

ADO #03 - Self-Organizing Maps

VERSÃO 03 - 04/05/2017

- Changelog (V2 - 20/04): detalhes sobre RGB adicionados.
- Changelog (V3 - 04/05): gráfico da Figura I estava incorreto. Thanks to Gabriel Lins.

Neste trabalho, criaremos a nossa segunda rede neural denominada Self-Organizing Maps (SOM). Ao contrário da rede MLP, a rede SOM é uma rede não-supervisionada e, portanto, não requer as informações sobre classes.

1. OBJETIVO DESTE TRABALHO

Este trabalho como objetivo estudar o mapeamento visual e a capacidade de agrupamento e geração de clusters de uma rede SOM.

2. REDE NEURAL SELF-ORGANIZING MAPS (SOM)

2.1 EXEMPLO DE USO E CONSTANTES

A seguir, segue um exemplo de uso da Self-Organizing Maps.

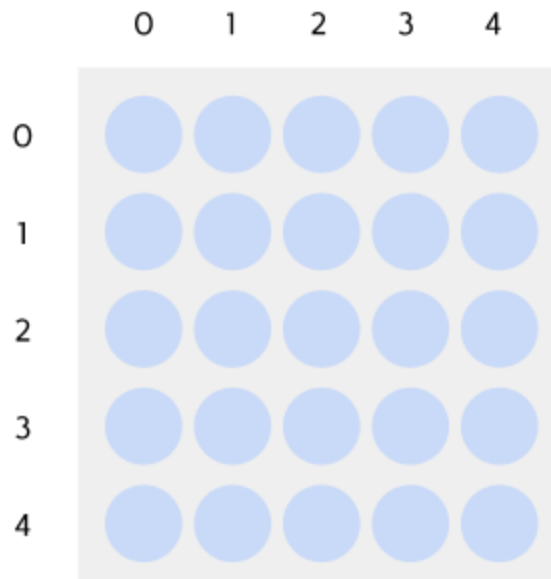
Para isso, criei uma tabela com conjunto de dados fictício (TABELA I), onde a informação da classe não será usada. Queremos ter 02 grupos, então precisamos de um mapa SOM com, pelo menos, $2 \times 10 = 20$ neurônios (veja a aula sobre SOM).

Criamos, então, um mapa SOM com $5 \times 5 = 25$ neurônios (veja a MATRIZ I).

TABELA I - Conjunto de dados fictício.

SEPAL LENGTH	SEPAL WIDTH	PETAL LENGTH	PETAL WIDTH	CLASS
0,194444444	0,625	0,101694915	0,208333333	1
0,222222222	0,541666667	0,118644068	0,166666667	1
0,805555556	0,666666667	0,86440678	1	2
0,555555556	0,541666667	0,847457627	1	2

MATRIZ I - Mapa SOM com 25 neurônios e seus respectivos índices linha x coluna.



Temos que criar uma matriz de pesos entre cada neurônio e cada atributo de entrada. Como temos 04 atributos de entrada e 25 neurônios, teríamos uma **matriz de pesos W com $4 \times 25 = 100$ posições**. A inicialização dos pesos depende dos valores de entrada. Se seus atributos tiverem entre 0.0 e 1.0, então os pesos podem ser inicializados entre 0.0 e 1.0 também. Porém, se seus dados estiverem entre 0 e 255, os pesos também deverão ter esse intervalo.

Primeiro, vamos listar todas as constantes do programa para o nosso caso:

Const.	Descrição	Valor
σ_0	Desvio padrão. O valor inicial de σ pode ser o tamanho do "raio" da rede de neural (considerando que ela tenha 2 dimensões) No nosso caso, observe que o raio é 2.	2
T_1	Sua fórmula é $\frac{1000}{\log \sigma_0}$. Como sabemos que, para nosso exemplo, σ_0 vale 2, temos $\tau_1 = \frac{1000}{\log \sigma_0} = \frac{1000}{\log 2} = \frac{1000}{0.301} = 3321.92$.	3321.92
η_0	Taxa de aprendizado inicial para atualização dos pesos. Na literatura é utilizado 0.1.	0.1
T_2	Constante que define o decaimento da função da taxa de aprendizado. Por definição, o valor padrão é 1000.	1000

A partir dessas constantes, agora podemos calcular as funções que dependem de n , que é o número de iterações. Temos:

- $\eta(n) = \eta_0 e^{(-n / \tau_2)} = 0.1 e^{(-n / 1000)}$

- $\sigma(n) = \sigma_0 e^{(-n / \tau_1)} = 2 e^{(-n / 3321.92)}$
- $h_{j,i}(n) = e^{(-d_{ij}^2 / 2\sigma^2(n))}$

Quer saber se essas funções são crescentes ou decrescentes? Veja a seção 2.6.

