Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Lee Moraes

Aprendizado por Reforço com Algoritmos Genéticos aplicado a Jogos

Monografia apresentada ao Departamento de Engenharia Elétrica da PUC-Rio como requisito parcial para obtenção do título de Especialização em *Business Intelligence*.

Orientador: Prof. Leonardo Alfredo Forero Mendoza

Rio de Janeiro
28 de Março de 2019

RESUMO

Neste trabalho utilizamos algoritmos genéticos para otimizar o desempenho de uma rede neural, esta foi usada para controlar o jogador no jogo Pong. O algoritmo genético foi usado para cruzar e modificar os pesos de cada neurônio da rede, que nesse caso são os genes do algoritmo evolucionário. O estado do jogo que usamos como entrada na rede neural são as posições da barra do jogador e da bola. Existem várias aplicações para algoritmos evolutivos, por exemplo o aprendizado de carros autônomos e controle de máquinas autônomas. Nossos experimentos mostram que a rede neural consegue aprender a jogar e vencer facilmente seu adversário somente com cruzamento e mutação de seus genes, sem a necessidade do uso de um gradiente descendente para otimização. Apesar do sucesso desse algoritmo, o comum é não termos as medidas do estado do jogo, e sim apenas a imagem, então apontamos como estudo futuro a otimização de uma rede convolucional com dados de imagem para jogar este jogo.

ABSTRACT

In this work we used genetic algorithms to optimize the performance of a neural network, this was used to control the player in the game Pong. The genetic algorithm was used to cross and mutate the weights of each neuron of the network, which in this case are the genes of the evolutionary algorithm. The state of the game we use as input into the neural network is the positions of the player's bar and ball. There are several applications for evolutionary algorithms, for example the learning of driving in autonomous cars and control of autonomous machines. Our experiments show that the neural network can learn to play and easily defeat its adversary only by crossing and mutating its genes without the use of a descending gradient for optimization. Despite the success of this algorithm, it is common not to have the measures of the state of the game, but only the image, so we point as future study the optimization of a convolutional network with image data to play this game.

Sumário

Lista de figuras

Lista de tabelas

Lista de Abreviaturas

RL – *Reinforcement Learning*
CNN – Redes Neurais Convolucionais (*Convolutional Neural Network*)
RNN – Redes Neurais Recorrentes (*Recurrent Neural Network*)
MDP – Processo de Decisão de Markov (*Markov Decision Process*)

1. Introdução

Com o crescente uso de automações nos mais diversos campos profissionais, entramos na era da robotização e da inteligência artificial. Trabalhos de baixa complexidade e repetitivos estão sendo completamente substituídos por máquinas e computadores, diminuindo a fadiga dos trabalhadores e evitando erros humanos.

A utilização da automação de processos por meio de aprendizado por reforço (RL) abrange muitos campos. Podemos usá-lo, por exemplo, no gerenciamento de recursos de clusters de computadores (1), no controle por um sistema multi-agente para uma rede de semáforos (2) e, principalmente, em robótica e jogos, em que treinam-se robôs para realizar ações a partir do processamento de uma imagem/video.

No caso deste trabalho, utilizaremos algoritmos genéticos para a otimização de uma rede neural para aprendizado por reforço do jogo Pong. Apesar de não ser uma aplicação voltada a resolução de problemas de negócios, ele nos mostra que podemos resolver problemas de otimização de controladores utilizando aprendizado por reforço, além de ser um trabalho divertido de se fazer.

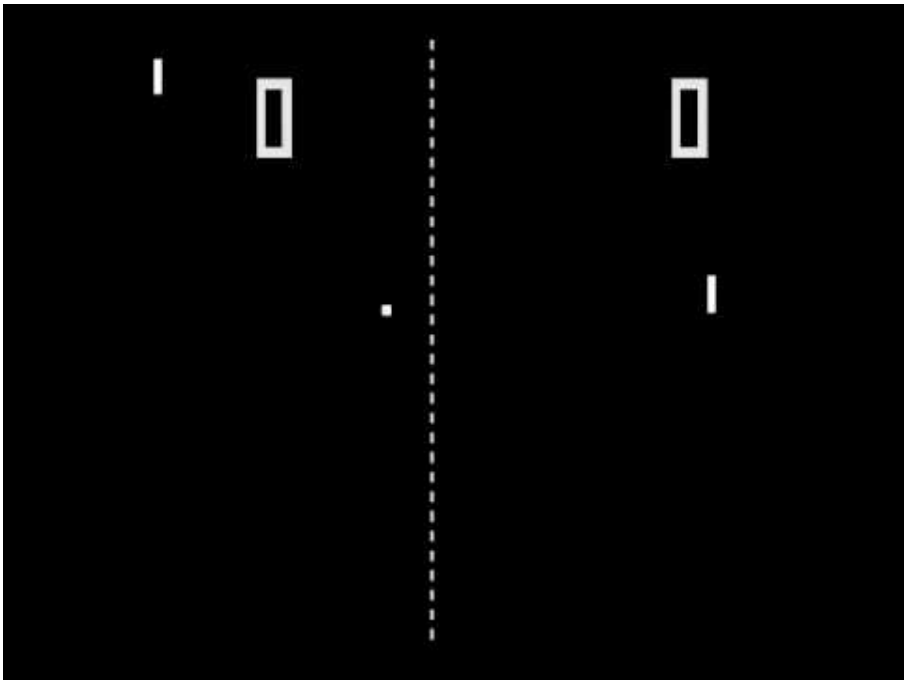

Figura 1.1: Jogo Pong do Atari2600

2. Trabalhos Correlatos

Existem diversas pesquisas em que foram desenvolvidos algoritmos de aprendizado por reforço para jogar jogos com habilidade sobrehumana, sendo alguns dos casos mais famosos o AlphaGo (3) e o AlphaGo Zero (4). O AlphaGo, primeiro a ser criado, foi alimentado com inúmeros jogos humanos previamente gravados e tinha performance superior à humana usando um mecanismo de busca em árvore. Já o AlphaGo Zero foi treinado com um aproximação mais pura de aprendizado por reforço, aprendendo do zero sem conhecimento humano. Ele jogava contra si mesmo para treinar e, após seu aprendizado, derrotou seu antecessor por 100-0.

