Implementação de um agente utilizando sistemas neurais híbridos para o ambiente CartPole

Carlos Henrique Kayser* Élder Francisco Fontana Bernardi †
João Mário Lopes Brezolin ‡

2018

Resumo

Este trabalho tem como objetivo apresentar os resultados da implementação de um sistema inteligente híbrido, por meio da otimização dos pesos de uma rede neural multicamadas, utilizando uma abordagem apresentada pelos algoritmos genéticos. Para avaliação do desempenho da metodologia proposta foi usado o desafio *CartPole-v0* da biblioteca *Gym* do *OpenAI*. Após os experimentos realizados foram obtidos resultados consideráveis, alcançando uma média de 280 episódios para a resolução do desafio.

Palavras-chave: Inteligência Artificial. Redes Neurais. Algoritmos Genéticos. Agentes. *OpenAI*.

1 Introdução

Um dos grandes problemas conhecidos em aprendizado por reforço (*Reinforcemment Learning*) é que treinar um algoritmo em um ambiente real, é muito complexo, pois no mundo real há diversas variáveis que podem impactar o treinamento de um algoritmo de inteligência artificial, bem como a validação deste.

Dada a complexidade desse problema, faz-se necessária a investigação de soluções capazes de lidar com a imprevisibilidade. Além disso, torna-se pertinente a implementação de soluções independentes das fornecidas pelas APIs de bibliotecas já

*Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet, na cidade de Passo Fundo, em 2018. Email: <carloshkayser@gmail.com>

†Orientador, professor do IFSUL. e-mail: <elder.bernardi@passofundo.ifsul.edu.br>

‡Co-orientador, professor do IFSUL. e-mail:<joao.brezolin@passofundo.ifsul.edu.br>

existentes, pois assim, pode-se contribuir com um maior entendimento dos mecanismos inerentes às suas soluções.

Dessa maneira, o intuito deste trabalho consiste no desenvolvimento de uma pesquisa investigativa experimental, desenvolvendo soluções em inteligência artificial, como redes neurais e algoritmos genéticos, ao oposto de optar por uma solução pronta, como por exemplo, a utilização de bibliotecas já implementadas, como *Keras*. Vale a pena ressaltar que não foram encontradas bibliotecas prontas em *Python* para a implementação proposta neste trabalho, por esse motivo, optou-se pelo desenvolvimento de uma rede neural híbrida em *Python*.

Portando, se faz necessária a pesquisa do referencial teórico necessário para auxiliar na implementação das Redes Neurais Artificiais e Algoritmos Genéticos. A implementação da metodologia proposta é vista como porta de entrada para o entendimento dos mecanismos existentes por trás de cada técnica, a fim de buscar a melhor compreensão desses.

Com o objetivo de isolar o ambiente de teste e evitar com que alguma variável possa impactar no desenvolvimento, treinamento e validação de um algoritmo inteligente, foi desenvolvido, pelo instituto *OpenAI*, a biblioteca *Gym*, que oferece diversos ambientes simulados, com o intuito de treinar algoritmos inteligentes e comparar seus resultados.

A fim de se desenvolver soluções em inteligência artificial e analisar seus resultados de maneira objetiva, faz-se uso do ambiente *CartPole-V0* da biblioteca *Gym*, que tem como objetivo manter um pêndulo inverso equilibrado em 90º graus em relação a sua base. Para tal feito, foram utilizadas redes neurais artificias e algoritmos genéticos, formando assim um sistema neural híbrido.

Este trabalho visa como principal objetivo implementar e compreender o funcionamento de Redes Neurais Artificiais otimizadas por meio do uso das heurística formuladas por algoritmos genéticos. Bem como a implementação dessas técnicas, levando em consideração que não há bibliotecas em *Python* desenvolvidas para essa finalidade, para então tornar possível a avaliação desta metodologia.

Este artigo está organizado em sete seções. A primeira introdutória que consiste na apresentação do trabalho proposto. A Seção 2 consiste no Referencial Teórico utilizado na pesquisa, como um breve funcionamento das redes neurais de multicamadas, algoritmos genéticos, agentes e uma descrição da abordagem evolutiva utilizada, por meio da otimização das redes neurais de multicamadas utilizando os algoritmos genéticos. A Seção 4 consiste na apresentação de outros trabalhos utilizando uma abordagem parecida da qual está descrito neste trabalho. Na Seção 3, está descrito o funcionamento do algoritmo proposto. Na Seção 5, está descrita a metodologia de avaliação do algoritmo desenvolvido utilizando a abordagem e o referencial proposto. Os resultados obtidos por meio da execução do algoritmo desenvolvido, avaliação e comparação com outros algoritmos se encontram na Seção 6. Na Seção 7 estão dispostas as considerações finais do projeto.

2 Referencial Teórico

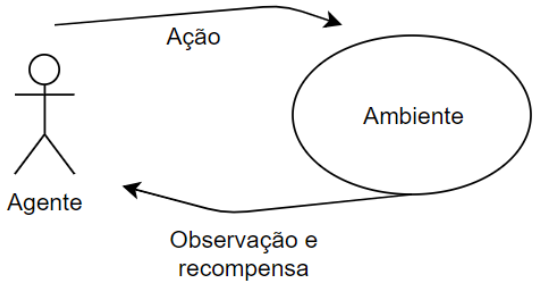
No decorrer desta seção serão apresentadas as metodologias e conceitos necessários para a compreensão e entendimento das técnicas abordadas no projeto, quais foram essenciais para o desenvolvimento do algoritmo.

2.1 Agentes

Para Idgem e Agents (1995), um agente é um sistema computacional, qual está situado em algum ambiente, e é capaz de realizar ações neste ambiente com o intuito de atingir um objetivo específico. Para Shoham (1997) um agente é uma entidade de software que funciona de forma contínua e autônoma em um ambiente, frequentemente habitado por outros agentes e processos.

De acordo com Russell e Norvig (2010), conforme Figura 1 um agente é considerado tudo o que é capaz de captar informações do ambiente, por meio de sensores, e é capaz de realizar ações sobre o ambiente, por meio de atuadores.

Figura 1 – Fluxograma de agente e ambiente



Fonte: Adaptado de OpenAI (2016)

Russell e Norvig (2010) declaram que um agente humano tem olhos, ouvidos e outros órgãos, os quais podem ser considerados como sensores, e possuem mãos, pernas, entre outras partes do corpo que servem de atuadores, levando em consideração que por meio desses atuadores o agente pode interagir com o ambiente. Dessa forma, um agente robótico pode ter câmeras, detectores que podem ser considerados como sensores e motores como atuadores.