2.2 TREINAMENTO DA REDE

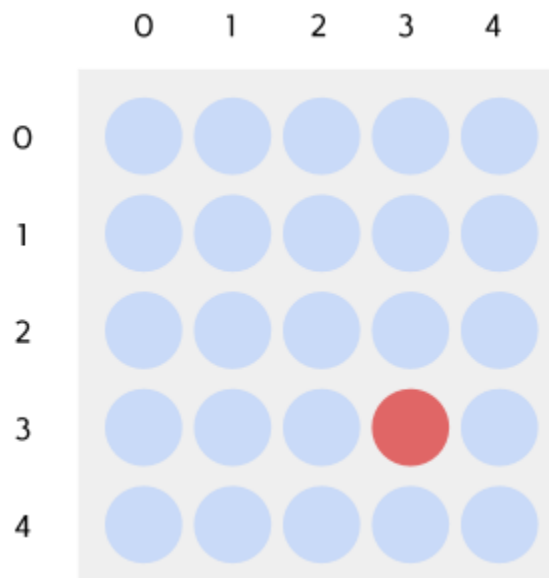
Como não há Cross-Validation, o treinamento da rede é composto por todas as instâncias do conjunto de dados que serão apresentadas uma-a-uma para a rede..

Lembre-se que o objetivo aqui é ajustar os pesos para obtermos agrupamentos.

2.3 NEURÔNIO VENCEDOR

Para encontrar o neurônio vencedor, basta encontrar o neurônio com a menor a distância euclidiana entre seus respectivos pesos w_i e a instância x .

Vamos supor que para a primeira entrada da TABELA 1 encontrou como neurônio o mostrado em vermelho no índice [3,3], destacado abaixo.



Uma vez encontrado o neurônio, atualizar os pesos w_i do **neurônio vencedor** i com a entrada x . Observe que o **neurônio vencedor** i tem conexão com cada atributo da entrada x . Todos eles devem ser atualizados da seguinte maneira:

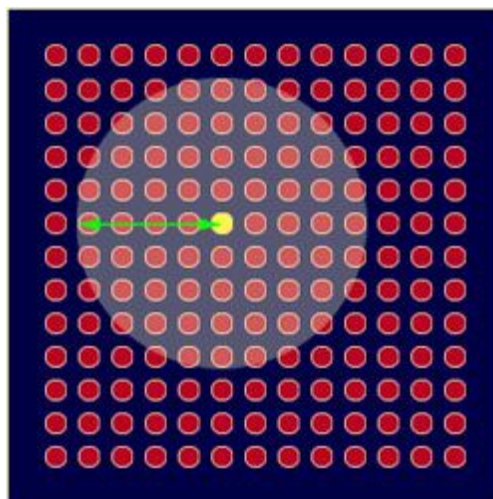
$$w_i(n+1) = w_i(n) + \eta(n)h_{i,i}(n)(x - w_i)$$

Para o neurônio vencedor, temos que $d_{i,i} = 0$, logo $h_{i,i}(n) = 1$. Temos que:

$$w_i(n+1) = w_i(n) + \eta(n)(x - w_i)$$

2.4 VIZINHOS DO NEURÔNIO VENCEDOR

Uma forma de considerar os vizinhos do neurônio vencedor seria criar um raio a partir do neurônio vencedor (destacado em amarelo na figura abaixo). Observe que o raio é igual a 4 neste caso.



