



Redes Auto-organizáveis de Kohonen

Gustavo Stein Santos Mattos Araújo dos Santos
Humberto da Silva Neto



Agenda

❖ Redes Auto-organizáveis de Kohonen

- ❖ Introdução
- ❖ Características
- ❖ Funcionamento

❖ Aplicações

❖ Implementação

- ❖ SOMTF
- ❖ SOMPY
- ❖ Testes

❖ Problema proposto e resultados

The background of the slide is a complex network diagram. It consists of numerous small circular nodes, some of which are solid grey and others are hollow with a grey outline. These nodes are interconnected by a web of thin, light-grey lines, creating a dense, interconnected pattern that resembles a neural network or a social network graph. The overall color scheme is light blue and grey.

Redes Auto-organizáveis de Kohonen

Introdução

- ❖ Proposta por Teuvo Kohonen, 1982
- ❖ Inspiração biológica no córtex cerebral
- ❖ Aprendizado competitivo
- ❖ Estrutura reticulada



Características

Redes Neurais usuais

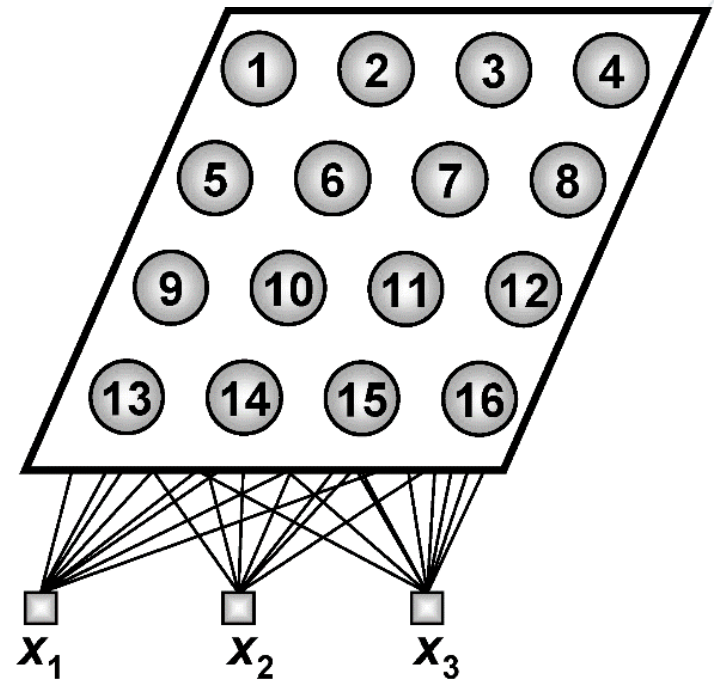
- ❖ ~~Série de camadas~~
- ❖ ~~Correção por erro~~
- ❖ ~~Aprendizado supervisionado~~

SOM

- ❖ Rede 2D*
- ❖ Por competição
- ❖ Aprendizado não supervisionado

Características

- ❖ Adapta-se à organização topológica de um dataset, permitindo a visualização de cluster's em potencial
- ❖ Lida melhor com variáveis contínuas