O aprendizado é um requisito muito importante para a autonomia dos agentes, onde o agente precisa possuir a habilidade de avaliar as ações que podem ser tomadas de acordo com as variâncias do ambiente. Dessa forma, o processo de aprendizagem por experiência do agente está relacionado aos acertos e erros, de acordo com as tomadas de decisões do agente em relação ao ambiente (COSTA, 2003).

Portanto, a metodologia proposta neste trabalho, conforme citado nos parágrafos anteriores, pode ser caracterizada como um agente inteligente, pelo fato de poder realizar ações em um ambiente, receber observações do ambiente, no qual esta disposto e aprender com o ambiente.

2.2 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) pode ser definida como uma classe de algoritmo que é inspirada na biologia dos seres vivos, mais precisamente no cérebro humano. Um cérebro humano é constituído por diversas células interligadas chamadas de neurônios, que são responsáveis pelo raciocínio, controle motor, reconhecimento de padrões, etc (HAYKIN, 2001).

De acordo com Haykin (2001), uma rede neural pode ser definida como um processador paralelo distribuído, que consiste em unidades de processamento simples, denominados neurônios artificias, que tem como objetivo armazenar experiencias e torná-las disponíveis para o uso. De acordo com o mesmo autor, as redes neurais artificias se assemelham ao cérebro em dois aspectos: o conhecimento é adquirido através de um processo de aprendizagem, equiparando a um humano ao reconhecer dígitos manuscritos por exemplo; e os pesos sinápticos são utilizados para armazenar o conhecimento adquirido. O algoritmo de aprendizagem, qual é usado para o treinamento da rede neural, tem por função modificar os pesos sinápticos da rede, com o intuito de atingir o objetivo desejado.

2.2.1 Redes neurais de múltiplas camadas

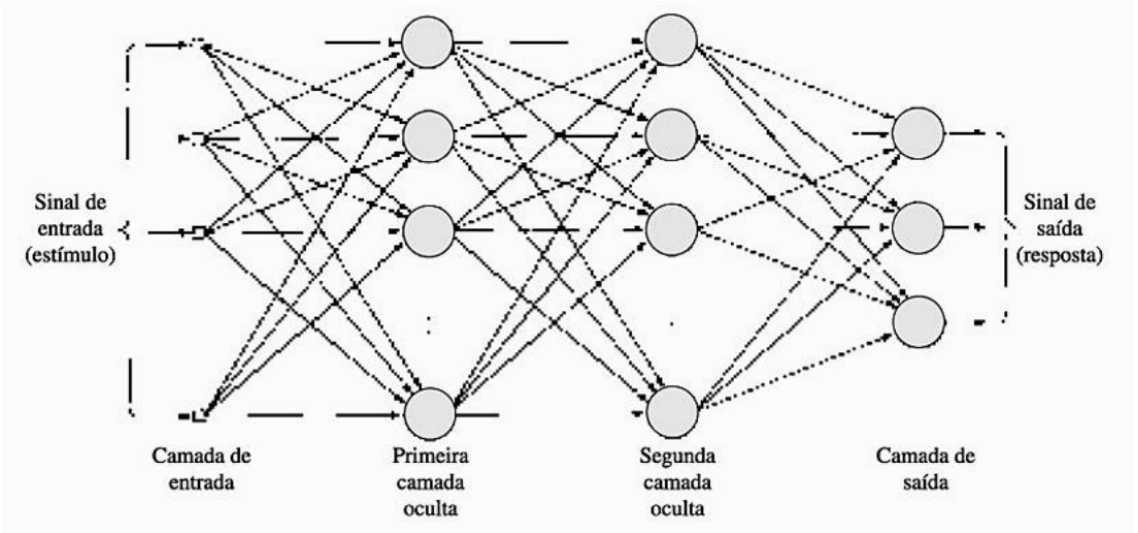
As redes neurais de múltiplas camadas, também conhecidas como *perceptrons* de multicamadas (MLP, *multilayer perceptron*), são consideradas uma classe de redes neurais artificiais. A arquitetura dessa classe de RNA's, consiste em uma rede neural composta por uma camada de entrada, uma ou mais camadas de neurônios, também conhecidos como nós de processamento, e uma camada de saída. Nesse tipo de rede neural, geralmente o sinal de entrada se propaga em um único sentido, da entrada da rede até a saída (HAYKIN, 2001).

Esse método de propagação também é conhecido como *feed-forward*, onde o impulso entra na rede neural pela primeira camada e se propaga até a última camada. Consistem em redes neurais que possuem camadas discretas de neurônios interconectadas, onde há uma camada de entrada, uma ou mais camadas ocultas ou discretas, que por sua vez executam alguns cálculos e passam o resultado desses cálculos para uma próxima camada, seja ela outra camada oculta ou a camada de saída (GRUS, 2016) (GOLDSCHMIDT, 2010).

As RNA's são compostas por neurônios interligados, dispostos em camadas de processamento. Os neurônios são conectados por ligações direcionadas, onde cada neurônio de uma camada é conectado com todos os neurônios da próxima camada. Cada ligação entre os neurônios possuem um peso numérico, também conhecido como peso sináptico (RUSSELL; NORVIG, 2010).

Na Figura 2 está representado uma rede neural de múltiplas camadas. A rede ilustrada possui uma camada de entrada, qual não realiza nenhuma mudança nos dados de entrada, duas camadas ocultas com quatro neurônios cada e uma camada de saída. Todos os neurônios estão conectados com todos os neurônios da próxima camada.

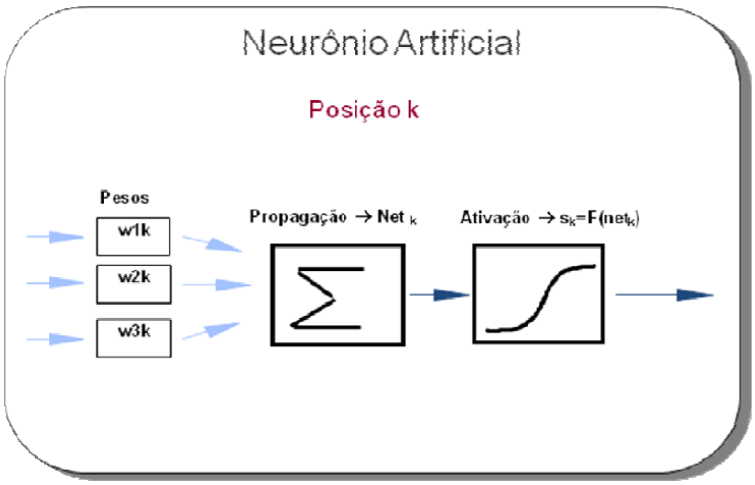
Figura 2 – Rede neural de múltiplas camadas



Fonte: Haykin (2001)

O funcionamento da RNA, pode ser exemplificado da seguinte forma: em cada entrada dos neurônios artificiais há um peso correspondente (W_{i_j}), onde cada entrada é multiplicado pelo seu peso correspondente, gerando valores de entradas ponderadas que em seguida são somados, gerando um valor NET (net_i ,potencial de ativação do neurônio artificial) que em seguida será comparado com o valor limite da ativação do neurônio (F). Caso o valor obtido ultrapasse o valor limite da ativação, o neurônio então é ativado (GOLDSCHMIDT, 2010) (GRUS, 2016). A estrutura de um neurônio artificial encontra-se ilustrada na Figura 3.