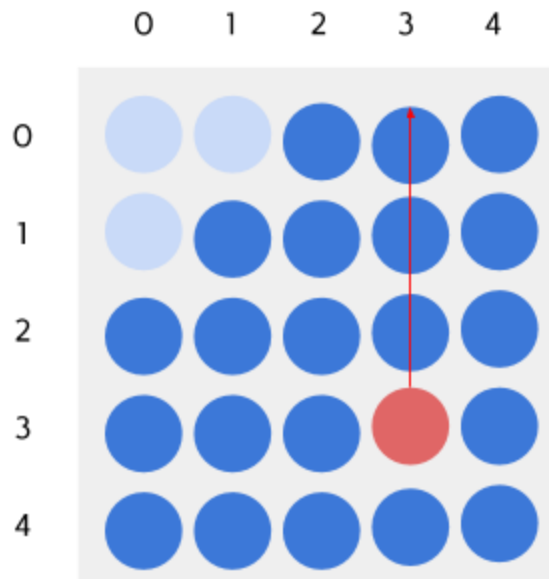
Fonte: <http://www.cs.us.es/~fsancho/?e=136>

Uma vez que a fórmula $h_{j,i}(n)$ já considera a distância entre os neurônios vizinhos até o neurônio vencedor, podemos definir um raio R para considerar os vizinhos.

Const.	Descrição	Valor
R	Raio de vizinhança do neurônio vencedor	3

Assim sendo, basta aplicar a fórmula de $h_{j,i}(n)$ para todos os neurônios com $d_{j,i} \leq R$.

Logo, em nosso Mapa SOM 5x5, temos os destacados abaixo.



Para atualizar os pesos do neurônio vizinho j até o neurônio vencedor i , você deve calcular a distância euclidiana entre j e i . Você pode utilizar os índices da MATRIZ I como posições de cada neurônio.

$$d_{j,i} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

Feito isso, calcular a função $h_{j,i}(n)$ para iteração n :

$$h_{j,i}(n) = e^{(-d_{ij}^2 / 2\sigma^2(n))}$$

Com isso, você pode atualizar os pesos w_j do neurônio j com a entrada x da seguinte maneira:

$$w_j(n+1) = w_j(n) + \eta(n)h_{ij}(n)(x - w_j)$$

2.5 CONVERGÊNCIA DO ALGORITMO

O número de iterações para atingir a convergência deve ser, pelo menos, 500 vezes a quantidade de neurônios na rede neural. No nosso caso, $25 * 500 = 12500$ iterações.

Cada iteração corresponde à apresentação de uma instância à rede. Portanto, haverá repetição de instâncias (e o ideal é que haja repetição).

2.6 ANÁLISE DAS FUNÇÕES PARAMETRIZADAS PELO NÚMERO DE ITERAÇÕES

Antes de prosseguir, você deve estar se perguntando se as funções $\eta(n)$, $\sigma(n)$ e $h_{j,i}(n)$ decaem com o tempo.

A fim de analisar isso, resolvi projetar seus gráficos nesta seção.

- $\eta(n) = \eta_0 e^{(-n / \tau_2)} = 0.1 e^{(-n / 1000)}$

Na Figura 1 abaixo temos a função $\eta(n)$ no eixo Y e o número de iterações no eixo X. Perceba a redução gradual de $\eta(n)$.

O que isso significa?

Quanto maior o número de iterações, menor é a taxa de aprendizado e, conseqüentemente, menor também será a taxa de atualização dos pesos da rede.

FIGURA 1 - Função $\eta(n)$ pelo número de iterações n .



Agora vamos olhar para a função $\sigma(n)$.

- $\sigma(n) = \sigma_0 e^{(-n / \tau_1)} = 2 e^{(-n / 3321.92)}$

Na Figura 2 abaixo, perceba o mesmo comportamento. Isso significa que, quanto maior o número da iteração, menor é a força de atualização dos vizinhos do neurônio vencedor.