As tentativas mais atuais de aprendizado por reforço tem utilizado algoritmos mais robustos para resolver problemas cada vez mais complexos, como a utilização de redes neurais convolucionais (CNN) e/ou redes neurais recorrentes (RNN) combinadas com RL, algoritmos do tipo Policy Gradient e etc. Temos como exemplo o uso de LSTM em rede recorrente juntamente com RL para a criação da Deep Recurrent Q-Network (DRQN) para jogar jogos de Atari2600 (5) .

Pode-se perceber que esses trabalhos citados são de elevada complexidade, o que pode levar a uma grande demora para treinar suas redes. Devido a isso, utilizaremos o estado do jogo não como imagem, mas sim as posições dos objetos.

Além disso, alguns estudos apontam que o uso de algoritmos genéticos para aprendizado por reforço é uma alternativa competitiva à otimização por algoritmos baseados em retropropagação (6), dado que os dados realmente úteis para o treino são esparsos, já que às vezes são necessárias muitas ações antes de se receber uma recompensa.

3. Aprendizado de Máquina

Aprendizado de máquina (ML) é o estudo científico de algoritmos e modelos estatísticos que os sistemas de computador usam para realizar uma tarefa específica sem usar instruções explícitas, confiando em padrões e inferência (7). É visto como um subconjunto da inteligência artificial. Algoritmos de aprendizado de máquina constroem um modelo matemático de dados de amostra, conhecido como dados de treinamento, para fazer predições ou decisões sem ser explicitamente programado para realizar a tarefa. Algoritmos de aprendizado de máquina são usados em uma ampla gama e variedade de aplicativos, como filtragem de e-mail (8) e visão computacional (9), onde é inviável desenvolver um algoritmo de instruções específicas para executar a tarefa. O aprendizado de máquina está intimamente relacionado à estatística computacional, que se concentra em fazer previsões usando computadores.

3.1 Tipos de Algoritmos de Aprendizado

Os tipos de algoritmos de aprendizado de máquina diferem em sua abordagem, o tipo de dados que eles inserem e emitem e o tipo de tarefa ou problema que eles pretendem resolver. São eles aprendizado supervisionado/semi-supervisionado, não supervisionado e por reforço.

3.1.1 Aprendizado Supervisionado

Algoritmos de aprendizado supervisionado constroem um modelo matemático a partir de um conjunto de dados que contém tanto as entradas quanto as saídas desejadas (10). Os dados são conhecidos como dados de treinamento e consistem em um conjunto de exemplos de treinamento. Cada exemplo de treinamento possui uma ou mais entradas e uma saída desejada, também conhecida como sinal de supervisão. No caso de algoritmos de aprendizado semi-supervisionados, alguns dos exemplos de treinamento não possuem o dado rotulado. No modelo matemático, cada exemplo de treinamento é representado por uma matriz ou vetor e os dados de treinamento por uma matriz. Por meio da otimização iterativa de uma função objetivo, os algoritmos de aprendizado supervisionado aprendem uma função que pode ser usada para prever a saída associada a novos insumos. Uma função ideal permitirá que o algoritmo determine corretamente a saída para entradas que não fazem parte dos dados de

treinamento. Um algoritmo que melhora a precisão de suas saídas ou previsões ao longo do tempo é dito ter aprendido a executar essa tarefa.

3.1.2 Aprendizado Não Supervisionado

Os algoritmos de aprendizado não supervisionados usam um conjunto de dados que contém apenas entradas e localizam estrutura nos dados, como agrupamento ou agrupamento de pontos de dados (11). Os algoritmos, portanto, aprendem com dados de teste que não foram rotulados, classificados ou categorizados. Em vez de responder ao feedback, os algoritmos de aprendizado não supervisionados identificam pontos comuns nos dados e reagem com base na presença ou ausência de tais pontos comuns em cada novo dado. Uma aplicação central da aprendizagem não supervisionada está no campo da estimativa de densidade na estatística, embora a aprendizagem não supervisionada englobe outros domínios envolvendo resumir e explicar as características dos dados.

3.1.3 Aprendizado por Reforço

Aprendizado por reforço é uma área de aprendizado de máquina preocupada com a forma como os agentes de software devem tomar ações em um ambiente, de modo a maximizar alguma noção de recompensa cumulativa (12). Devido à sua generalidade, o campo é estudado em muitas outras disciplinas, como teoria dos jogos, teoria de controle, pesquisa operacional, teoria da informação, otimização baseada em simulação, sistemas multiagentes, inteligência de enxames, estatística e algoritmos genéticos. No aprendizado de máquina, o ambiente é tipicamente representado como um Processo de Decisão de Markov (MDP) (13). Muitos algoritmos de aprendizado por reforço usam técnicas de programação dinâmica. Algoritmos de aprendizado por reforço não assumem o conhecimento de um modelo matemático exato do MDP, e são usados quando modelos exatos são inviáveis. Algoritmos de aprendizagem de reforço são usados em veículos autônomos ou em aprender a jogar um jogo contra um oponente humano.

