

# **Implementação do Self Organizing Map como rede neural para um jogo simples**

A equipe de desenvolvimento é formada por Lucas Figueredo Varela Alves e Luis Felipe Vanin Martins, discentes da disciplina de Tópicos Avançados em Informática I, ministrada pelo docente Orivaldo Vieira de Santana.

## **Introdução**

Quando se trata de inteligência artificial o principal objetivo daqueles que trabalham com essa tecnologia é o de criar um modelo que possua a capacidade de executar as mesmas atividades de um ser humano com uma eficiência equivalente ou melhor. Para demonstrar as capacidades de engines e IA's, empresas e desenvolvedores promovem embates diretos entre máquinas e humanos, como por exemplo confrontos em jogos ou esportes. Um dos casos mais memorável do confronto "máquina vs humano" foi o jogo de xadrez realizado pela IBM(International Business Machines Corporation), em 1996 e 1997, que colocou Garry Kasparov contra a CPU "Deep Blue" com o único propósito de mostrar as capacidades de seus softwares e hardwares.

Em 2019, com o avanço no desenvolvimento de algoritmo de Machine Learning(aprendizado de máquina), o grupo OpenIA criou uma IA do game DOTA 2, famoso por ser um jogo de estratégia complexo, que é capaz de ganhar de 99.99% dos jogadores.



**(fig 1 - jogo de xadrez entra Garry Kasparov e Deep blue em 1996.)**

A partir do que se foi supra-apresentado, devemos compreender a importância de criar modelos de machine learning para jogos, pois a mesma aplicação serve para testar a eficiência de uma rede neural em relação ao ser humano. Com isso em mente, foi desenvolvido no atual trabalho uma IA com base no algoritmo de aprendizado de máquina SOM(self-organized maps) que é capaz de buscar a vitória em um jogo elaborado e programado em python pelo grupo do projeto.

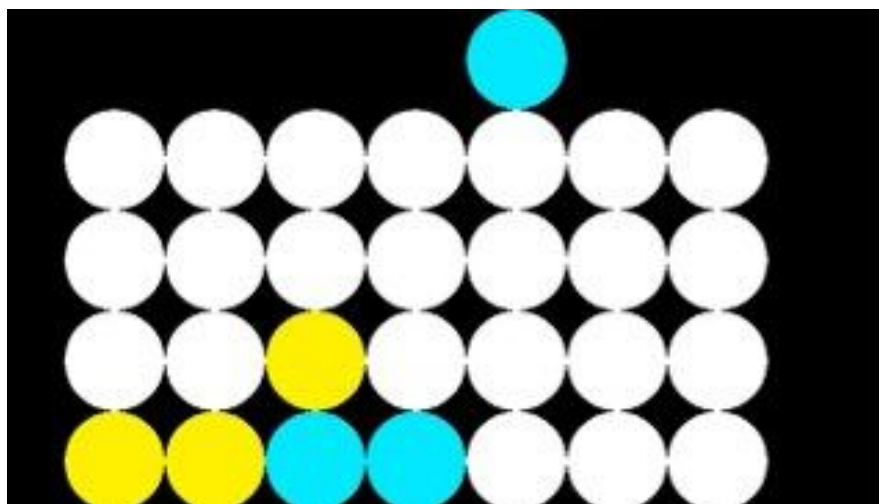
## **Metodologia**

Já que o foco de nosso trabalho é o desenvolvimento de uma inteligência artificial por métodos de machine learning, o jogo desenvolvido foi feito de forma simples, assim, usamos mais tempo na síntese da rede SOM e no tratamento/coleta dos dados. Para uma melhor compreensão do nosso trabalho é necessário explicar as regras e mecânicas do jogo elaborado.

## JOGO

Tendo como base outros jogos clássicos, como o “jogo da velha” e “connect four”, o nosso game possui um tabuleiro com dimensões iguais a 4x7 (4 linhas e 7 colunas), em que as unidades dessa matriz são representados por círculos(bolas) brancas.O jogo foi desenvolvido para funcionar com uma capacidade máxima de dois jogadores em que cada jogador possui seu turno na rodada. Cada espaço do tabuleiro pode ser preenchido por um jogador pelo da vez por uma bola com uma cor respectiva ao indivíduo(jogador 1 tem a cor ciano e o jogador 2 tem a cor amarela), vale salientar que um mesmo espaço não pode ser preenchido duas vezes, ou seja, uma vez de uma cor a unidade do campo permanecerá com esta cor.

