membros apropriado para a busca em várias regiões. [2]

Os Algoritmos Genéticos (AGs) diferem dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos:

1. AGs trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros.

2. AGs trabalham com uma população e não com um único ponto.

3. AGs utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar.

4. AGs utilizam regras de transição probabilísticas e não determinísticas.

C. Neuroevolução

Na neuroevolucãao, em vez de usar algoritmos convencionais de aprendizado, como Retropropagação em modo supervisionado, para treinar uma rede neural, as caracteríısticas das redes neurais como pesos e topologia podem ser codificadas como genomas artificiais que serão evoluídos e selecionados de acordo com sua performance em achar as solucções esperadas. [3]

D. Configuração da rede

Foram utilizados um total de 14 sensores como entrada da rede:

- Distância no eixo x e no eixo y para o inimigo (total de 2 sensores)
- A indicação de se o personagem principal está apontando na direção do inimigo (total de 1 sensor)
- A indicação da direção que o inimigo está apontando (total de 1 sensor)
- Distância no eixo x e no eixo y para os 5 projéteis mais próximos (total de 10 sensores)

Em todas as distâncias dos sensores foram aplicadas uma função de maneira à normalizar os valores entre -1 e 1, além de atribuir maior peso às mudanças de valores caso estas sejam próximas ao personagem.

A saída da rede consiste em 5 opções não-exclusivas correspondentes às ações a serem tomadas:

- Cima
- Baixo
- Atirar
- Pular
- Soltar

Estas saídas foram determinadas com a utilização da função de ativação sigmoide e eram realizadas caso fossem superiores ao ponto de corte, que foi determinado como sendo 0.5.

Foram utilizadas duas camadas intermediárias densas com respectivamente 32 e 12 neurônios em que eram aplicadas a função de ativação relu.

E. Aplicação da neuroevolução neste projeto

A solução encontrada para resolver o problema consiste na utilização de um algoritmo neuroevolutivo para determinar os pesos a serem utilizados em uma rede neural de pesos pré-determinados. Desta forma, os indivíduos utilizados no algoritmo genético consistiam do conjunto dos pesos entre cada par de neurônios da rede. Para cada indivíduo inicial foram atribuídos pesos de uma distribuição uniforme de valores entre -1 e 1.

Em cada geração foi realizada a recombinação dos indivíduos da geração anterior com o auxílio da heurística 'Estratégia Evolutiva', criada inicialmente para trabalhar com problemas cujos estados são representados por valores contínuos. Desta maneira, os pesos de cada par de indivíduos eram multiplicados por um fator complementar entre 0 e 1 e somados.

Em cada geração, todos os indivíduos sofriam diversas mutações em que eram somados valores de uma distribuição gaussiana à cada peso deste. Dentre os indivíduos criados e o original, apenas o melhor foi mantido. O fator de mutação, que determina o desvio padrão da gaussiana, era diminuído a cada geração na tentativa de beneficiar a exploração nas iterações iniciais e permitir a explotação adequada ao longo do tempo.

Após a mutação dos indivíduos da geração anterior e resultantes da recombinação destes, é realizada a seleção dos melhores indivíduos de maneira à sempre existir o mesmo número de indivíduos em cada geração e manter apenas indivíduos promissores.

Após um conjunto de iterações serem realizadas, parte dos indivíduos são descartados para a introdução de indivíduos completamente novos de maneira a explorar regiões novas do espaço de busca.

F. Fitness

Foram utilizadas duas funções para avaliar o fitness. Uma é dada pela média do fitness subtraída do desvio padrão. Já a outra leva em consideração a soma dos fitness.

G. Normalização

Para a normalização das entradas, foram utilizadas duas equações, uma dada pela biblioteca numpy, que trata arrays na linguagem Python, e a outra modelada pelos autores dada a equação

$$a^{-(x/b)^c}$$

IV. Experimentos e resultados obtidos

Diversos experimentos foram realizados, classificados na seguinte ordem: quantidade de projétil, função de ativação, tamanho da camada intermediária (camada1/camada2) em neurônios, se o fitness calculado é dado pela equação do próprio jogo (game-fit) ou pela equação proposta pelos autores (our-fit), se a função para normalizar as entradas é a minmax ou não (minmax) ou proposta pelos autores (our-norm) e as fases de treinamento.