Características

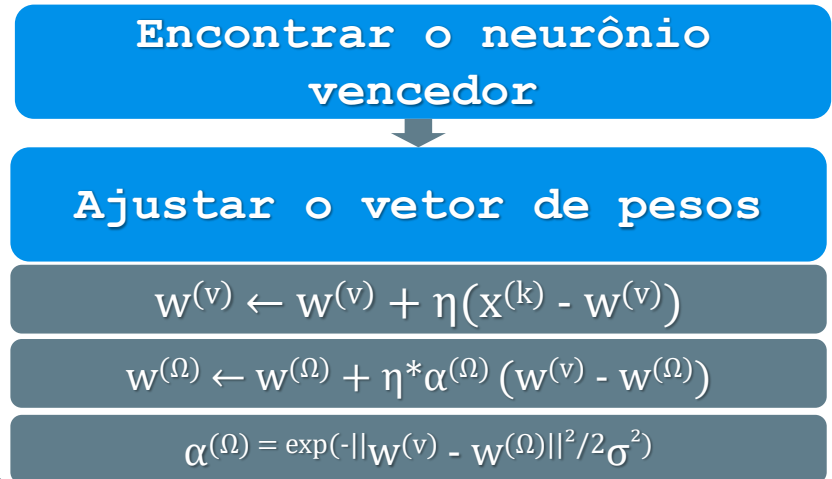
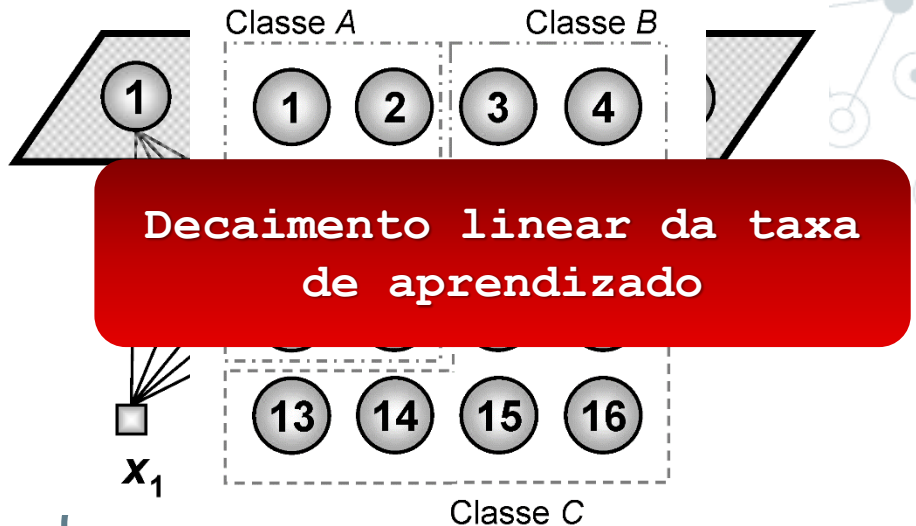
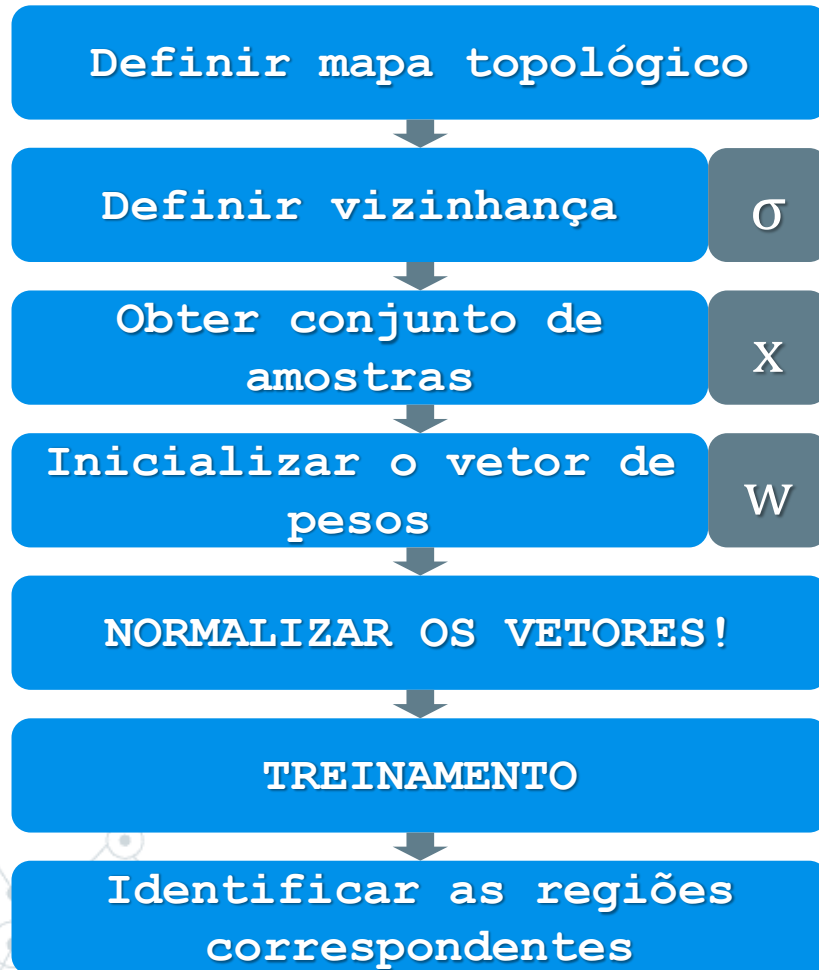
Vantagens

- ❖ Representação visual (Redução da dimensionalidade)
- ❖ Organização de dados de forma a tornar inferências sobre os mesmos possível
- ❖ Grande número de variáveis

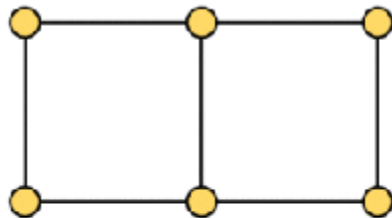
Desvantagens

- ❖ Não lida bem com variáveis categóricas
- ❖ Computacionalmente dispendiosa
- ❖ Soluções nem sempre facilmente correlacionáveis