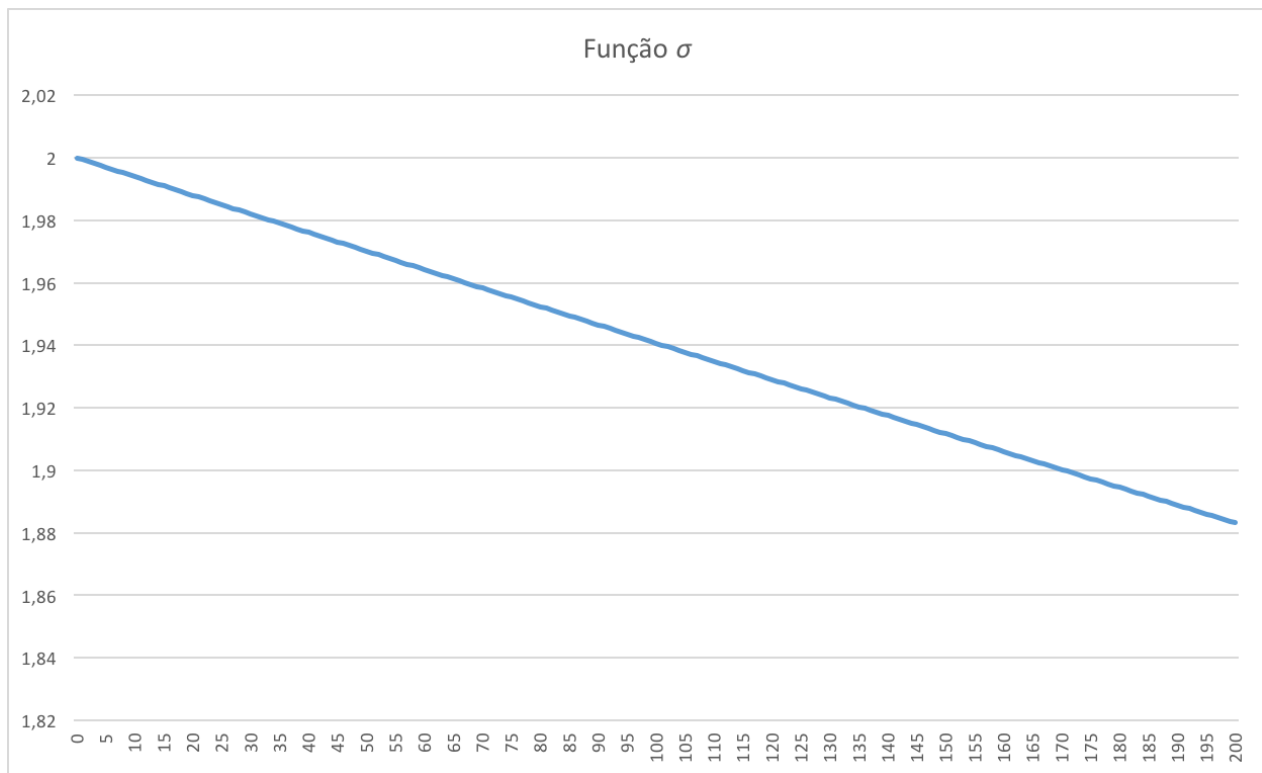


FIGURA 2 - Função $\sigma(n)$ pelo número de iterações n .

Por fim, vamos analisar a função $h_{j,i}(n)$.

- $h_{j,i}(n) = e^{(-d_{ij}^2 / 2\sigma^2(n))}$

Para analisar essa função, precisamos observá-la sobre dois pontos diferentes. O primeiro deles é fixar a distância (por exemplo, $d = 1$) e entender como o número de iterações é trabalhado sobre vizinhos com esse valor de distância do neurônio vencedor.

Veja na Figura 3 que quando fixamos $d = 1$ temos um decaimento da função $h_{j,i}(n)$ com o número de iterações.

Isso nos mostra que a força de atualização dos vizinhos vai diminuindo com o passar das iterações.