4. Aprendizado por Reforço

Aprendizado por Reforço, como vimos no capítulo anterior, é um método de aprendizado de máquina em que um agente aprende a realizar ações em um ambiente que levam a maior recompensa possível. Ele aprende através da exploração do ambiente e das possíveis ações em cada estado após inúmeras iterações, com o objetivo de maximizar a recompensa de suas ações. É necessário termos a terminologia bem definida, portanto listo-os abaixo (14):

- **Agente (Agent)**: O modelo de decisão que realiza uma ação em um ambiente para ganhar uma recompensa;
- **Ação (Action)**: As possíveis ações que o agente pode realizar;
- **Ambiente (Environment)**: O cenário em que o agente deve enfrentar;
- **Estado (State)**: A situação corrente retornada pelo ambiente;
- **Recompensa (Reward**: O retorno imediato dado pelo ambiente para avaliar a última ação feita pelo agente;
- **Política (Policy)**: A estratégia que o agente emprega para determinar a próxima ação baseada no estado corrente;
- **Valor (Value)**: O retorno esperado de longo prazo com desconto, em oposição à recompensa de curto prazo;
- **Q-Valor (Q-Value)**: O valor Q é semelhante ao Valor, exceto pelo fato de que é necessário um parâmetro extra, a ação atual a.

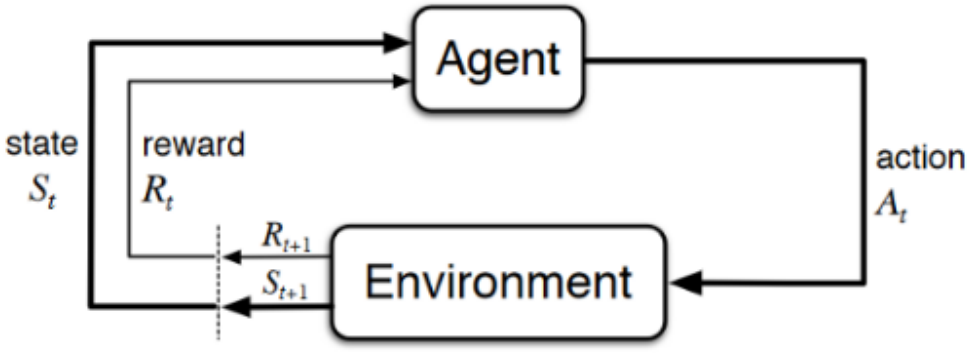

Figura 4.1: Modelo de Aprendizado por Reforço

A RL é única no uso da combinação de quatro máquinas distintas: aproximação estocástica, programação dinâmica, inteligência artificial e aproximação de função. Ao explorar esses mecanismos, a RL abriu o caminho para a solução de Processos de Decisão de Markov (MDP) de grande escala (e suas variantes) em sistemas de eventos discretos, considerados intratáveis no passado (15)u a heurística industrial em vários dos estudos de caso publicados de importância industrial. RL como uma ciência é relativamente jovem e já causou um impacto considerável na pesquisa operacional. Embora um grande trabalho no desenvolvimento algorítmico de aprendizado por reforço já tenha ocorrido, a RL continua a atrair a atenção da pesquisa.

5. Algoritmos Genéticos

Os algoritmos genéticos são heurísticas inspiradas pela teoria da evolução natural de Charles Darwin. Esses algoritmos imitam o processo de seleção natural em que os mais aptos são selecionados para passar seus genes para as próximas gerações (16). O processo de seleção de nosso algoritmo começa com a seleção dos indivíduos mais aptos, no caso de nosso jogo, os agentes mais capazes de rebater as bolinhas e fazer pontos contra o adversário. Os selecionados produzem descendentes que herdam as características de seus progenitores para a próxima geração. Se os progenitores tem melhor aptidão, seus descendentes serão melhores e terão maior chance de sobreviver. Esse processo é iterado por várias gerações e no final, os indivíduos com melhor aptidão serão encontrados.

Podemos considerar 5 etapas para o algoritmo genético:

5.1 População Inicial

O processo começa com um grupo de indivíduos chamado População. Cada indivíduo desses é uma solução para o problema que queremos resolver. Cada indivíduo é caracterizado por parâmetros conhecidos como Genes que juntos formam o Cromossomo (o própria modelo de agente). Neste trabalho, o agente do algoritmo de aprendizado por reforço é uma rede neural cujos pesos são os genes.

5.2 Função de Aptidão (*Fitness Function*)

A Fitness Function determina o quão apto um indivíduo é para realizar a tarefa proposta. Ao longo do treinamento, cada indivíduo consegue uma pontuação de aptidão (Fitness Score). A probabilidade do indivíduo ser escolhido para reprodução é baseada nesta pontuação.

5.3 Seleção

Na seleção, escolhemos os indivíduos mais aptos e deixamos eles passarem para as próximas gerações. Pares de indivíduos são escolhidos com base em seu Fitness Score, quanto maior sua pontuação maior a probabilidade de serem escolhidos.