Figura 3 – Estrutura Interna de um Neurônio Artificial



Fonte: Goldschmidt (2010)

O produto gerado a partir das somas ponderadas entre os valores de entrada dos neurônios e pesos sinápticos são transmitidos para uma função de ativação (F). Onde, conforme Equação 1, Y_i é a saída gerada pelo neurônio i, o potencial

de ativação do neurônio é representado por net_i. A constante b_i é conhecida como constante de polarização (bias), que tem por objetivo aumentar ou diminuir a entrada líquida da função de ativação. Após o cálculo nesse processo, o resultado é propagado para a próxima camada de neurônios artificiais (GOLDSCHMIDT, 2010; BARRETO, 1999; GRUS, 2016).

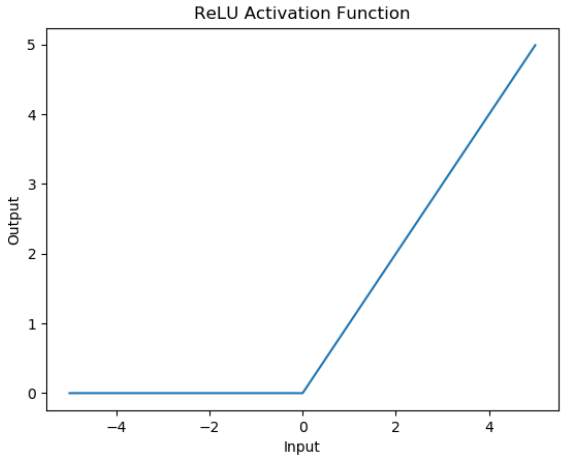
$$Y_i = F(net_i + b_i) \tag{1}$$

As funções de ativação desempenham um papel importante em uma RNA, pois evitam que ocorram mudanças bruscas nos resultados obtidos pelas ativações dos neurônios a partir de pequenas alterações nos pesos e bias da RNA. As funções de ativação sigmoidais (como a logística e a tangente hiperbólica) são muito comuns em RNA's. Porém, em alguns casos o uso da função de ativação retificadora linear (*rectified linear function*, ReLU) tem sido bem vista para RNA's de grande complexidade. Isso se dá pelo fato de que as funções de ativação sigmoidais saturam, a partir de um determinado ponto, enquanto que a função de ativação ReLU é uma função retificadora linear que cancela os valores negativos (PONTI; COSTA, 2018).

Na Figura 4 está disposta graficamente a função de ativação ReLU, cuja equação de encontra em Equação 2. No gráfico, é possível notar que a utilização da função ReLU cancela todos os valores negativos, mantendo os valores positivos, pelo fato de ser uma função identidade para valores positivos.

$$ReLU(x) = max\left\{0, x\right\} \tag{2}$$

Figura 4 – Função de ativação reLU



Fonte: do autor

2.3 Algoritmos Genéticos

Os algoritmos genéticos (AG) são algoritmos adaptativos baseados no processo evolutivo e adaptativo dos seres vivos. São usados para resolver problemas de busca e otimização. Na natureza, a combinação de boas características provenientes de

ancestrais, pode produzir descendentes com características mais fortes, cuja adaptação tende ser maior do que seus ancestrais, dessa forma a evolução dos organismos se torna constante (FERNANDES, 2003).

No trabalho de Holland (1992), o autor descreve que os algoritmos genéticos são técnicas baseadas na seleção natural, proposta por Darwin (1859), e a reprodução sexual dos organismos.

Segundo Holland (1992), as primeiras tentativas de juntar a computação com a biologia, entre 1950 e final da década de 1960, não obtiveram sucesso, pois as técnicas utilizadas naquela época se baseavam na mutação dos seres vivos. Nesse mesmo período e na década seguinte, Holland desenvolveu a técnica de programação que tempo depois se tornaria o algoritmo genético, que se baseia na evolução por acasalamento e mutação.

Os algoritmos genéticos trabalham com populações de indivíduos, que por sua vez representam uma possível solução para um problema proposto. Quanto maior for a aptidão desse indivíduo com o problema, maior é a chance deste ser selecionado para se reproduzir, cruzando seu material genético com outro indivíduo cuja aptidão seja semelhante (FERNANDES, 2003).

Uma população é composta por indivíduos. Indivíduos são conjuntos de parâmetros, também conhecidos como genes, que em conjunto formam o cromossomo do indivíduo. Dessa forma, os indivíduos são possíveis soluções para o problema (FERNANDES, 2003).

O mecanismo de codificação dos algoritmos genéticos possui grande importância no algoritmo genético, pois dependo da forma em que um algoritmo genético for codificado, o processo de busca pode ser facilitado. Há duas representações: codificações binárias e codificações utilizando o sistema numérico decimal. Cada qual é apropriada para um tipo de problema (NETO, 2006).

A avaliação dos indivíduos é realizada de forma individual, por meio de uma função de aptidão. A função de aptidão também é vital para verificar se o indivíduo atingiu o objetivo ou o ponto procurado (NETO, 2006).

Os algoritmos genéticos são constituídos por operadores genéticos (OG), tais como: o cruzamento, a mutação e a seleção. A utilização desses mecanismos biologicamente inspirados são vitais para o funcionamento de um algoritmo genético (HOLLAND, 1992; NEGNEVITSKY, 2005).

No processo de seleção, os indivíduos são selecionados conforme seu nível de aptidão para formarem uma nova população, contribuindo com seu material genético (genes) (NETO, 2006).