Funcionamento: treinamento

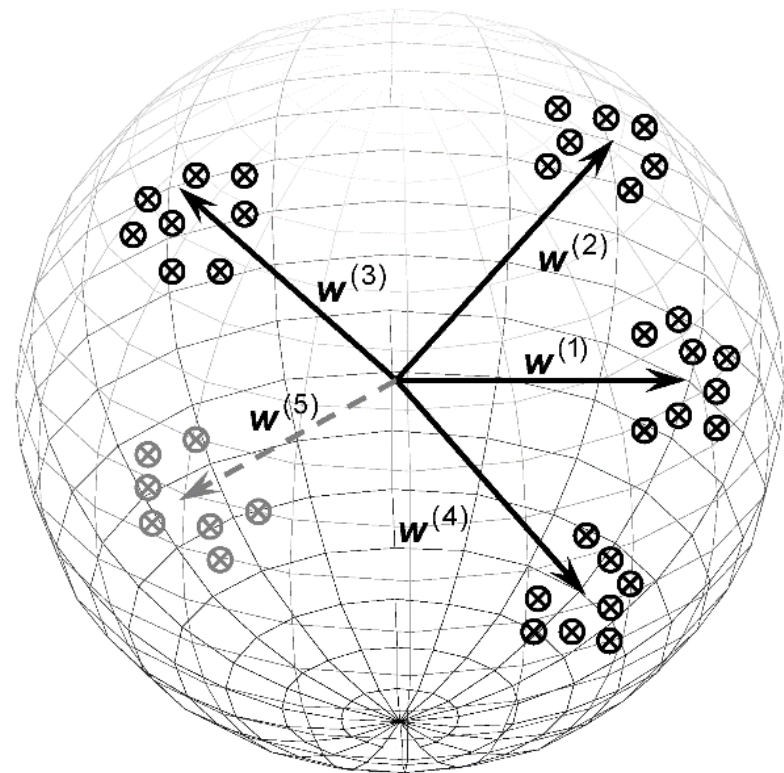
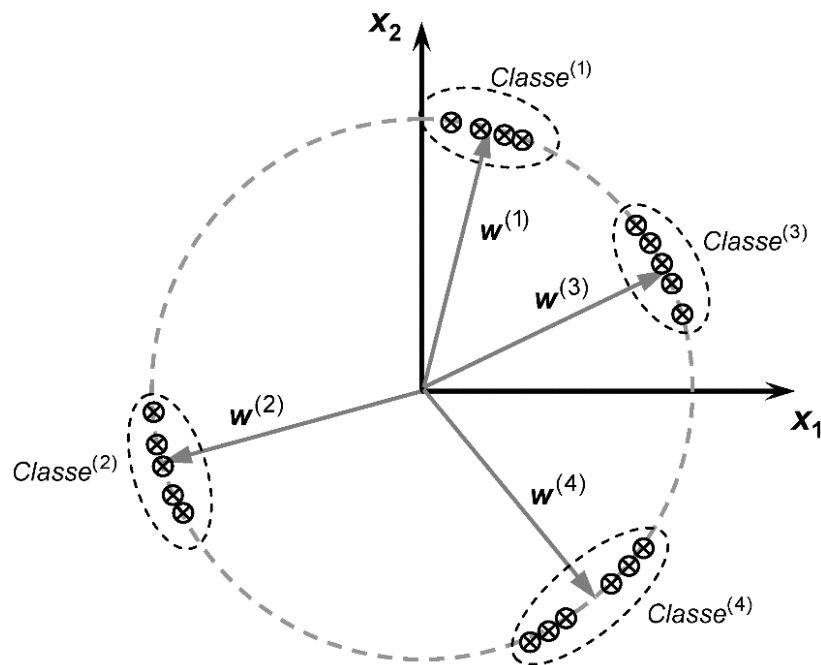


Funcionamento: treinamento



Step 0: Position neurons (orange) in data space.

Funcionamento: agrupamento



Funcionamento: algoritmos

Início {Algoritmo Kohonen – Fase de Treinamento}

- <1> Definir o mapa topológico da rede;
- <2> Montar os conjuntos de vizinhança $\{\Omega^{(j)}\}$;
- <3> Iniciar o vetor de pesos de cada neurônio $\{w^{(j)}\}$ considerando os valores das n_1 primeiras amostras de treinamento;
- <4> Obter o conjunto de amostras de treinamento $\{x^{(k)}\}$;
- <5> Normalizar os vetores de amostras e de pesos;
- <6> Especificar a taxa de aprendizagem $\{\eta\}$;
- <7> Iniciar o contador de número de épocas $\{época \leftarrow 0\}$;
- <8> Repetir as instruções:
 - <8.1> Para todas as amostras de treinamento $\{x^{(k)}\}$, fazer:
 - <8.1.1> Calcular as distâncias euclidianas entre $x^{(k)}$ e $w^{(j)}$, conforme a expressão (8.1);
 - <8.1.2> Declarar como vencedor o neurônio j que contenha a menor distância euclidiana:
$$vencedor = \arg \min_j \{\|x^{(k)} - w^{(j)}\|\}$$
 - <8.1.3> Ajustar o vetor de pesos do vencedor, conforme a regra 1 da expressão (8.4);
 - <8.1.4> Ajustar o vetor de pesos dos neurônios vizinhos ao vencedor, definidos em $\Omega^{(j)}$, conforme regra 2 da expressão (8.4) ou (8.5);
 - <8.1.5> Normalizar o vetor de pesos que foi ajustado na instrução anterior;
 - <8.2> $época \leftarrow época + 1$;
- Até que: não haja mudanças significativas nos vetores de pesos;
- <9> Analisar o mapa visando extração de características;
- <10> Identificar regiões que possibilitem a definição de classes.