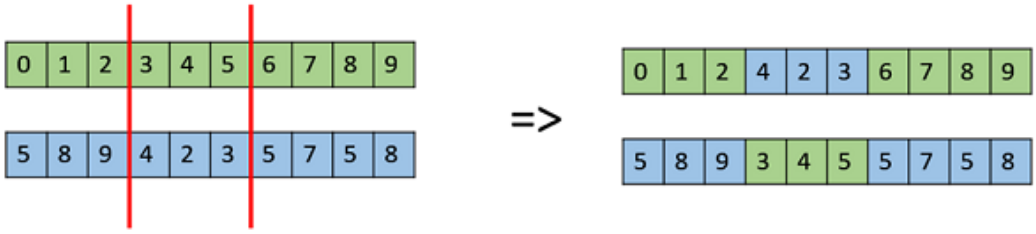

Figura 5.1: Cruzamento

5.3.1 Elitismo

Usando o Elitismo, passa-se também para as gerações seguintes os indivíduos com melhores Fitness Score, a fim de guardar os agentes com melhor resultado e forçar a geração de indivíduos com sua genética (17). Este tipo de seleção foi usada em nosso trabalho para acelerar a convergência na solução ótima.

5.4 Cruzamento

O Cruzamento genético é a parte mais importante deste algoritmo. Nesta etapa, parte dos genes de um indivíduo são trocados com a mesma parte de outro indivíduo, criando assim dois novos indivíduos para a próxima geração, como podemos ver na figura .

5.5 Mutação

Em alguns dos novos indivíduos gerados após o cruzamento, realizamos mutações. Definimos uma probabilidade pequena de realizar mutações para introduzir aleatoriedade às novas gerações. Desse modo tentamos criar descendentes que saiam de máximos locais para encontrar máximos globais.

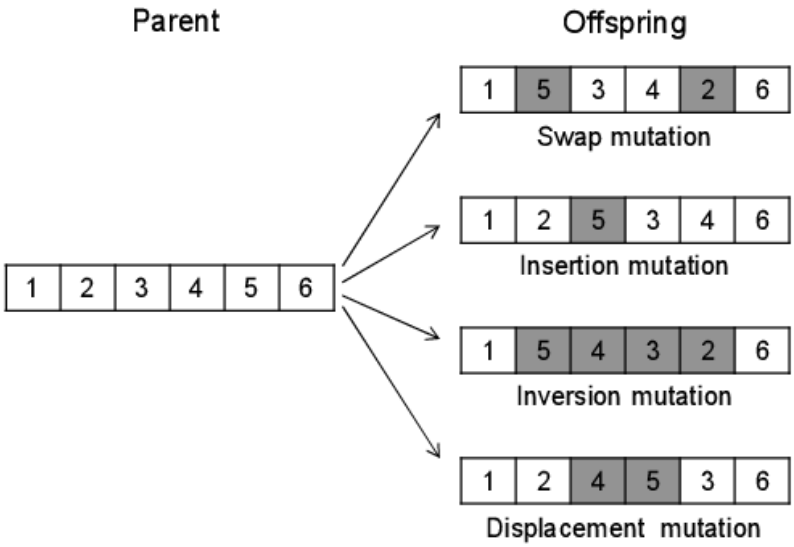

Figura 5.2: Tipos de Mutação

6. Redes Neurais

Redes Neurais são um tipo de algoritmo que tenta imitar o funcionamento do cérebro humano e é usado para identificação de padrões (18). Elas são capazes de interpretar dados como imagens, textos e outros atrasés de percepção de máquina, rotulação e agrupamento de uma entrada. A entrada dos dados na rede deve ser numérica, então dados de imagem, som, texto ou séries temporais deve ser traduzidas em vetores numéricos.

O objetivo das redes neurais é fazer a classificação, regressão e/ou agrupamento necessárias para a resolução do problema escolhido. Assim como outros mecanismos de aprendizado de máquina, a rede neural é treinada com dados rotulados a fim de encontrar padrões que ajudem a classificar novas entradas não rotuladas.

A rede neural, do tipo feedforward como em nosso trabalho, é formada por camadas de neurônios, cada um consistindo de uma função que tem a camada anterior como entrada e saída para todos os neurônios da camada seguinte. Cada neurônio possui pesos para cada uma dessas entradas e um viés, resultando em um número que é a entrada para os próximos neurônios. Ao longo da rede, a entrada é processada e no final temos o resultado da rede, que pode ser uma classificação ou uma regressão.

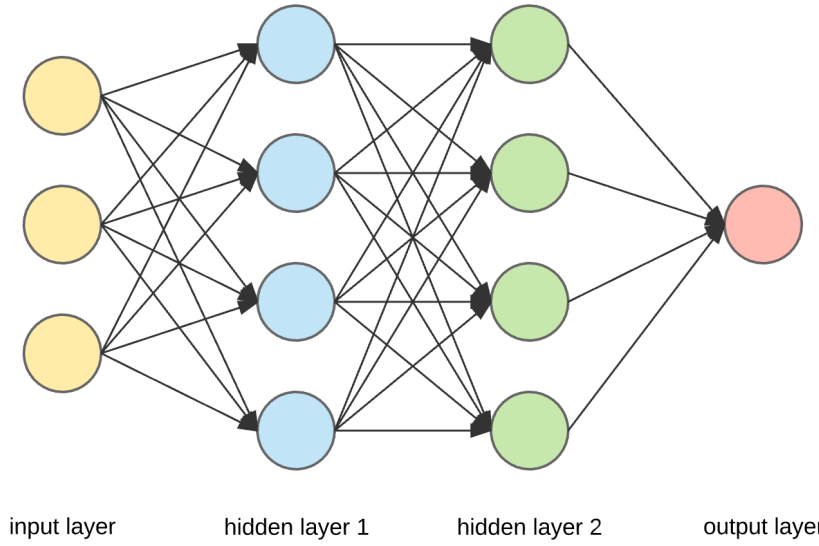

Figura 6.1: Rede Neural

6.1 Neurônios

O neurônio é a unidade mais elementar da rede neural. Ele recebe uma ou mais entradas (como sinais sinápticos nos dendritos do neurônio de nossos cérebros), multiplica cada entrada por diferentes pesos e somam tudo com um valor de viés para produzir a saída (19). Essa saída passa por uma função de ativação, que cria a não-linearidade da rede neural, e passa para a próxima camada da rede.