Um método de seleção geralmente utilizado, é o método da roleta, onde cada indivíduo tem um fatia de uma roleta com uma proporção equivalente ao seu nível de aptidão, isto significa que quanto maior for a taxa de aptidão do indivíduo, maior será sua fatia. Cada fatia da roleta é dividida por meio da normalização das notas de todos os indivíduos da população, por meio da Equação 3, onde a nota normalizada é calculada através da divisão da nota original, pela soma de todas as demais notas originais (BRAGA; CARVALHO; LUDERMIR, 2000).

$$nota_j^{normalizada} = \frac{nota_j^{original}}{\sum_N^{i=1} nota_i^{original}} \quad (3)$$

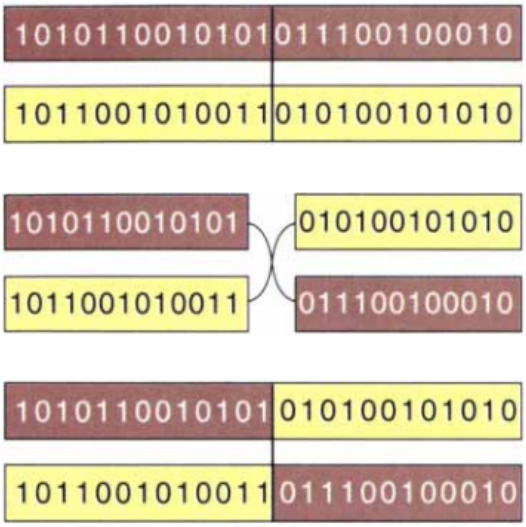
Neste método, a roleta é girada de forma aleatória N vezes, selecionando os N indivíduos para a realização do processo de reprodução. Consequentemente, o indivíduo com maior taxa de aptidão terá uma maior probabilidade de ser selecionado (BRAGA; CARVALHO; LUDERMIR, 2000).

O processo de cruzamento é realizado pela aproximação dos dois indivíduos que estão mais aptos ao ambiente. Este processo consiste na criação de novos indivíduos a partir do cruzamento dos cromossomos de seus ancestrais. Porém as características dos cromossomos antes do processo de cruzamento são passadas para os novos cromossomos resultantes do processo (HOLLAND, 1992).

O cruzamento é o operador responsável pela recombinação de características dos pais, permitindo que elas sejam herdadas pelas próximas gerações. Como esse processo é considerado predominante, é aplicado com o auxilio de uma probabilidade, chamada de taxa de cruzamento, que é maior que a taxa de mutação. O cruzamento entre dois indivíduos pode ser feito de três formas: cruzamento de um ponto, cruzamento de multipontos e cruzamento uniforme (BRAGA; CARVALHO; LUDERMIR, 2000). Abordaremos neste trabalho somente o cruzamento de um ponto.

Holland (1992), ilustra o processo de cruzamento ou *crossover* entre os cromossomos, conforme Figura 5.

Figura 5 – Processo de *Crossover*



Fonte: Sivanandam e Deepa (2008)

Segundo Barreto (1999) o processo de mutação consiste na alteração genética dos cromossomos por meio de vários mecanismos, que resultará em um novo cromossomo com alguma ou nenhuma característica de seus ancestrais. Dentre esses

mecanismos de alteração genética, há os mecanismos de troca simples, inversão e translocação. Segue abaixo a explicação de cada um deles:

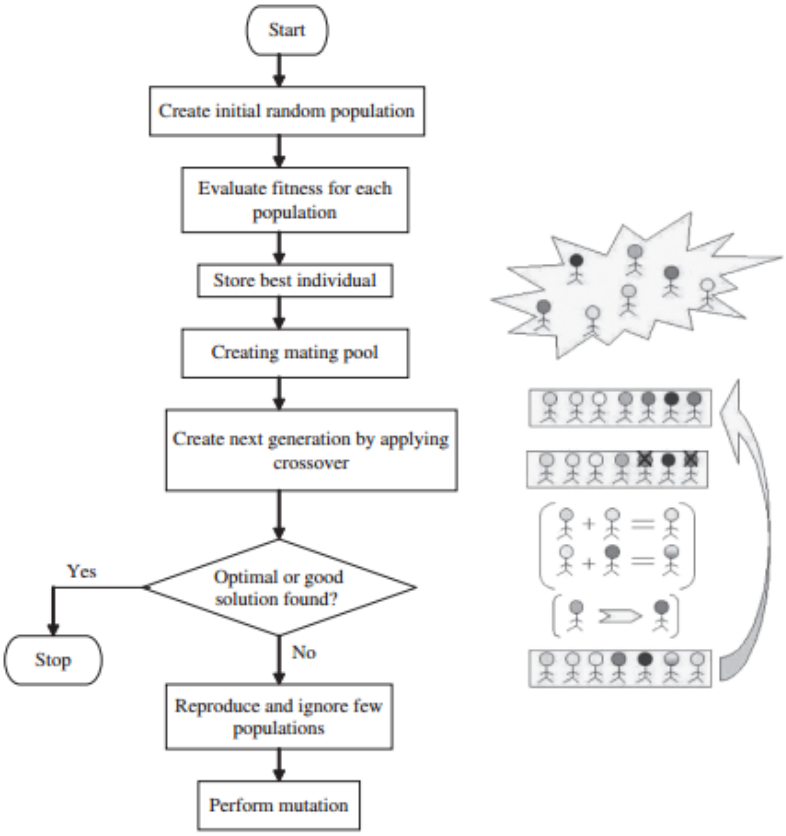
- O mecanismo de troca simples, consiste em trocar aleatoriamente o valor de alguns genes no cromossomo, onde por exemplo, no caso do valor do gene ser binário, o método de troca simples trocaria o valor para seu oposto, de 0 para 1 ou de 1 para 0;
- O mecanismo de inversão consiste em retirar uma pequena fatia de genes do cromossomo, inverter a sequência desses genes e recolocar essa fatia em sua posição anterior;
- O mecanismo de translocação consiste em retirar uma fatia de genes do cromossomo e recolocá-la em outra posição no mesmo cromossomo, seja ela no início, meio ou fim do cromossomo.

De acordo com Sivanandam e Deepa (2008), a Figura 6 apresenta o fluxograma do processo que é executado pelos algoritmos genéticos, onde:

- Início/*Start*: é realizada a criação randômica da genética dos cromossomos;
- Validação/*Fitness*: função destinada a avaliar o quanto apto o cromossomo é em relação ao ambiente;
- Seleção: a partir do nível de aptidão fornecida pela função de validação, são selecionados os dois melhores cromossomos da população para ser realizado posteriormente o processo de *crossover*;
- Cruzamento/*Crossover*: a partir da seleção dos melhores cromossomos e dependendo da probabilidade de cruzamento então é realizado o processo de cruzamento;
- Mutação/*Mutation*: dependendo da probabilidade de mutação, a mesma é realizada após o processo de cruzamento, qual ocorre nos genes dos cromossomos;
- Aceitação: após a execução de todos os processos descritos anteriormente, é executado o processo de aceitação dos novos cromossomos.