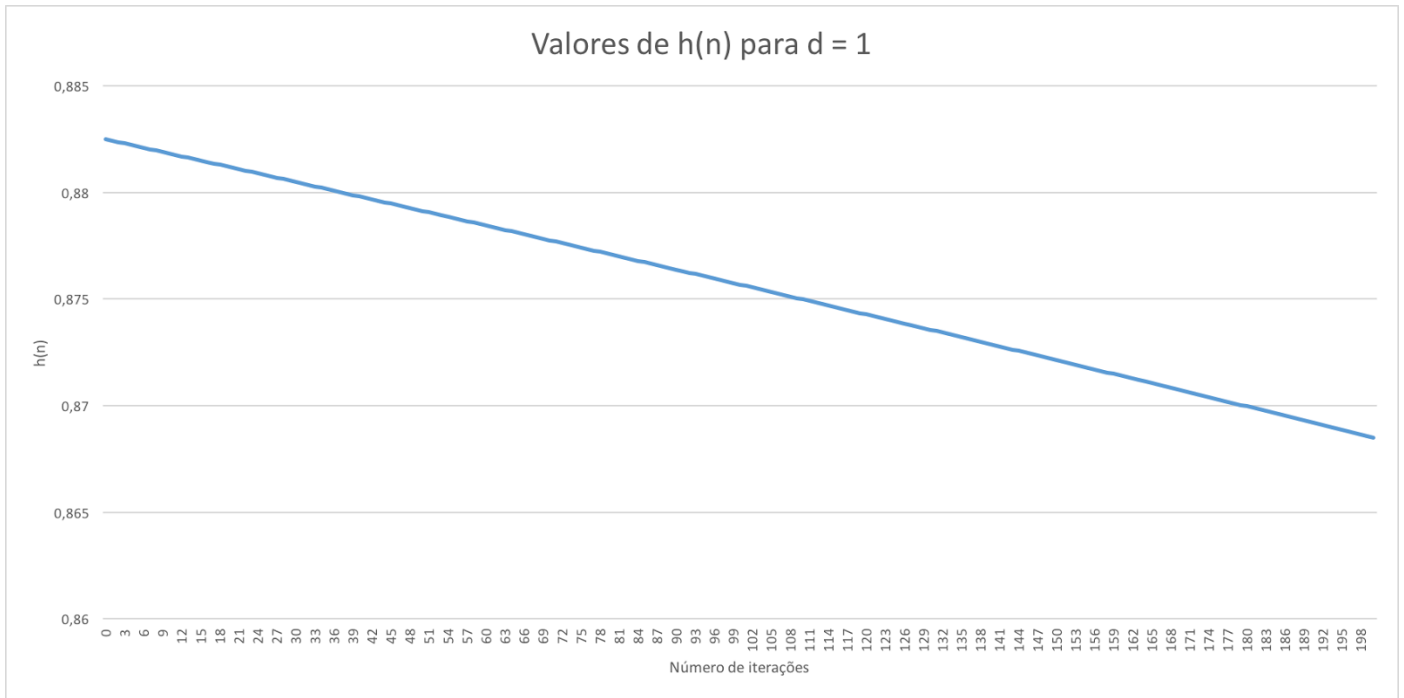


FIGURA 3 - Função $h_{j,i}(n)$ pelo número de iterações n .

O segundo ponto que precisamos observar é fixar uma determinada iteração (por exemplo, $n = 0$) e entender como a função trabalha com os vizinhos do neurônio vencedor variando-se os valores de $d_{i,j}$

- $h_{j,i}(0) = e^{(-d_{i,j}^2 / 2\sigma^2(0))} = e^{(-d_{i,j}^2 / 2(2)^2)} = e^{(-d_{i,j}^2 / 8)}$

Veja na Figura 4 que $h_{j,i}(0)$ tem valores menores quando aumentamos a distância $d_{i,j}$. Isso significa que neurônios muito distantes do neurônio vencedor quase não sofrem alterações.