Fim {Algoritmo Kohonen – Fase de Treinamento}

Início {Algoritmo Kohonen – Fase de Operação}

- <1> Apresentar a amostra $\{x\}$ a ser classificada e normalizar;
- <2> Assumir os vetores de pesos $\{w^{(j)}\}$ já ajustados durante a fase de treinamento;
- <3> Executar as seguintes instruções:
 - <3.1> Calcular as distâncias euclidianas entre x e $w^{(j)}$, conforme a expressão (8.1);
 - <3.2> Declarar como vencedor o neurônio j que contenha a menor distância euclidiana;
 - <3.3> Localizar o neurônio vencedor dentro do mapa auto-organizável;
 - <3.4> Associar a amostra à classe que foi identificada a partir da confecção do mapa de contexto;
- <4> Disponibilizar a eventual classe em que a amostra foi associada.

Fim {Algoritmo Kohonen – Fase de Operação}

The background of the slide is a light gray network pattern. It consists of numerous small circles, some of which are solid gray and others are hollow with a gray outline. These circles are interconnected by a web of thin, light gray lines, creating a complex, organic structure that resembles a molecular or digital network.

Aplicações

Classificação de países por condição socioeconômica

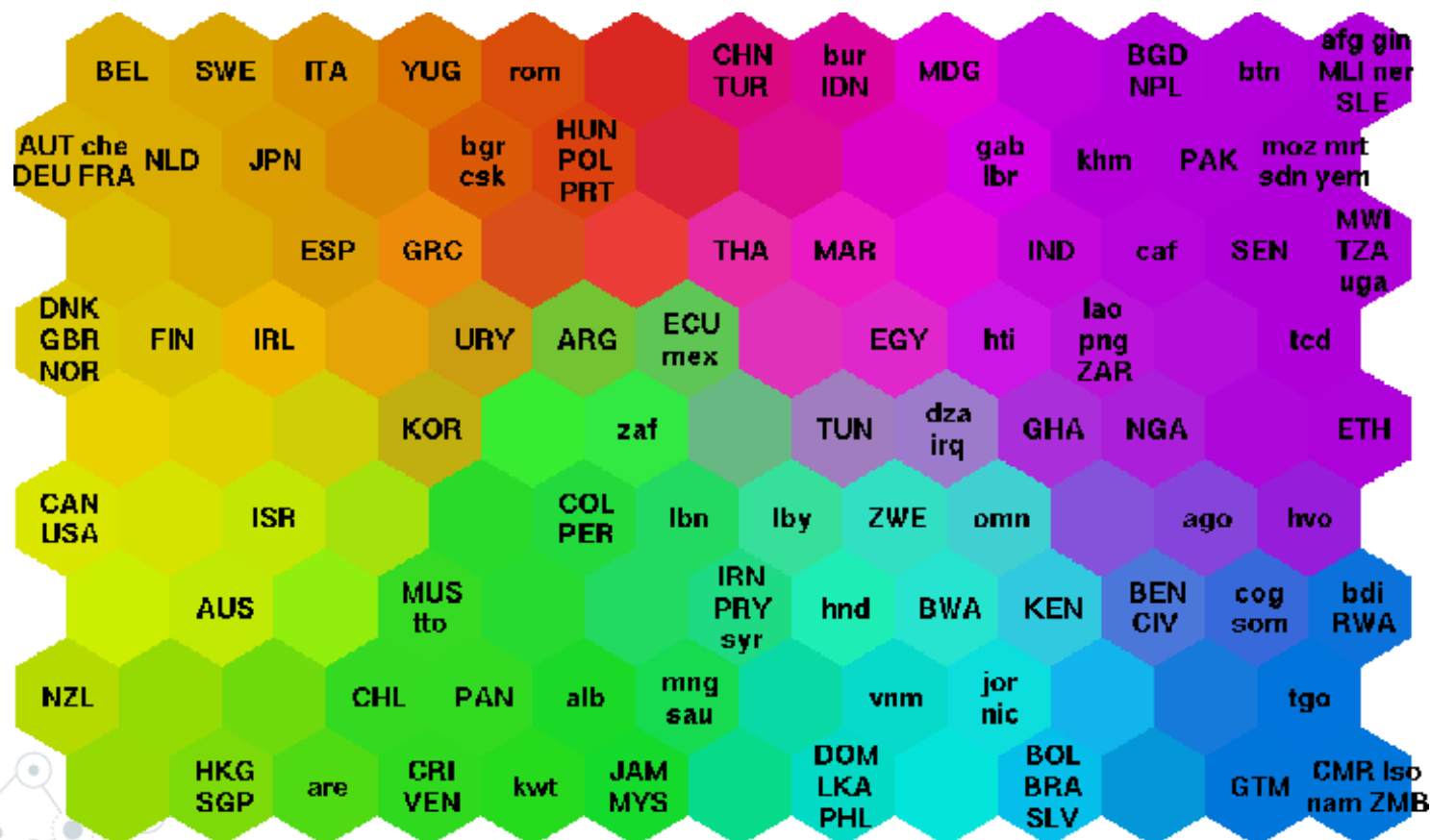


A world map illustrating the socioeconomic classification of countries. The map uses a color-coded system to categorize nations based on their socioeconomic status. The colors used include yellow, green, blue, orange, red, purple, and grey. The map shows a clear division between developed and developing regions, with developed countries generally colored in yellow, green, or blue, and developing countries in orange, red, or purple. The map is set against a black background, and the colors are used to distinguish between different socioeconomic groups across the globe.