Caso os novos cromossomos não se encontrem aptos para solucionarem o problema a eles proposto, é realizado o processo de validação, seleção, operadores genéticos e aceitação novamente, até que seja alcançado o objetivo proposto.

Figura 6 – Fluxograma do processo do algoritmo genético



Fonte: Sivanandam e Deepa (2008)

O processo executado pelos operadores genéticos são normalmente executados na seguinte ordem: primeiramente é aplicado o processo de *crossover* e após é executado o processo de mutação (BARRETO, 1999).

A taxa de mutação é uma taxa que define a probabilidade de um indivíduo ser mutado ou não, cuja probabilidade possui valores típicos entre 0.001 até 0.1. Basicamente a taxa de mutação impede que a busca por indivíduos fique estagnada. (NETO, 2006).

O indivíduo elite, é abordado por Neto (2006) e Braga, Carvalho e Ludermir (2000) como politica elitista. Este é um parâmetro existente em algoritmos genéticos que consiste em manter uma quantidade de indivíduos intacta nas próximas gerações, cuja taxa de aptidão é alta. Neste trabalho, utilizamos apenas um indivíduo elite.

2.4 Sistemas Neurais Híbridos

Diante de diversas vantagens e desvantagens existentes nos principais algoritmos abordados por *machine learning*, nasce a necessidade de unir algoritmos a fim de complementar as desvantagens de cada um. Dessa maneira surge o termo Sistemas Neurais Híbridos (SNH) (ARNAUD, 2007).

Os SNH's são conhecidos pela combinação de uma RNA e outro método conhecido em Inteligência Artificial, como: algoritmos genéticos, estatística, linguagem

natural, etc (BRAGA; CARVALHO; LUDERMIR, 2000).

Os algoritmos genéticos são bem vistos no treinamento dos pesos de uma RNA, baseando-se no processo evolutivo dos indivíduos e seu aperfeiçoamento por meio dos operados genéticos (ARNAUD, 2007).

Nessa metodologia, uma RNA pode ser representada pela concatenação de seus pesos, bem como o bias. A concatenação desses valores forma um individuo formulado em algoritmos genéticos, que pode ser representado em forma de vetor (YAMAZAKI, 2004).

Dessa maneira, neste trabalho, foi utilizado uma metodologia baseada na otimização dos pesos sinápticos de uma rede neural de multicamadas, neste caso, por meio da utilização dos operadores genéticos estipulados por algoritmos genéticos.

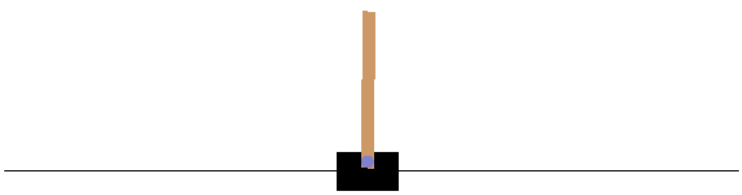
2.5 *CartPole*

O *CartPole-v0* é um ambiente da biblioteca *Gym*, desenvolvida pela empresa *OpenAI*. Esta, é uma empresa de pesquisa de Inteligência Artificial sem fins lucrativos, que busca busca criar ferramentas destinadas ao desenvolvimento e comparação de algoritmos de aprendizagem por reforço.

O ambiente *CartPole-v0* consiste em um pêndulo inverso preso a um carrinho que pode se mover em um eixo horizontal, para a esquerda ou direita, no qual o objetivo é manter o pêndulo equilibrado a 90º graus. O ambiente aqui citado oferece uma interface para que seja possível a realização de ações e obtenção de informações referentes ao estado no qual se encontra no ambiente, onde a cada ação realizada, pode-se observar sua posição, velocidade, ângulo e velocidade angular.

A cada ação realizada no ambiente, o sistema retorna uma recompensa, que pode ser 1 se a ação realizada pontuou de forma a alcançar o objetivo do ambiente, ou 0 se a ação realizada não foi benéfica. Caso o pendulo caia, por exemplo, o ambiente através da interface comentada anteriormente, sinaliza que a simulação terminou através de uma *flag* com valor *booleano*. Um episodio é caracterizado por uma simulação do ambiente, e termina com a sinalização da *flag* informada anteriormente. O desafio do *CartPole-v0* é considerado concluído quando conseguirmos manter o pêndulo equilibrado por 200 pontos (OPENAI, 2016; CHAN, 2016). O ambiente citado está ilustrado na Figura 7.

Figura 7 – Ambiente *CartPole-v0* em execução



Fonte: do autor

A biblioteca *Gym*, é uma biblioteca implementada na linguagem de programação *Python*, na versão 3.5.

3 Metodologia proposta

Este trabalho tem por objetivo implementar e analisar uma metodologia baseada em sistemas neurais híbridos, para otimização dos pesos sinápticos de uma rede neural MLP, utilizando uma abordagem baseada em algoritmos genéticos, a fim de cumprir os objetivos impostos pelo ambiente *CartPole-v0* da biblioteca *Gym*, do *OpenAI*.

O *CartPole-v0* da biblioteca *Gym*, foi utilizado por ser um ambiente virtual controlado que está salvo de eventos que poderiam impactar no processo de aprendizagem do agente, bem como a avaliação do desempenho e análise de métricas das decisões tomadas pelo agente. Outro ponto importante, relacionado à utilização de um ambiente virtual controlado, está associado à presença da facilidade da captura das informações do ambiente, como por exemplo, as informações referentes ao aos ângulos do pêndulo.

A Rede Neural Artificial foi implementada com base nos exemplos existentes em Grus (2016). A implementação da estrutura proposta pelos algoritmos genéticos, foram implementadas pelo autor, com base no referencial sobre o assunto.

A partir dos dados de entrada, os quais foram capturados durante o tempo de execução do agente, ou seja, durante a interação do agente com o ambiente, será realizado o processamento desses dados pela rede neural multicamadas, o qual será responsável por gerar uma saída correspondente aos dados de entrada. Essa saída refletirá em qual ação que o agente irá tomar sob o ambiente.

A partir de cada ação tomada pelo agente, será avaliada se a ação executada causou algum impacto positivo no jogo, ou seja, se a ação tomada está relacionada com o objetivo proposto pelo jogo. Dessa forma, será fornecido ao agente uma avaliação de sua aptidão em relação ao ambiente.