1) 3, Tanh, 8, our-fit, minmax, (1,3,6,7).
2) 3, Tanh, 8/5, our-fit, minmax, (1,3,6,7).
3) 3, Tanh, 20/10, our-fit, minmax, (1,3,6,7).
4) 3, Relu, 20/10, our-fit, minmax, (1,3,6,7).
5) 4, Relu, 20/10, our-fit, our-norm, (1,3,6,7).
6) 4, Relu, 25/12, game-fit, our-norm, (1,3,6,7).
7) 4, Relu, 25/12, our-fit, our-norm, (1,3,4,7).
8) 5, Relu, 20/10, our-fit, our-norm, (1,3,6,7).
9) 5, Relu, 28/10, our-fit, our-norm, (1,3,6,7).
10) 5, Relu, 32/12, our-fit, our-norm, (1,3,6,7).

Para as saídas foi estipulada a função de ativação sigmoide, com 5 neurônios.

Resultados:

1) GENERATION: 83
   BEST FITNESS: 91.70195646704545
2) GENERATION: 72
   BEST FITNESS: 100.31068628909998
3) GENERATION: 123
   BEST FITNESS: 100.41354942151821
4) GENERATION: 129
   BEST FITNESS: 60.05776494197335
5) GENERATION: 129
   BEST FITNESS: 107.97885392065825
6) GENERATION: 99
   BEST FITNESS: 118.68644486573218
7) GENERATION: 99
   BEST FITNESS: 89.10633302278589
8) GENERATION: 47
   BEST FITNESS: 67.18923345949815
9) GENERATION: 84
   BEST FITNESS: 104.8396473482006
10) GENERATION: 149
    BEST FITNESS: 117.97882177051068

O décimo agente obteve o maior desempenho, apresentado a seguir:

Fase 1: 94.02326625757942
Fase 2: 86.85873645484158
Fase 3: 86.93157441175589
Fase 4: 21.401578041001624
Fase 5: 92.42226955283408
Fase 6: 86.38896154643432
Fase 7: 75.18586946817493
Fase 8: 65.75195712549157
Média em fases de treino: 85.632417921
Média em fases de teste: 66.608635294
Média em todas as fases: 76.120526607

Além disso foi capaz de derrotar cinco dos oito adversários, em que 2 destes pertenciam ao grupo de teste.

Análise dos resultados e conclusão

A utilização de relu ou tangente hiperbólica nas camadas intermediárias pouco influenciou nos resultados, com a relu apresentando resultado final pouco superior. Em experimentos posteriores permanece válido o teste com ambas para a comparação do desempenho.

A utilização de softmax nas saídas da rede para realizar uma ação única a cada instante se demonstrou ineficienteA utilização de tangente hiperbólica na saída de maneira a excluir ações opostas (saltar e soltar ou esquerda e direita) também foi ineficiente nos treinos. Dessa maneira acreditamos que a melhor alternativa é utilizar sigmoide.

Ao utilizar a soma dos fitness individuais em cada fase de treino, percebemos que o agente tendia a se especializar nas fases mais fáceis rapidamente e apresentava dificuldade em melhorar nas demais fasesAo utilizar a média dos fitness menos o desvio padrão os resultados apresentavam fitness próximos e demonstrou ser melhor que a primeira função utilizada, porém após determinado ponto não apresentava mais melhorias e não conseguiam derrotar nenhum adversário. A solução encontrada foi utilizar o menor dos valores mais a metade da média, dessa maneira conseguimos evitar a especialização em um único indivíduo enquanto fornecíamos flexibilidade para a melhoria dos indivíduos.

Devido aos dados dos sensores possuírem escalas distintas (-600 a 600 para a distância ao oponente e -1 à 1 para a direção deles) foi necessária a normalização dos dados. Inicialmente foi utilizada uma escala para realizá-la que apresentou desempenho satisfatório. Posteriormente, criamos uma função própria com o intuito de aumentar a importância dos dados mais próximos ao personagem. Esta função se demonstrou capaz de melhorar os resultados obtidos

Diversos resultados foram capazes de apresentar desempenho satisfatório nos oponentes de treino. Contudo, ao analisar os resultados obtidos no teste foi possível perceber um resultado insatisfatório. Desta forma, estes resultados apenas foram obtidos com o overfitting dos dados e foram desconsiderados.

Referências

[1] http://conteudo.icmc.usp.br/pessoas/andre/research/neural/
[2] http://conteudo.icmc.usp.br/pessoas/andre/research/genetic/
[3] https://pdfs.semanticscholar.org/b351/e208d4c0a7b29ca86e20e410b168916c370d.pdf