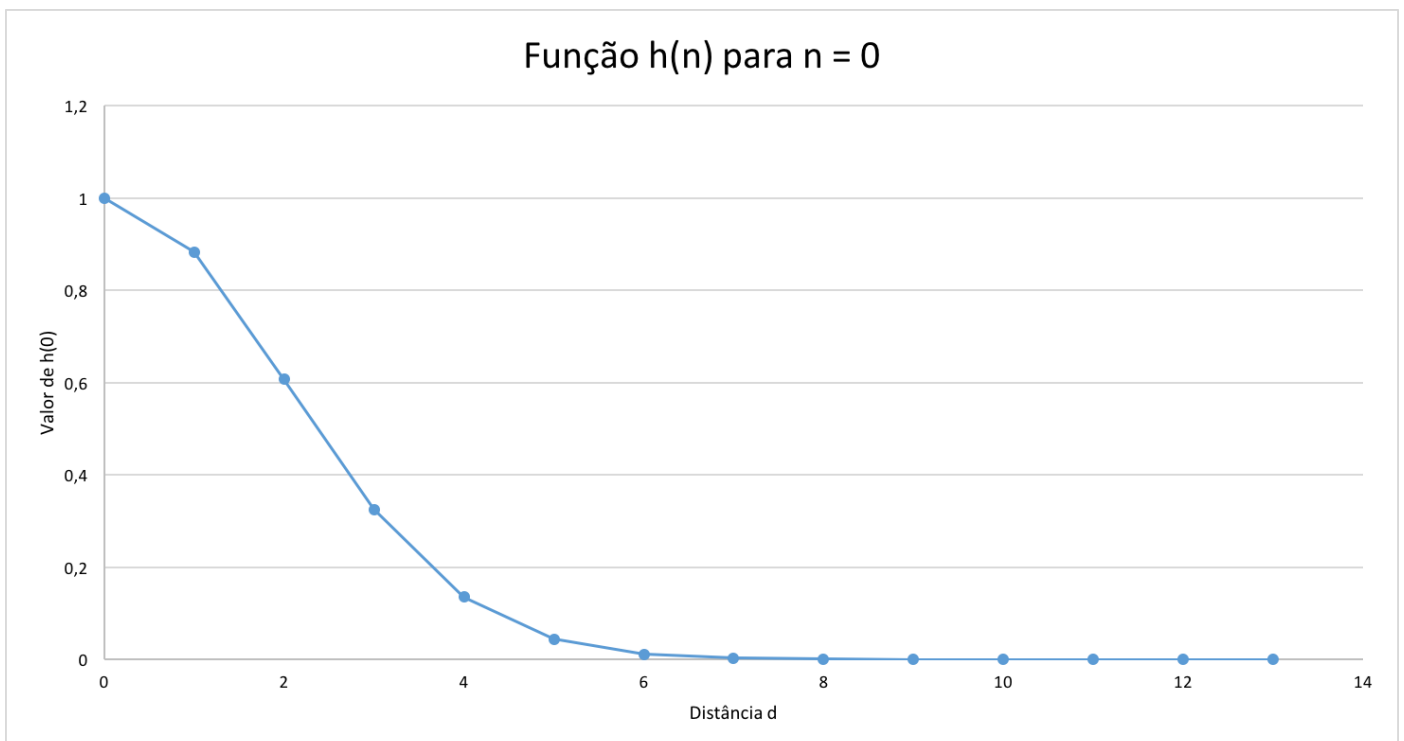


FIGURA 4 - Função $h_{j,i}(0)$ pela distância do neurônio vizinho até o neurônio vencedor.

3. PROJETO

Essa seção descreve como o projeto deverá ser desenvolvido e entregue.

3.1. PARTE 1

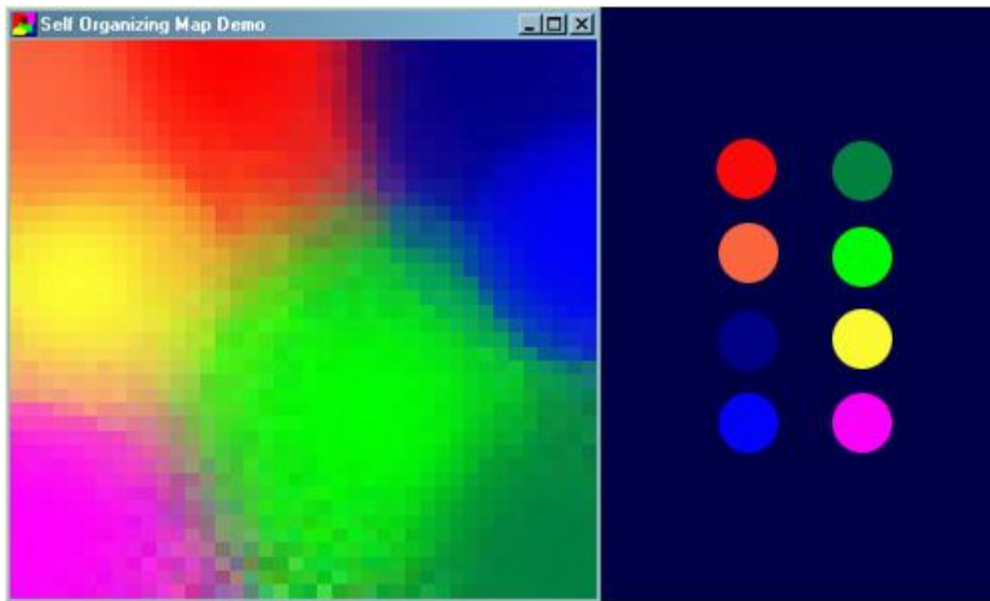


FIGURA 5 - Mapa SOM treinado com 08 cores.