3.1 Tecnologias e ambiente de desenvolvimento

A arquitetura aqui proposta, foi implementada na linguagem de programação *Python*, na versão 3.5.2, com apoio da biblioteca Numpy na versão 1.14.5 e a biblioteca *Matplotlib* na versão 2.2.3 para plotagem dos dados em forma de gráficos. Os experimentos foram realizados em um notebook, contendo 8GB de memoria RAM DDR4 e um processador Intel® Core™ i5-7200U.

4 Trabalhos relacionados

No artigo publicado por Chan (2016) e no trabalho de Clark, Lakshminarayanan e Sonawani (2018), os autores implementaram e apresentaram os resultados obtidos da resolução do desafio *CartPole-v0* utilizando um método de *Machine Learning* conhecido como *Q-Learning*, que consiste na criação de uma politica baseada

em acertos e erros. O autor Chan (2016) conseguiu resolver o desafio *CartPole-v0* em 136 episódios, após uma série de adaptações no algoritmo de *Q-Learing*. Já os autores Clark, Lakshminarayanan e Sonawani (2018), conseguiram resolver o desafio com 268 tentativas ou episódios.

A dissertação de mestrado de Pereira (2012),"Q-Learning Pessimista: um algoritmo para geração de bots de jogos em turnos". A dissertação aborda a criação de um algoritmo capaz de gerar robôs virtuais, capazes de jogar jogos baseados em turnos, no qual busca contribuir para a obtenção de melhores resultados através da extensão do algoritmo *Q-Learing*.

Desse modo, o presente trabalho também possui como objetivo implementar uma solução para o desafio *CartPole-v0* e analisar seus resultados, em comparação com os abordados nesta seção. Dissertado os trabalhos relacionados existentes, na próxima seção está descrita a metodologia utilizada neste trabalho para a solução do desafio *CartPole-v0*.

4.1 Solução proposta

Para tornar o agente autônomo, foi utilizada as técnicas propostas pelos algoritmos genéticos, para realizar a otimização dos pesos de uma RNA, onde primeiramente foi gerada uma população para o treinamento do agente, qual constituiu por 50 indivíduos.

Cada indivíduo é composto por um vetor de pesos, ou seja, cada indivíduo possui uma possível solução para o ambiente. O tamanho do indivíduo depende da quantidade de camadas e neurônios artificiais da RNA. A primeira população foi gerada de forma randômica, ou seja, os valores dos indivíduos foram definidos aleatoriamente. Esses pesos foram utilizados para realizar os cálculos nos neurônios artificiais na RNA, em conjunto com os valores de entrada da rede.

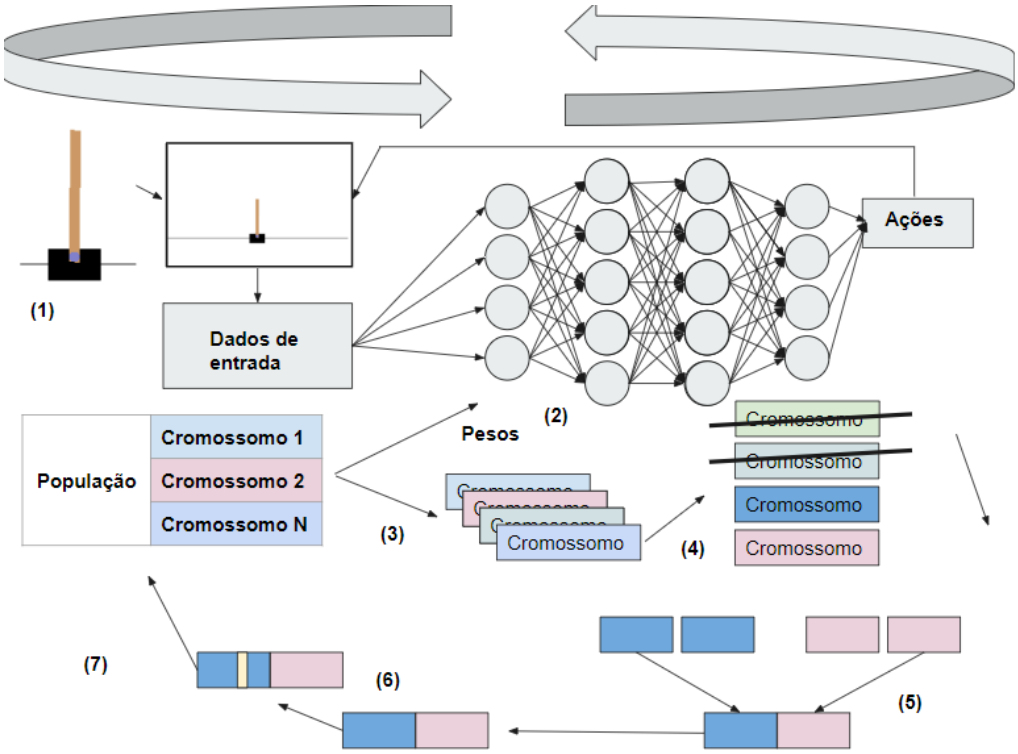
No passo seguinte estabeleceu-se a interação da primeira população com o ambiente, onde cada indivíduo da população foi avaliado, a fim de verificar qual indivíduo teve o maior índice de aptidão em relação ao ambiente, ou seja, qual individuo teve os melhores pesos sinápticos, quais foram processados pelos neurônios artificiais da rede neural, que executaram as melhores ações.

A partir da obtenção dos dois melhores indivíduos da população inicial, foi realizado entre esses indivíduos o processo de cruzamento, e a partir dos novos indivíduos gerados, foi executado o processo de mutação com a intuição de realizar pequenas alterações no indivíduo, que em conjunto com os novos indivíduos gerados aleatoriamente, formou uma nova população. Esse processo foi executado repetidamente, até gerar um indivíduo que conseguisse atingir o objetivo do ambiente, neste caso, atingisse 200 pontos no ambiente *CartPole-V0*.

A RNA foi utilizada para efetuar os cálculos necessários, a partir dos dados de entrada obtidos do ambiente no qual o agente se encontra. Para o cálculo, foram utilizados os pesos comentados anteriormente, onde a partir de várias execuções das populações ou gerações, tendeu-se a encontrar quais são os pesos ideais para gerar as melhores saídas.

O fluxograma ilustrado na Figura 8, apresenta qual foi o fluxo dos processos executados, utilizando as técnicas abordadas, a fim de tornar o agente totalmente autônomo, ou seja, quais foram os passos necessários para fazer com que agente atingisse os objetivos propostos pelo ambiente sem nenhuma intervenção. O processo de mutação e cruzamento dos cromossomos foi executado inúmeras vezes, até que o objetivo proposto pelo ambiente seja alcançado pelo agente.