Para preencher um espaço no campo, o jogador do turno tem somente a capacidade de escolher a coluna a qual quer jogar e, assim como no “connect four”, a bola sempre se deposita ao fundo do tabuleiro, ou seja, a segunda linha de uma coluna só pode ser ocupada se já houver uma bola na primeira linha dessa mesma coluna.



(fig 2 - demonstração do tabuleiro explicado do jogo criado)

Após estabelecermos todas as mecânicas do jogo necessita-se estabelecermos no código uma condição de vitória que, dessa vez será idêntico ao “jogo da velha”, ou seja, para que um dos jogadores venha a vencer é necessário

preencher três espaços seguidos, sejam esses espaços linhas diagonais, horizontais ou verticais, bolas de uma mesma cor, vence o jogador que forma uma linha de três com as bolas de sua respectiva cor.

A partir de todos esse atributos que o jogo possui decidimos denominá-lo **COLOR PUZZLE**.

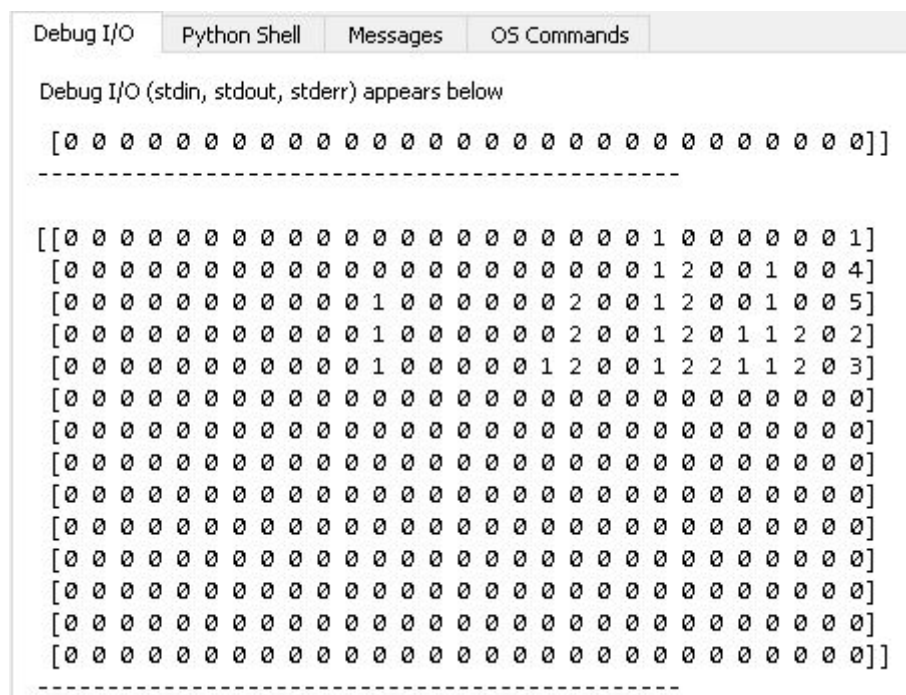
## **COLETA E TRATAMENTO DOS DADOS**

Sabendo que para o desenvolvimento de um modelo de rede neural é necessário de uma boa quantidade de dados e que esses dados sejam fáceis de serem interpretados pela rede. Por causa disso é importante explicar como se deu a coleta de dados e quais das informações do jogo foi recolhida.

Para o processo de coleta de dados foi criado uma versão do jogo especifica para esta funcionalidade, a qual essa versão funciona somente com a existência de dois jogadores humanos, cada um com uma cor no tabuleiro. Nesta versão, em toda rodada se cria duas matrizes(variáveis no código) que armazenam a situação do tabuleiro no turno de cada jogador, uma delas se chama “matrizAmarelo” e a outra “matrizCiano”, nestas matrizes cada item corresponde um espaço real no tabuleiro em que os espaços vazios são denominados de por “0”, os espaços preenchidos pelo jogador da vez é dado como “2” e os espaços do adversários recebem a atribuição de “1”.