13



Classificação de países por condição socioeconômica



Similaridade em Textos

- ❖ 80% dos textos obtidos em datamining não são estruturados ou catalogados
- ❖ Combinação de PNL com SOM's para sintetização e classificação de textos

```
title 15 Ultramar 64 versus 42 versus 50 DATELINE 28
blah 14 Sterling 18 loss 33 cent 25 cent 24
Blah 14 loss 8 profit 32 DATELINE 20 div 24
TITLE 14 Z 8 cent 19 share 18 record 18

title 16 loss 57 versus 50 coffee 10
blah 16 versus 30 DATELINE 10 quota 8
Blah 16 Net 13 cent 9 delegate 8
TITLE 16 cent 11 Sales 8 price 7

title 20 correction 22 Oper 40 Oper 44 dividend 22
blah 19 read 15 loss 37 versus 39 declare 12
Blah 19 correct 11 versus 32 net 20 split 11
TITLE 19 paragraph 10 cent 17 cent 18 split 11

bond 14 franc 36 earnings 10 share 16
issue 10 issue 5 dollar 9 offering 10
percent 8 bond 4 quarter 8 prefer 8
manager 7 issue 4 report 6 stock 6

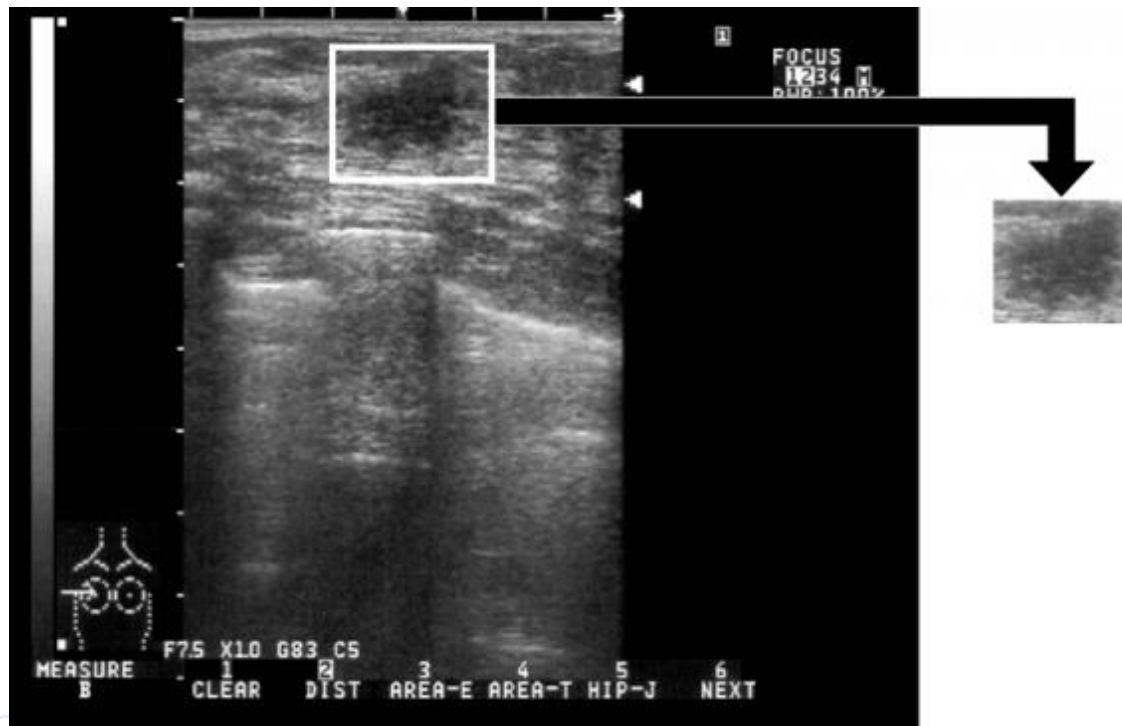
bank 17 percent 16 sale 21 acquire 6 offer 9
Sterling 7 rise 5 car 9 acquisition 6 share 7
loan 4 year 5 percent 5 merger 5 stake 6
rate 4 rose 5 year 5 Inc 4 group 4

U 4 plant 7 unit 6 trade 10
Reagan 3 strike 6 venture 4 exchange 7
trade 3 ton 6 Inc 3 future 6
Japan 3 gold 5 agreement 3 stock 4

tonne 22 oil 9 contract 10 president 14 debenture 14
wheat 6 barrel 8 system 4 officer 11 debt 9
sugar 4 reserve 6 computer 3 chairman 9 subordinate 8
corn 3 OPEC 4 order 3 resign 7 offering 6
```

Diagnóstico de Cancer de Mama

- ❖ A intensidade e textura de uma anomalia pode ser usada para classificar a mesma como maligna ou benigna



The background of the slide is a light gray network pattern. It consists of numerous small circles, some of which are solid gray and others are hollow with a gray outline. These circles are interconnected by a web of thin, light gray lines, creating a complex, organic structure that resembles a molecular or neural network.