O objetivo dessa primeira parte é gerarmos um mapa de cores semelhante ao mostrado na Figura 05. Para isso, você e sua equipe deverão implementar uma rede neural SOM em uma linguagem que desejarem.

Além disso, vocês deverão implementar uma **interface gráfica** semelhante ao da Figura 05, onde seja possível representar o Mapa SOM. Cada quadriculado desse mapa gráfico corresponde à um neurônio da rede SOM. Lembre-se que a quantidade de linhas e colunas é parametrizável (no geral, o mapa será um quadrado).

A maioria das linguagens de programação usadas na ADO #01 possuem bibliotecas de interface gráfica. O Java, por exemplo, possui o Swing e Python possui o Plotly.

Veja um exemplo de funcionamento neste site:

http://www.aforgenet.com/framework/samples/neuro_som.html

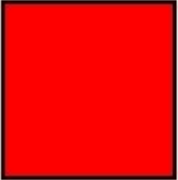
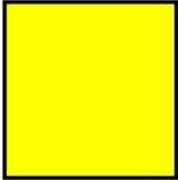
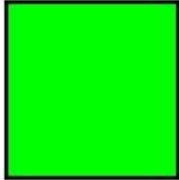
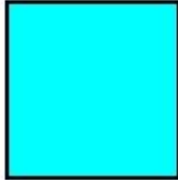
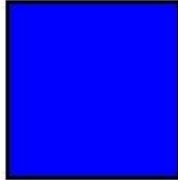
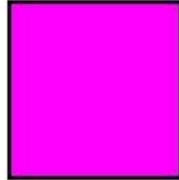
3.1.1 INICIALIZAÇÃO DOS PESOS

Os pesos deverão ser inicializados com valores aleatórios **inteiros** entre 0 e 255.

3.1.2 DADOS DE ENTRADA

Como entrada, você terá 03 atributos. Cada atributo corresponde a um valor RGB.

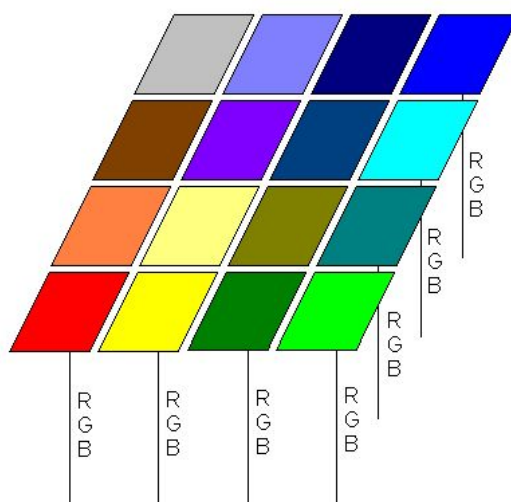
Haverão apenas 06 entradas, mostradas abaixo:

					
R = 255 G = 0 B = 0	R = 255 G = 255 B = 0	R = 0 G = 255 B = 0	R = 0 G = 255 B = 255	R = 0 G = 0 B = 255	R = 255 G = 0 B = 255

Perceba que isso irá definir a quantidade de grupos que iremos ter na rede. Cada neurônio da rede terá 03 pesos, correspondente à cada atributo de entrada. Vamos chamá-los de pesos R, G e B.

3.1.3 APRESENTAÇÃO

Em cada quadriculado de sua interface gráfica, você deverá exibir uma cor. **Essa cor corresponde ao RGB dos pesos R, G e B.** Abaixo temos um exemplo de mapa 4x4 onde o valor do quadriculado corresponde ao RGB dos pesos.



Você deverá mostrar o estado inicial do mapa em sua interface gráfica e, com o decorrer das iterações, fazer animações para mostrar as modificações que ocorrem no mapa até atingir sua convergência.