Para passar os dados citados acima de uma partida inteira foi modificado o formato da matriz que irá ser passada para o banco de dados, de maneira a resumir todas as matrizes de cada rodada em uma só matriz, para isso os componentes individuais de cada jogada foram vetorizados, ou seja, todo o tabuleiro de cada rodada é colocado em um só vetor a qual os 7 primeiros ítems do vetor representam as colunas da quarta linha do tabuleiro naquela jogada, os 7 seguintes são as colunas da terceira linha e assim por diante. Agora que os dados de cada jogada estão armazenados em um vetor podemos criar uma matriz que concatena todos os dados de todas as jogadas da partida de cada jogador, bastando que cada linha leve um vetor tabuleiro de cada jogada em ordem. Vale salientar que além dessa modificação na maneira de se armazenar os dados do tabuleiro também foi adicionado mais um valor como parâmetro do banco de dados, esse parâmetro é

um inteiro que vai de 0 a 6 e representa a coluna que o jogador da vez realizou ao se deparar com a situação do tabuleiro armazenado nas variáveis “matrizAmarelo” ou “matrizCiano” dependendo de quem é a vez, esse último valor tem grande propósito no momento do treinamento da rede SOM. No fim o input lançado ao data set será uma matriz de dimensões 14x29(14 linhas e 29 colunas), sendo a vigésima nona coluna a posição escolhida pelo jogador da vez e as 14 linhas é quantidade máxima de rodadas permitidas no jogo.



```

Debug I/O Python Shell Messages OS Commands
Debug I/O (stdin, stdout, stderr) appears below

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
-----

[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 1 2 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 0 0 1 1 2]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 0 0 1 2 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
-----

```

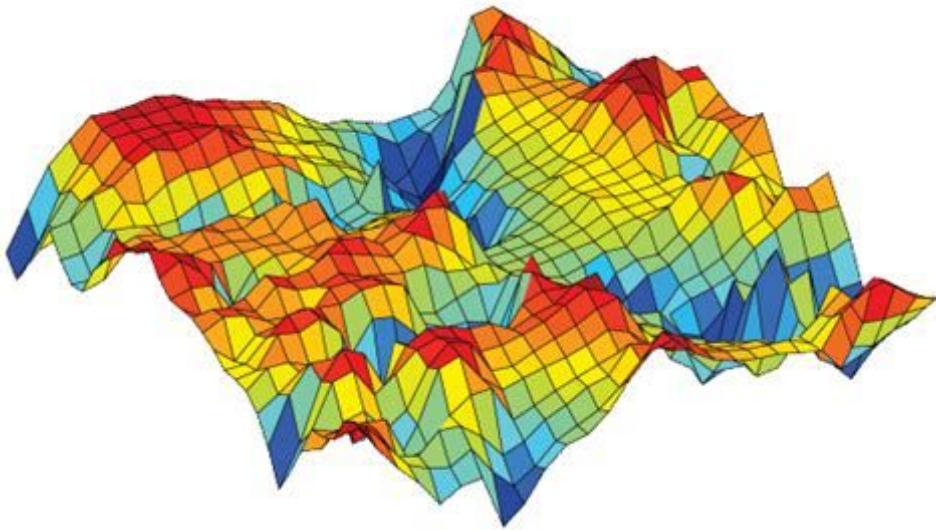
**(fig 3 - matriz de uma partida inteira, a qual cada linha representa a situação do tabuleiro em cada rodada. Percebe-se que depois da quinta linha só há zeros, pois o jogo acabou na quinta rodada.)**

Por fim vem o tratamento desses valores que irão ser lançados no banco de dados. Não serão todas as matrizes que serão armazenados no data set, em nosso caso apenas a matriz de jogadas e as colunas jogadas respectivas as rodadas do jogador vencedor serão adicionadas no banco de dados. Isso se dá pois a IA deve sempre buscar as opções de melhores lances, logo, sua rede neural deve ser construída sobre exemplos de jogadas que resultaram em vitórias.

## Self Organizing Maps (SOM)

No campo do aprendizado de máquina, o Self-Organizing Map desenvolve uma rede neural artificial, a partir de uma aprendizagem não supervisionada, capaz de produzir uma redução dimensional, normalmente para duas, dos dados de entrada.

Um dos maiores diferenciais desse tipo de rede é a preservação das características topológicas dos dados, visto que a distribuição dos neurônios é composta por regiões de transição entre picos e vales.