Figura 8 – Fluxograma do agente. Figura ilustrativa



Fonte: do autor

A Figura 8 demonstra o fluxograma dos processos executados, onde em primeiro momento foram obtidos os dados de entrada do ambiente pelo agente (1), os dados então foram processados pela RNA em conjunto com os pesos definidos pelo AG (2), após o processamento desses dados, é gerado pela RNA uma saída, que neste caso será entre 0 e 1 que representará qual ação será realizada no ambiente, após o processamento de todos os cromossomos da população, os mesmos são avaliados (3), logo é realizado o processo de seleção dos melhores cromossomos, ou seja, os quais possuíram as maiores taxas de aptidão no ambiente (4), quais posteriormente passaram pelo processo de cruzamento (5) e mutação (6), após a execução de todos os processos é gerada a nova população de cromossomos (7). Esse ciclo é executado até o agente alcançar a autonomia.

Para definirmos quais são os melhores indivíduos, quais teoricamente deveriam possuir os melhores pesos, foi utilizada uma função de aptidão, que é responsável em pontuar o quanto adepto um agente é em relação ao ambiente no qual foi alocado. Quando o algoritmo encontrar uma solução para o ambiente proposto, isto é, quando

o algoritmo atingir uma solução e ficar estagnado nesta, um mecanismo de parada irá finalizar as simulações e apresentar os resultados.

4.2 Considerações sobre o desenvolvimento

O protótipo foi implementado em *Python*, onde foi usado uma rede neural multicamadas de quatro neurônios artificiais na primeira camada, quatro neurônios na primeira camada oculta, três neurônios na segunda camada oculta e um neurônio para saída. Como o ambiente aceita como ação somente 0 ou 1, foi implementado na saída da rede neural um conversor, onde, se o resultado gerado pela rede neural for maior que 0,50 a saída então será 1, caso o resultado da rede neural fosse menor que 0,50 a saída será 0.

No protótipo, na implementação da Rede neural Artificial, utilizou-se a função de ativação *ReLU*. Como parametrização dos algoritmos genéticos, utilizamos como taxa de mutação e taxa de cruzamento o valor de 0,01. O tamanho da população utilizada foi de 50 indivíduos. O processo de seleção dos indivíduos efetuado pelo método da roleta, foi implementado para selecionar 50% dos indivíduos da população.

Foi implementada a função de parada, com o objetivo de finalizar as simulações quando o algoritmo de treinamento estagnar, isto é, ao alcançar uma determinada pontuação e não progredir. O cruzamento dos cromossomos foi implementado de ponto a ponto. O processo de mutação em nossa implementação consiste em alterar os pesos substituindo-os por um valor gerado de forma aleatória entre 0 a 1, gerados com apoio da biblioteca *Numpy*.

5 Avaliação do protótipo

O desempenho do agente será avaliado levando em consideração a quantidade de episódios necessários para convergir os pesos da RNA a fim de se obter domínio do ambiente, isto é, manter o pêndulo equilibrado.

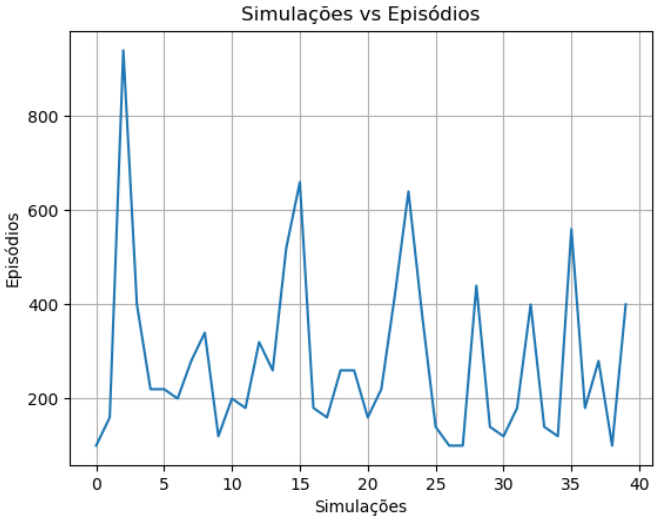
Como a metologia proposta este trabalho utiliza algoritmos genéticos, que consiste em métodos baseados em aleatoriedade, para validar o desempenho do algoritmo implementado, foi realizado uma bateria de testes, com 50 execuções do algoritmo, onde será extraído dessa bateria de testes as médias de episódios necessários para alcançar o objetivo proposto pelo ambiente. Os resultados serão apresentados na Seção 6.

6 Resultados

Os resultados da execução da bateria de testes citada na Seção anterior gerou os seguintes resultados: o mínimo de episódios necessários para solucionar o desafio do *CartPole-v0* foi de 100 episódios, o máximo de episódios necessários para solucionar o desafio foram 940 episódios e a média de episódios foi de 280 episódios. Com o algoritmo implementado, obteve-se uma taxa de 80% de sucesso, onde de 50 testes realizados, tivemos um total de 40 simulações que obtiveram sucesso. Esses dados

estão ilustrados graficamente na Figura 9, onde é relatada a relação entre episódios necessários para solucionar o desafio e a quantidade de simulações que obtiveram sucesso.

Figura 9 – Relação entre simulações e episódios necessários para atingir o objetivo

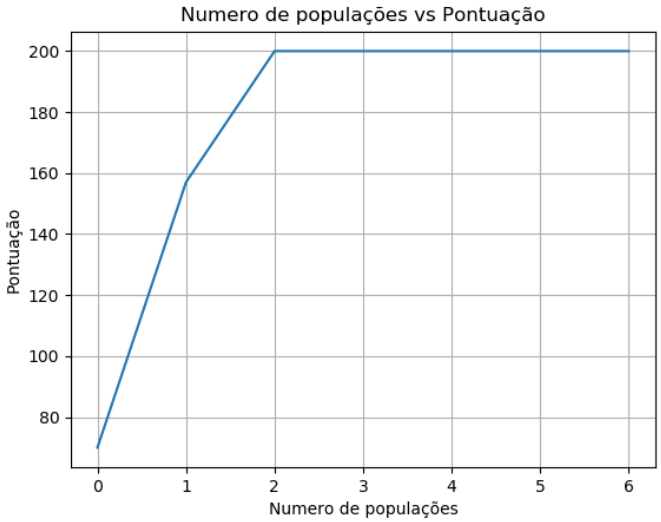


Fonte: do autor

Foram necessárias uma média de 280 iterações do algoritmo proposto para que fosse possível atingir o objetivo proposto pelo ambiente. Vale a pena ressaltar que nos experimentos realizados, a utilização da função de ativação *sigmoid* não gerou resultados válidos, levando em consideração que as saídas da RNA não geravam boas ações no ambiente.