Implementação

SOMTF

```
somtf.py x
1 import tensorflow as tf
2 import numpy as np
3
4 class SOM(object):
5     """
6     2-D Self-Organizing Map with Gaussian Neighbourhood function and linearly decreasing learning rate.
7     """
8     # To check if the SOM has been trained
9     _trained = False
10
11 def __init__(self, m, n, dim, n_iterations = 100, alpha=.1, sigma=None, decay_in_time = True):
12     """
13     Initializes all necessary components of the TensorFlow Graph
14
15     Args:
16         m: map width
17         n: map height
18         dim: dimension of the input vector
19         n_iterations: number of iterations (epochs) - Default value is 100
20         alpha: initial value of the learning rate - Default value is 0.1
21         sigma: initial value of neighborhood radius - Default value is max(m, n)/2
22         decay_in_time: decay the learning rate value through time
23     """
24     # Assign input variables
25     self._m = m
26     self._n = n
27     self._n_iterations = int(n_iterations)
28
29     alpha = float(alpha)
30     if sigma is None:
31         sigma = max(m,n)/2.0
32     else:
33         sigma = float(sigma)
34
35     if decay_in_time is not True:
36         decay_in_time = False
37
38     # TensorFlow Graph
```

https://github.com/hsneto/redes_neurais/blob/master/neural_networks/somtf.py

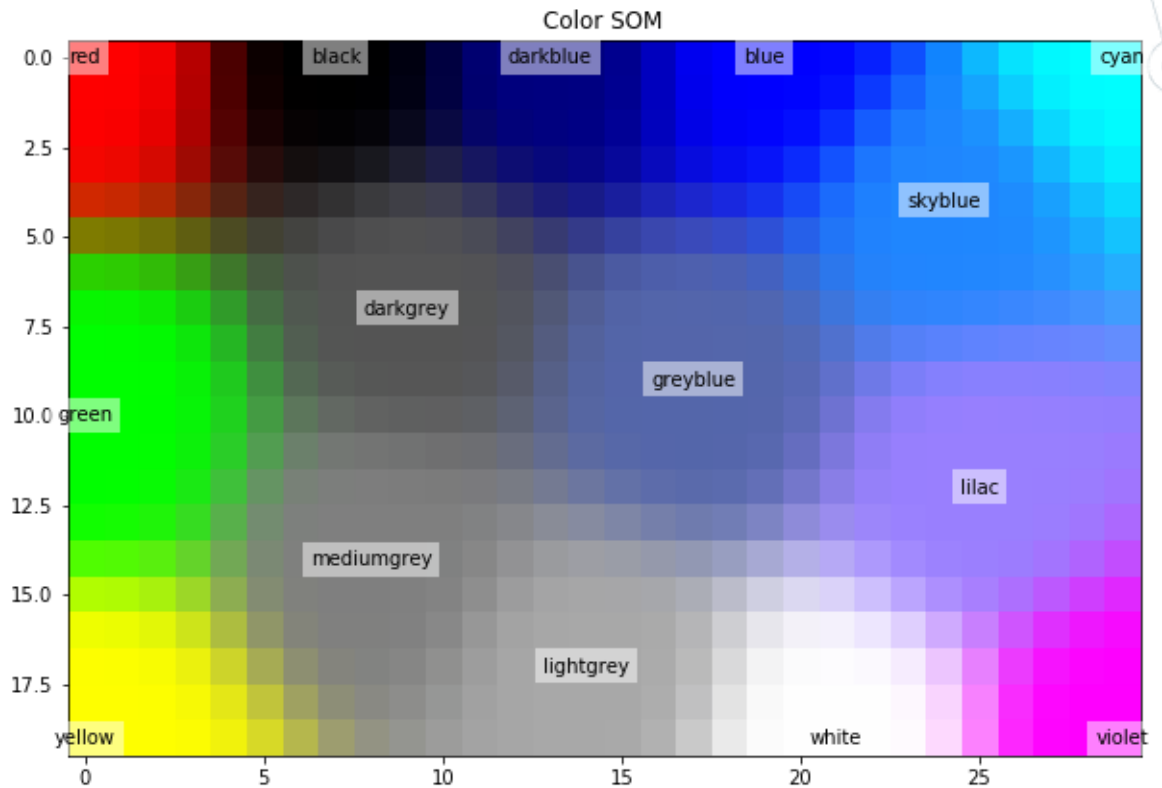
SOMPY

```
sompy.py x
1 import numpy as np
2
3 class SOM(object):
4     ...
5     2-D Self-Organizing Map with Gaussian Neighbourhood function and linearly decreasing learning rate.
6     ...
7     # To check if the SOM has been trained
8     _trained = False
9
10 def __init__(self, m, n, dim, n_iterations = 100, alpha=.1, sigma=None, decay_in_time = True):
11     ...
12     Initializes all necessary components of the TensorFlow Graph
13
14     Args:
15         m: map width
16         n: map height
17         dim: dimension of the input vector
18         n_iterations: number of iterations (epochs) - Default value is 100
19         alpha: initial value of the learning rate - Default value is 0.1
20         sigma: initial value of neighborhood radius - Default value is max(m, n)/2
21         decay_in_time: decay the learning rate value through time
22     ...
23     # Assign input variables
24     self.m = m
25     self.n = n
26     self.dim = dim
27     self.n_iterations = int(n_iterations)
28
29     self.alpha = float(alpha)
30     if sigma is None:
31         self._sigma = max(m,n)/2.0
32     else:
33         self._sigma = float(sigma)
34
35     self._decay_in_time = True
36     if decay_in_time is not True:
37         self._decay_in_time = False
```