**(fig 3 - visualização possível do treinamento usando SOM)**

Por conseguinte, fez-se jus utilizar essa rede para analisar a melhor jogada a ser feita pela máquina. Em termos de treinamento da rede, ele funciona a partir de aprendizado competitivo e a distância Euclidiana é utilizada para computar a proximidade de um novo dado ou entre os dados de treinamento. Ou seja, os dados de entrada disputam entre si posições no mapa de saída.

Com os dados coletados, tornou-se possível realizar o treinamento da rede, com os seguintes parâmetros:

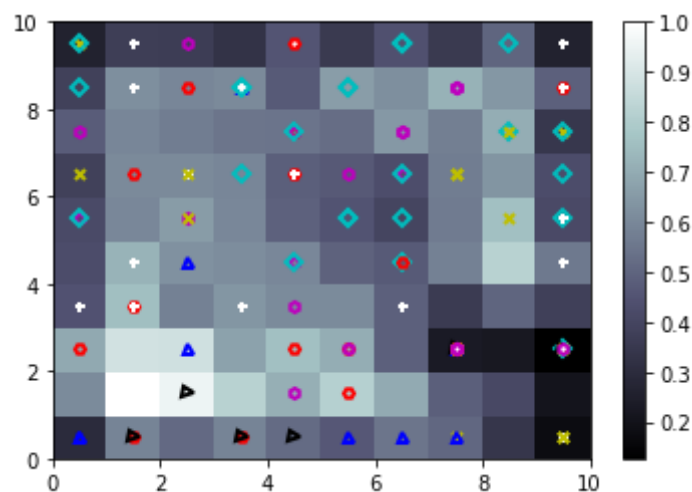
```
som = MiniSom(x = xtam, y = ytam, input_len = 28, sigma = 1.0, learning_rate = 0.4)
```

```
som.random_weights_init(X)
som.train_random(data = X, num_iteration = 2500)
```

O tamanho da rede foi de 10x10, com um sigma padrão de 1.0, que diz respeito ao raio de distanciamento entre os vizinhos da rede e a taxa de aprendizado como 0.4, dado que a diminuição desse valor não era tão significativo no resultado.

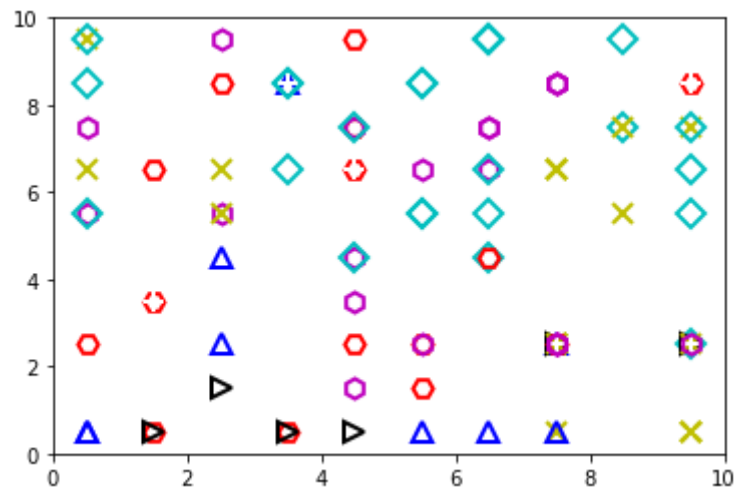
## Resultados

O treinamento supracitado deu como resultado os seguintes mapas de neurônios:

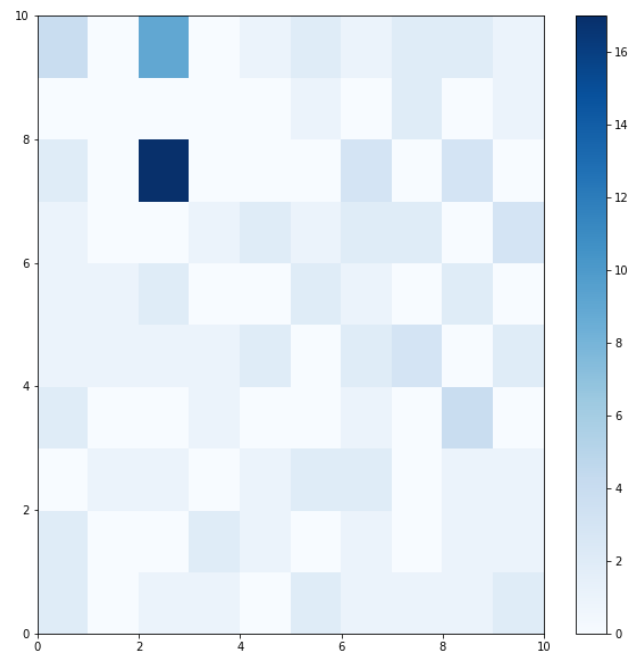


**(fig 4 -Mapa dos neurônios com disparos de classes e características topológicas)**

O mapa da figura 4 é uma boa maneira de visualizar os neurônios, foi-se possível analisar que um mesmo quadrante se relacionava com mais de uma classe. No intuito de melhorar essa visualização, têm-se mais três mapas:

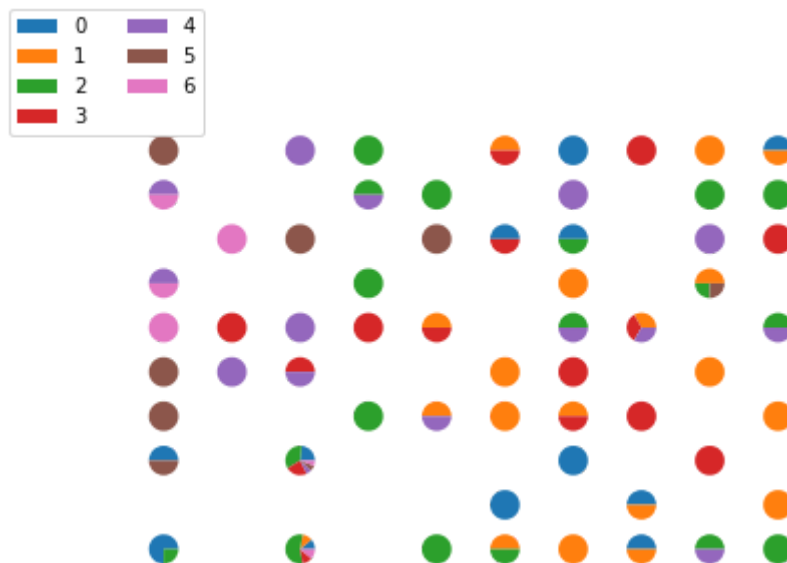


(fig 5 -Mapa dos neurônios com disparos de classes)



(fig 5 -Mapa dos neurônios com características topológicas)





(fig 6 -Mapa dos neurônios com disparos de classes)

Os mapas acima se completam quando se trata de analisar o desempenho da distribuição das classes entre os neurônios. O próximo passo foi testar essas classes utilizando a seguinte estratégia: neurônios com mais de uma classe disparada, deve-se sortear delas para ser a vencedora, caso aquele quadrante seja tido como vencedor durante o jogo. A demonstração do desempenho inicial da rede pode ser vista no seguinte link: <https://youtu.be/JXoAxB4Qln8>

## Conclusão

De acordo com o mostrado no vídeo acima, o treinamento teve um resultado mediano, podendo-se pontuar sua preferência por vencer do jogador de forma vertical, além de deter jogadas esquisitas, como a de sobrepor a do humano, caso ele tente vencer na vertical. No geral, para que o desempenho seja melhorado, pode-se vislumbrar que o aumento da base de dados deve impulsionar sua inteligência.

O jogo desenvolvido possui um problema estrutural, que é necessitar apenas de 3 casas para vencer. Após algumas pesquisas, percebemos que o ideal seria utilizar uma condição de 4 círculos alinhados como critério de vitória, além de adequar o tabuleiro para 8x8, o que tornaria a disputa mais justa.

Em suma, o projeto foi muito proveitoso, visto que pudemos exercitar a rede SOM, vista em sala de aula, além de desenvolver um jogo simples para aplicá-la. Ademais, a experiência na determinação de estratégias para treinar a rede de acordo com o cenário criado foi de grande relevância, porque em um contexto real e profissional, nem sempre esses detalhes estão predefinidos.

### **Links de referência**

1. <https://heartbeat.fritz.ai/introduction-to-self-organizing-maps-soms-98e88b568f5d>
2. <https://towardsdatascience.com/self-organizing-maps-ff5853a118d4>