Temos a seguir a função que representa o neurônio:

$$y_k = \varphi\left(\sum_{0}^{m} w_{kj} x_j\right)$$

Sendo φ a função de ativação, w_0 até w_m os pesos para cada entrada mais um para viés, $x_0 = 1$, logo viés $w_{k0} = b_k$.

6.1.1 Perceptron

No aprendizado de máquina, o perceptron é um algoritmo para aprendizado supervisionado de classificadores binários (20). Um classificador binário é uma função que pode decidir se uma entrada, representada por um vetor de números, pertence ou não a uma classe específica. É um tipo de classificador linear, ou seja, um algoritmo de classificação que faz suas previsões com base em uma função preditora linear combinando um conjunto de pesos com o vetor de recursos.

6.2 Multilayer Perceptron

Um perceptron multicamada (MLP) é uma classe de rede neural artificial feedforward. Um MLP consiste em, pelo menos, três camadas de nós: uma camada de entrada, uma camada escondida e uma camada de saída. Exceto pelos nós de entrada, cada nó é um neurônio que usa uma função de ativação não-linear. O MLP utiliza uma técnica de aprendizagem supervisionada chamada backpropagation para treinamento (21). Suas múltiplas camadas e ativação não-linear distinguem o MLP de um perceptron linear. Pode distinguir dados que não são linearmente separáveis.

7. Modelagem

O jogo Pong utilizado neste trabalho é uma versão modificada do "Very simple Pong game"encontrado no site da biblioteca PyGame (http://www.pygame.org/project-Very+simple+Pong+game-816-.html), utilizada para fazer o jogo. Foram introduzidas algumas aleatoriedades nas rebatidas da bola para o jogo não prender em alguma posição e possivelmente travar o treinamento. O oponente tem uma inteligência simples para rebater a bolinha, retornando-a de volta na maioria das vezes, mas não todas. Acoplamos então ao jogo o modelo de aprendizado para ensinar o agente a jogar Pong.

Em nosso trabalho, o modelo de aprendizado por reforço usa como agente uma rede neural. Essa rede recebe como entrada 5 variáveis do jogo, tem 6 neurônios na camada escondida e apenas uma saída, que nos dá um valor de 0 até 1.

Os 5 valores de entrada na rede são:

- Posição no Eixo X da Bola;
- Posição no Eixo Y da Bola;
- Posição no Eixo X da Barra;
- Posição no Eixo X da Bola no Último Estado;
- Posição no Eixo Y da Bola no Último Estado.

Os 6 diferentes pesos da camada escondida da rede neural são atualizados pelo algoritmo genético - e não o tradicional backpropagation - a fim de maximizar a recompensa. A Fitness Function do nosso algoritmo recompensa o agente da seguinte forma:

- **Rebate a Bola**: 1 ponto;
- **Pontua contra o Oponente**: 5 pontos;

Ao longo de cada partida até o primeiro jogador fazer 20 pontos, a recompensa vai sendo somada e usamos o valor final para definir os melhores indivíduos.

Foi escolhido arbitrariamente o valor de 16 indivíduos por geração de nosso algoritmo. A população da primeira geração é formada por redes neurais inicializadas com pesos aleatórios. As gerações seguintes serão formadas por cruzamentos e mutações dos 3 melhores indivíduos da geração anterior, usando Elitismo para guardar os três melhores indivíduos para a geração seguinte.

O modelo faz o cruzamento entre os 3 melhores indivíduos da geração anterior, criando dois descendentes para cada par. Dois descendentes provenientes do cruzamento do Campeão e Vice-Campeão, mais dois do cruzamento do Campeão com o Terceiro Lugar e outros dois do par Vice-Campeão. Com esses 6 descendentes, criamos novos 6 fazendo uma mutação pequena e aleatória em todos os pesos. Finalmente, criamos um indivíduo novo com pesos aleatórios para inserir mais aleatoriedade ao modelo, totalizando 16 indivíduos para a nova geração.

A saída da rede representa as 3 possíveis ações do agente:

- **Mover-se para Baixo**: de 0 a 0,44;
- **Ficar Parado**: de 0,44 a 0,55;
- **Mover-se para Cima**: de 0,55 a 1.

Ao longo das gerações o modelo deve convergir para alguns indivíduos que ganham do oponente e consigam rebater bem as bolinhas.

8. Resultados

O modelo foi treinado até a geração 71, a geração imediatamente após a primeira geração a ganhar todos os jogos. Podemos não apenas ver nos gráficos deste capítulo a evolução da recompensa, da pontuação e do número de vitórias ao longo das geerações, como também a diminuição dos pontos feitos pela barra adversária.