Na Figura 10, está ilustrado a execução de somente uma simulação, qual completou o desafio em 350 episódios. Nessa simulação, o algoritmo foi treinado por meio de seis populações, cada uma constituída por 50 indivíduos. É possível notar que o máximo de pontos obtidos, cresce a cada nova população gerada.

Figura 10 – Relação entre a pontuação e quantidade de populações



Fonte: do autor

7 Considerações finais

Este trabalho abordou a implementação e análise de um sistema neural híbrido, que consiste na implementação de uma RNA, qual possui seus pesos otimizados por meio do uso e inspiração biológica abordados pelos algoritmos genéticos, para a realização do desafio *CartPole-v0* de *OpenAI*.

Para a implementação da metodologia proposta foi utilizada a linguagem de programação Python e as bibliotecas Numpy, para a manipulação dos dados, e matplotlib, para a plotagem dos dados.

Com a implementação da metodologia, obteve-se resultados condizentes com o esperado na proposta de avaliação, atingindo o objetivo do ambiente em uma média de 280 episódios. Outras implementações utilizando outras abordagens, como os utilizados, por exemplo, nos trabalhos de Chan (2016) e Clark, Lakshminarayanan e Sonawani (2018), atingiram o objetivo do ambiente proposto com 200 e 136 episódios respectivamente. Embora os resultados obtidos foram inferiores aos trabalhos citados, salienta-se que a metodologia proposta é baseada em mecanismos aleatórios embasados na biologia dos seres vivos.

Aproveitando a implementação da metodologia proposta, foi realizado experimentos com o ambiente *CartPole-v1*, que diferentemente do utilizado no trabalho (*CartPole-v0*) possui uma maior dificuldade, onde é necessário 500 pontos para atingir o objetivo do ambiente e não somente 200. Após esses experimentos foi constatado que o algoritmo necessitou de apenas 300 episódios para atingir o objetivo.

Este trabalho pode auxiliar a comunidade de inteligencia artificial, contribuindo para o desenvolvimento e compreensão de algoritmos genéticos e redes neurais artificiais, para a formulação de um sistema neural híbrido. Como forma de contribuição, o código da implementação da metologia está disponível no usuário Carlos

Henrique Kayser, no GitHub. [1]

Para trabalhos futuros é sugerido a implementação da metodologia proposta em outros ambientes da biblioteca *Gym*, bem como utilizar outras técnicas abordadas por *machine learning*, a fim de comparar a eficiência das metodologias.

Abstract

This work aims to present an implementation of a multilayer neural network that uses genetic algorithms to optimize the weights of the synaptic neurons. For the evaluation of the proposed methodology, we use the CarPole-v0 challenge from gym library of OpenAi. After execution of the experiments, effective results were obtained, reaching an average of 280 episodes to solve the challenge.

Keywords: Artificial intelligence. Neural networks. Genetic Algorithms. Agents. Hybrid Neural Systems. CartPole. OpenAI.

Referências

ARNAUD, A. Abordagem híbrida para otimização de redes neurais artificiais para previsão de séries temporais. *UFPE, Dissertação de Mestrado*, 2007.

BARRETO, J. M. Inteligência artificial no limiar do século xxi. *Florianópolis: PPP edições*, v. 97, 1999.

BRAGA, A. d. P.; CARVALHO, A.; LUDERMIR, T. B. *Redes neurais artificiais: teoria e aplicações*. [S.l.]: Livros Técnicos e Científicos Rio de Janeiro, 2000.

CHAN, M. *Cart-Pole Balancing with Q-Learning*. 2016. Disponível em: <https://medium.com/@tuzzer/cart-pole-balancing-with-q-learning-b54c6068d947>.

CLARK, G.; LAKSHMINARAYANAN, V.; SONAWANI, S. Final project-q learning to balance inverted pendulum. 2018. Disponível em: <http://venkatavaradhan.com/docs/FinalProject_Clark_Sonawani_Lakshminarayanan.pdf>.

COSTA, M. T. C. *Uma arquitetura baseada em agentes para suporte ao ensino à distância. 1999*. Tese (Doutorado) — Tese (Doutorado)–Departamento de Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, SC., 1999. Disponível em:< http://www. eps. ufsc. br/teses99/thiry/>. Acesso em: maio de, 2003.

DARWIN, C. On the origins of species by means of natural selection. *London: Murray*, v. 247, p. 1859, 1859.

[1] Disponível em: http://bit.ly/TCC-Artificial-Intelligence

FERNANDES, A. M. da R. *Inteligência Artificial: noções gerais*. [S.l.]: Visual Books, 2003.

GOLDSCHMIDT, R. R. Uma introdução à inteligência computacional: fundamentos, ferramentas e aplicações. *Rio de Janeiro Brasil: IST-Rio*, p. 32, 2010.

GRUS, J. *Data Science do zero: Primeiras regras com o Python*. [S.l.]: Alta Books Editora, 2016.

HAYKIN, S. Redes neurais princípios e aplicações. *Segunda Edição. Porto Alegre*, 2001.

HOLLAND, J. H. Genetic algorithms. *Scientific american*, JSTOR, v. 267, n. 1, p. 66–73, 1992.

IDGEM, W.; AGENTS, J. I. N. R. I. Theory and practice. *The Knowledge Engineering Review*, v. 2, 1995.

NEGNEVITSKY, M. *Artificial intelligence: a guide to intelligent systems*. [S.l.]: Pearson Education, 2005.

NETO, N. Aplicação de redes neurais artificiais e algoritmos genéticos na previsão de carga elétrica em médio prazo. *Master's thesis, UFPE*, 2006.

OPENAI. *Getting Started with Gym*. 2016. Disponível em: <https://gym.openai.com/docs/>.

PEREIRA, A. B. Q-learning pessimista: um algoritmo para geração de bots de jogos em turnos. PUC-Rio, 2012.

PONTI, M. A.; COSTA, G. B. P. da. Como funciona o deep learning. *arXiv preprint arXiv:1806.07908*, 2018.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. [S.l.]: Malaysia; Pearson Education Limited„ 2010.

SHOHAM, Y. An overview of agent-oriented programming. *Software agents*, Menlo Park, CA, v. 4, p. 271–290, 1997.

SIVANANDAM, S.; DEEPA, S. Genetic algorithm optimization problems. In: *Introduction to Genetic Algorithms*. [S.l.]: Springer, 2008. p. 165–209.

YAMAZAKI, A. Uma metodologia para otimização de arquiteturas e pesos de redes neurais. Universidade Federal de Pernambuco, 2004.