https://github.com/hsneto/redes_neurais/blob/master/neural_networks/sompy.py

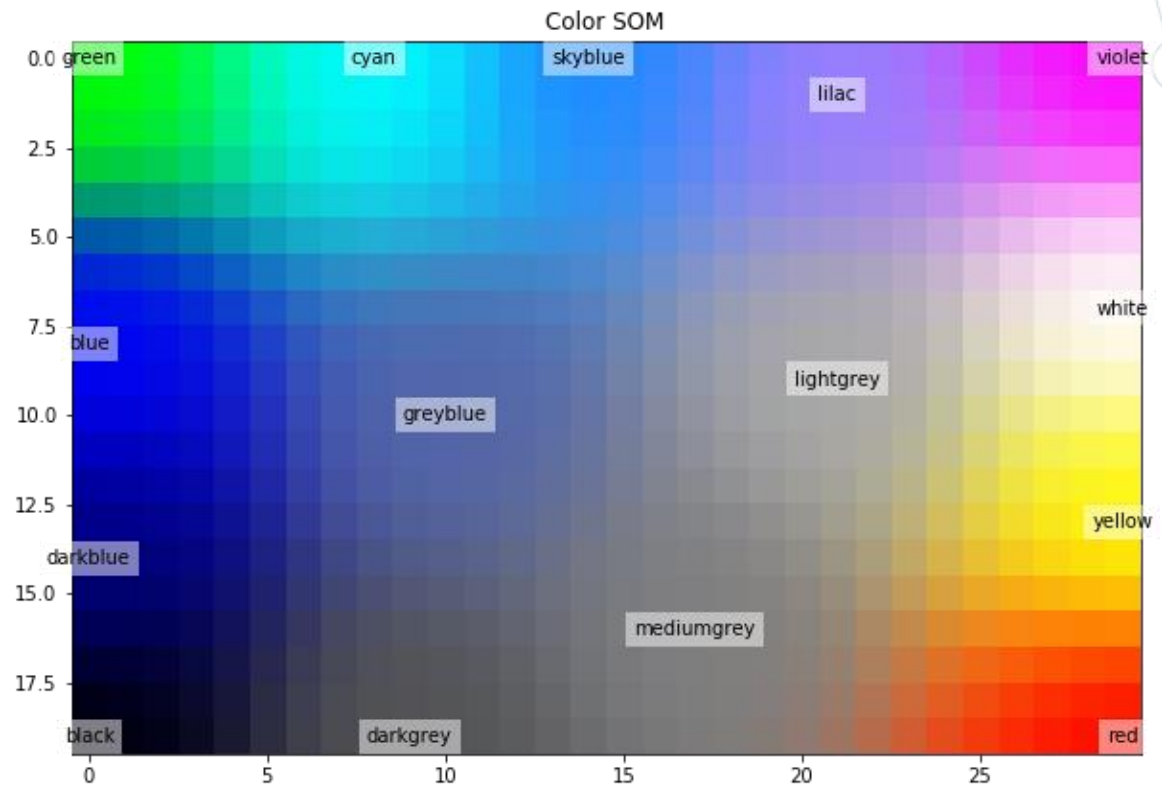
Testes

- ❖ $m = 20$
- ❖ $n = 30$
- ❖ $\text{dim} = 3$
- ❖ $n_iterations = 400$
- ❖ $\alpha = 0.5$
- ❖ $\sigma = \text{None}$
- ❖ $\text{decay_in_time} = \text{True}$