O algoritmo genético construído foi capaz de aprender a rebater e fazer pontos contra o oponente em pouco menos de 20 horas de jogo, treinando uma rede neural em uma CPU. No entanto, o jogo não foi acelerado ou paralelizado para o treinamento ocorrer mais rápido.

Nos gráficos abaixo podemos observar as curvas formadas pelas médias móveis de 5 e 10 dias. Elas foram inseridas nas imagens para nos ajudar a perceber melhor a evolução do modelo, os gráficos ficaram com muita variação já que há uma aleatoriedade no jogo que pode causar picos de performance em uma geração ou outra. Acontece que, por um erro do jogo, a barra adversária não consegue pegar a bolinha em um ponto específico do campo e nós pontuamos, porém a bolinha reaparece no meio do campo com a mesma trajetória da anterior para o mesmo ponto específico e pontua novamente, e depois de novo, causando pontos a nosso favor até acabar a partida de 20 pontos.

Foram inseridos também neste capítulo as tabelas com resultados das gerações 0, 30 e 70. Comparando os resultados das três gerações, vemos que a Geração 0 faz muitos pontos mas com baixa recompensa. Isso acontece devido ao erro comentado no parágrafo anterior, o adversário não rebate mais nenhuma bola e ganhamos o jogo sem encostar mais na bola. No caso das gerações posteriores, vemos que as recompensas são bem maiores para um mesmo número de pontos do jogador, significando que houveram muito mais rebatidas de bola do nosso agente.

Análise de Gerações			
Geração 0			
Indivíduo	Reward	Jogador	Oponente
1	3	0	20
2	112	20	11
3	17	0	20
4	62	7	20
5	6	0	20
6	81	10	20
7	45	4	20
8	0	0	20
9	102	20	1
10	18	3	20
11	7	0	20
12	20	1	20
13	121	20	6
14	11	0	20
15	0	0	20
16	107	20	9

Tabela 8.1: Análise da Geração 0

Análise de Gerações			
Geração 30			
Indivíduo	Reward	Jogador	Oponente
1	169	20	0
2	144	20	11
3	120	20	0
4	26	0	20
5	122	20	6
6	184	20	0
7	174	20	0
8	128	20	0
9	189	20	0
10	110	20	7
11	113	20	5
12	134	20	0
13	154	20	0
14	187	20	0
15	182	20	0
16	33	0	20

Tabela 8.2: Análise da Geração 30

Análise de Gerações			
Geração 70			
Indivíduo	Reward	Jogador	Oponente
1	191	20	0
2	122	20	0
3	165	20	0
4	157	20	0
5	143	20	0
6	167	20	0
7	196	20	0
8	126	20	0
9	173	20	0
10	223	20	0
11	158	20	0
12	120	20	0
13	120	20	0
14	158	20	0
15	147	20	0
16	115	20	15