Exemplo: <https://www.youtube.com/watch?v=-6a7LATC-9g>

Se possível, faça igual ao desenvolvedor da rede do vídeo e deixe parametrizável:

- Taxa de aprendizado inicial da rede (e não apenas 0.1)
- Tamanho do mapa
- Quantidade de iterações da sua rede
- Quantidade de cores desejadas para treino

Observação: no momento da apresentação, pode ocorrer de algum peso ficar com **valor negativo**. Neste caso, para o valor do RGB deve ser considerado **zero**.

3.2. PARTE 2

Usaremos a interface gráfica feita na Parte 01 de uma maneira diferente. As cores agora não serão os pesos.

3.2.1 CONJUNTOS DE DADOS DE ENTRADA

Na ADO #01 você e sua equipe fizeram a normalização (escala de 0 a 1) dos seguintes conjuntos de dados do UCI Machine Learning Repository:

- Iris - 3 classes
- Wine - 3 classes
- Breast Cancer Wisconsin (*o campo Id pode ser removido*) - 2 classes

Para evitar um trabalho adicional, vamos usar esses conjuntos de dados neste trabalho.

3.2.2 PRÉ-PROCESSAMENTO

Você deverá utilizar seu conjunto de dados já pré-processado (isto é, normalizado).

Antes de utilizar os dados nesse trabalho, é importante **EMBARALHAR** o conjunto todo. Caso contrário, pode ser que instâncias de uma mesma classe fiquem agrupadas e pode atrapalhar o treinamento.

Esse último detalhe não foi dito na ADO #01, porém uma função **deve ser criada para tal**.

3.2.3 INICIALIZAÇÃO DOS PESOS

Os pesos deverão ser inicializados com valores aleatórios **inteiros** entre 0.0 e 1.0.

3.2.4 CORES ÀS CLASSES

Para cada classe do conjunto de dados, atribua uma cor RGB.

O Íris, por exemplo, terá 03 cores, assim como o Wine.

3.2.5 TREINAMENTO

Desconsidere a classe e faça o treinamento da rede utilizando os conceitos explanados nos slides de aula e nos itens 1 e 2 deste Relatório.

Use as constantes definidas no item 2 deste relatório, incluindo o raio R.

3.2.6 APRESENTAÇÃO

Após o término de treinamento e convergência da rede, faça o seguinte:

1. Para cada instância P do conjunto de dados:
 - a. Encontre o neurônio vencedor na rede neural;
 - b. Defina a cor RGB do neurônio vencedor como a cor da classe da instância P.
 - c. Caso o neurônio já esteja colorido, sobrescreva com a cor da nova instância.

4. VÍDEO EXPLICATIVO

Você deverá elaborar um vídeo da tela explicando:

1. **Algoritmo:** você deverá mostrar como está estruturado e como funciona cada parte do seu código-fonte.
2. **Execução:** você deverá mostrar a Parte 1 funcionando e na Parte 2 mostrar a sua execução para os 03 conjuntos de dados, explicando o que está acontecendo.

Nessa parte, você deverá utilizar um software gravador de tela. Os sistemas Windows 10 e macOS possuem softwares gravadores de tela nativos.

5. RELATÓRIO EM PDF

Um relatório deve ser confeccionado em PDF, preferencialmente gerado por meio do LaTeX.

Deve ser dividido em duas categorias:

1. PARTE 1

- a. Apresentar resultados das cores RGB

2. PARTE 2

- a. Apresentar resultados do conjunto Iris
- b. Apresentar resultados do conjunto Wine
- c. Apresentar resultados do conjunto Breast Cancer Wisconsin

ENTREGA E ENVIO NO BLACKBOARD

- O projeto pode ser feito em equipes de 01 a 03 integrantes.
- Os código-fontes devem ser compactados e enviados no Blackboard.
- O vídeo pode ficar com um tamanho um tanto grande. Por isso, você pode fazer o upload no YouTube como "não listado" e enviar o link via Blackboard.
- O relatório deve ser entregue em PDF (preferencialmente feito em LaTeX).

As dúvidas podem ser enviadas no e-mail willian.yhonda@sp.senac.br

BOM TRABALHO!