Tabela 8.3: Análise da Geração 70

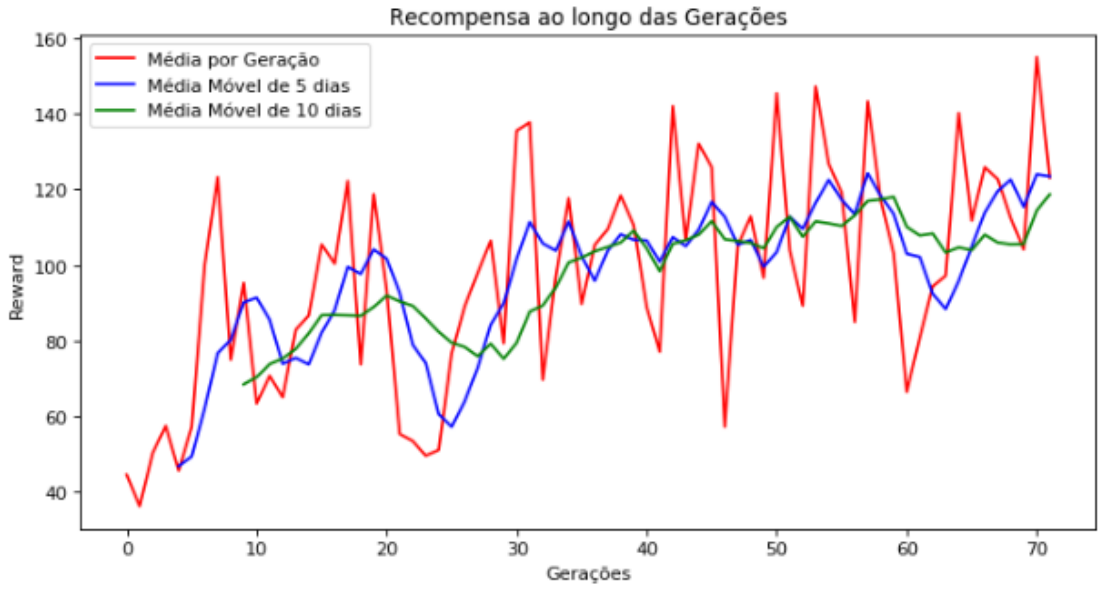

Figura 8.1: Recompensa ao longo das Gerações

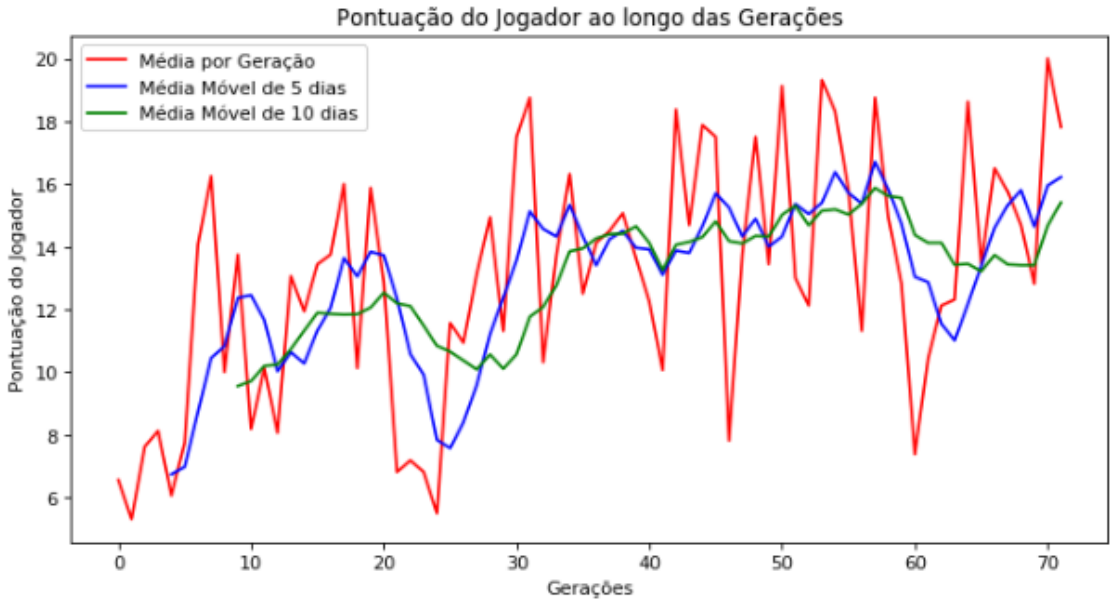

Figura 8.2: Pontuação do Jogador ao longo das Gerações

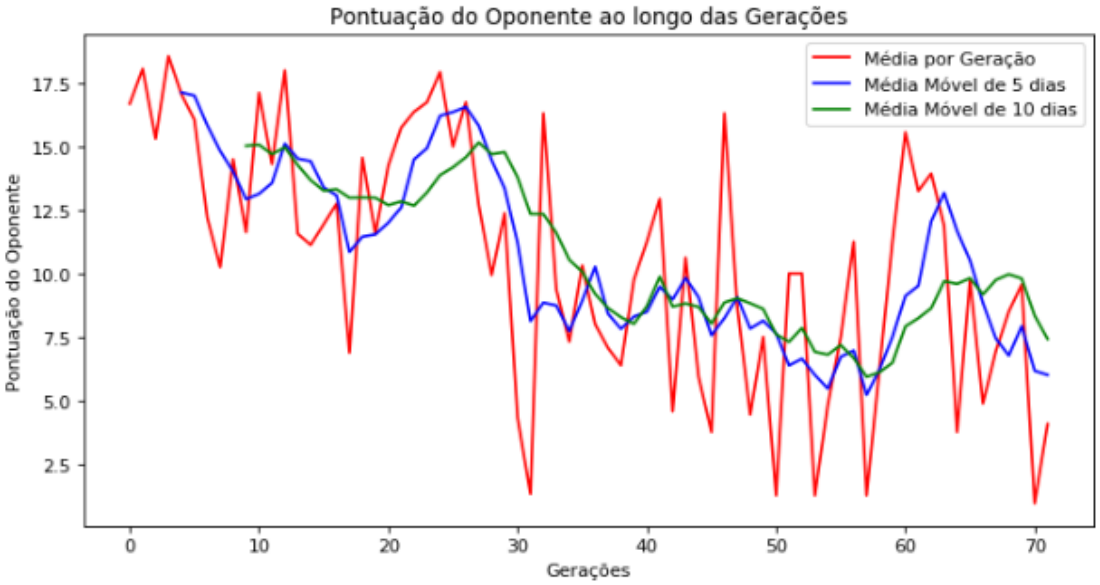

Figura 8.3: Pontuação do Oponente ao longo das Gerações

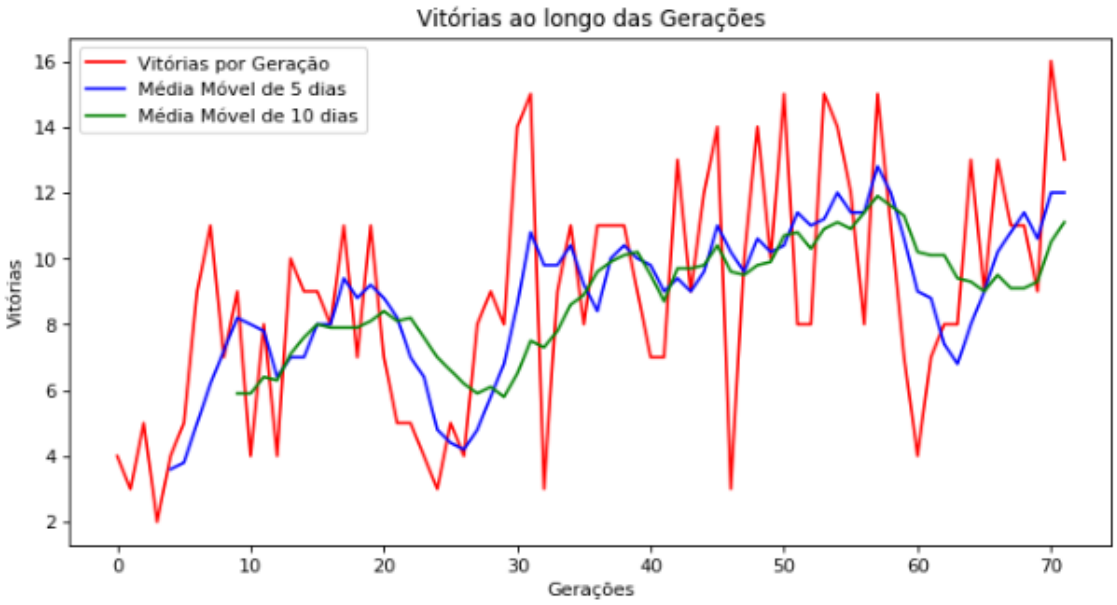

Figura 8.4: Vitórias ao longo das Gerações

9. Conclusões e Trabalhos Futuros

O modelo evolucional com algoritmos genéeticos obteve sucesso em seu treinamento no jogo Pong. Os agentes foram capazes de ganhar os jogos sem o oponente fazer pontos, um Perfect Score de vinte a zero, sem ocorrer o erro do jogo.

Apesar da simplicidade do modelo escolhido, se compararmos aos modelos mais recentes de aprendizado por reforço, o algoritmo convergiu para uma solução que ganha facilmente da barra oponente. Alguns indivíduos performaram tão bem que não sofreram nenhum ponto, conseguindo rebater todas as bolas. Por causa da simplicidade do modelo, com menos de 20 horas o modelo já tinha convergido e estava apenas refinando seus pesos.

Como neste trabalho utilizamos as posições da barra e da bolinha como entradas do modelo, não estamos retratando completamente a complexidade encontrada nos problemas reais, que em geral são apenas imagens. Como trabalho futuro, podemos utilizar as imagens do jogo ao invés das posições, recorrendo também a redes neurais convolucionais, mais profundas e complexas.

Outras abordagens que poderíamos tentar são a aceleração do jogo para treinar mais rápido e a paralelização por indivíduo, como se cada um estivesse jogando em uma janela diferente no computador. Dessa forma o treinamento seria muito mais rápido que o deste trabalho.

Referências bibliográficas

[1] MAO, H. *et al*. **Resource Management with Deep Reinforcement Learning**. ACM, New York, NY, hotnets '16 edition, 2016.

[2] AREL, I. *et al*. **Reinforcement learning-based multi-agent system for network traffic signal control**. IET Intelligent Transport Systems, 4:128–135, 2010.

[3] SILVER, D. *et al*. **Mastering the game of go with deep neural networks and tree search**. Nature, 529:484–489, 2016.

[4] SILVER, D. *et al*. **Mastering the game of go without human knowledge**. Nature, 550:354–359, 2017.

[5] HAUSKNECHT, M.J. AND STONE, P.. **Deep recurrent q-learning for partially observable mdps**. CoRR, abs/1507.06527, 2015.

[6] SUCH, F.P. *et al*. **Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning**. CoRR, abs/1712.06567, 2017.

[7] PROVOST, F. AND KOHAVI, R.. **On applied research in machine learning**. In: MACHINE LEARNING, p. 127–132, 1998.

[8] AWAD, W.A. AND ELSEUOFI, S.M... **Machine learning methods for spam e-mail classification**. International Journal of Computer Science and Information Technology, 3, 2011.

[9] WU, Q. *et al*. **The application of deep learning in computer vision**. In: 2017 CHINESE AUTOMATION CONGRESS (CAC), p. 6522–6527, Oct 2017.

[10] RUSSEL, S.T. AND NORVIG, P.. **Artificial Intelligence: A Modern Approach**, chapter 4, p. 510–565. Pearson, 01 1995.

[11] DELIANG, L.W.. **Unsupervised learning: Foundations of neural computation**. AI Magazine, 22:101–102, 06 2001.

[12] BERTSEKAS, D.. **Dynamic Programming and Optimal Control**, volumen 1, chapter 1, p. 40–43. MIT Press, 01 1995.

[13] VAN OTTERLO, M. AND WIERING, M.. **Reinforcement learning and markov decision processes**. Reinforcement Learning: State of the Art, p. 3–42, 01 2012.

[14] MNIH, V. *et al*. **Playing atari with deep reinforcement learning**. CoRR, abs/1312.5602, 2013.

[15] GOSAVI, A.. **Reinforcement learning: A tutorial survey and recent advances**. INFORMS Journal on Computing, 21:178–192, 05 2009.

[16] EIBEN, A. *et al*. **Genetic algorithms with multi-parent recombination**. In: PARALLEL PROBLEM SOLVING FROM NATURE, p. 78–87, 10 1994.

[17] BALUJA, R. AND CARUANA, R.. **Removing the genetics from the standard genetic algorithm**. In: THE PROCEEDINGS OF THE 12TH ANNUAL CONFERENCE ON MACHINE LEARNING, p. 38–46, 12 1995.

[18] VAN GERVEN, M. AND BOHTE, S.. **Editorial: Artificial neural networks as models of neural information processing**. Frontiers in Computational Neuroscience, 11, 12 2017.

[19] ZELL, A.. **Simulation neuronaler netze**. Universität Stuttgart, 08 1994.

[20] FREUND, Y. AND SCHAPIRE, R.. **Large margin classification using the perceptron algorithm**. Machine Learning, 37, 02 1999.

[21] VON DER MALSBURG, C.. **Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms**. Brain Theory, p. 245–248, 01 1986.