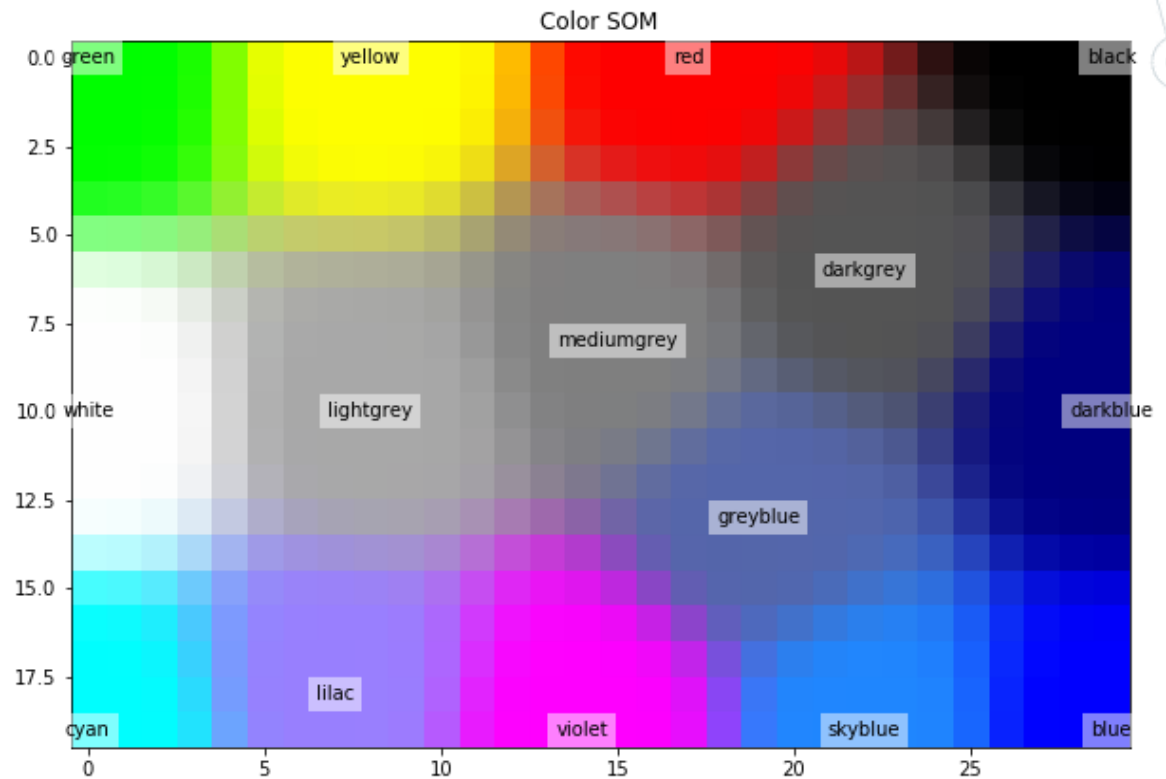
Testes

- ❖ $m = 20$
- ❖ $n = 30$
- ❖ $\text{dim} = 3$
- ❖ $n_iterations = 400$
- ❖ **$\alpha = 0.1$**
- ❖ $\text{sigma} = \text{None}$
- ❖ $\text{decay_in_time} = \text{True}$



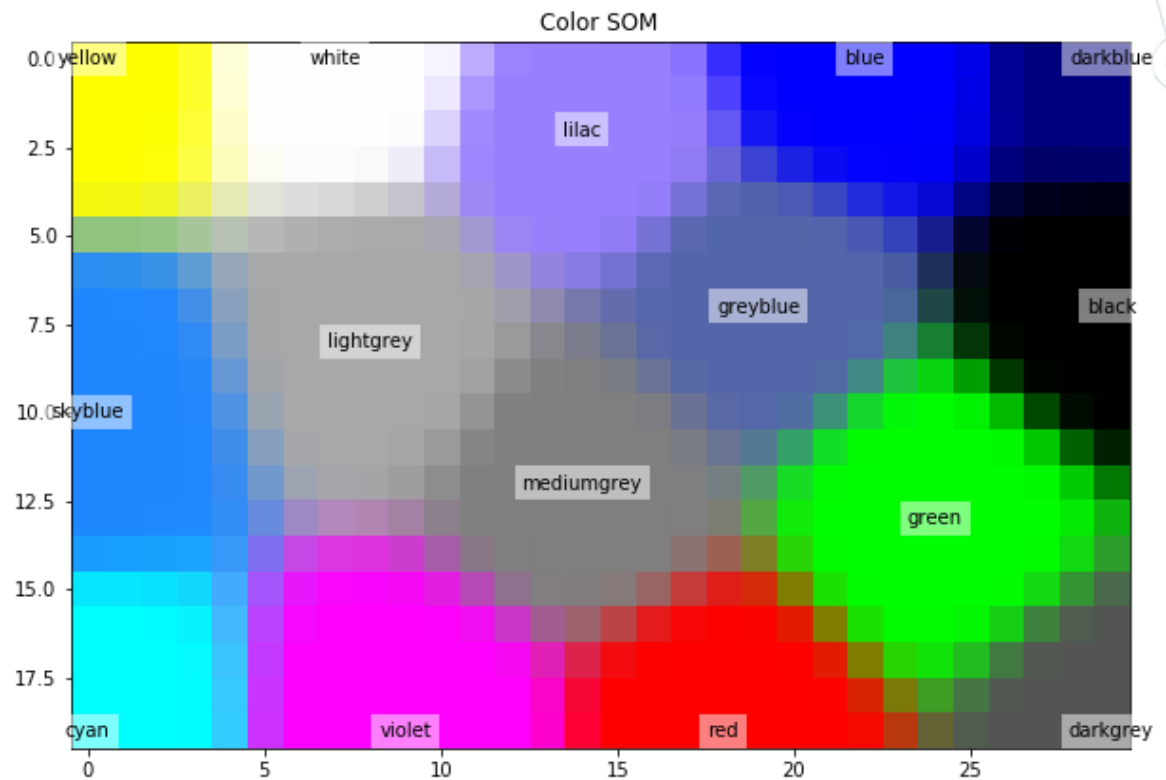
Testes

- ❖ $m = 20$
- ❖ $n = 30$
- ❖ $\text{dim} = 3$
- ❖ $n_iterations = 400$
- ❖ **$\alpha = 1.0$**
- ❖ $\text{sigma} = \text{None}$
- ❖ $\text{decay_in_time} = \text{True}$



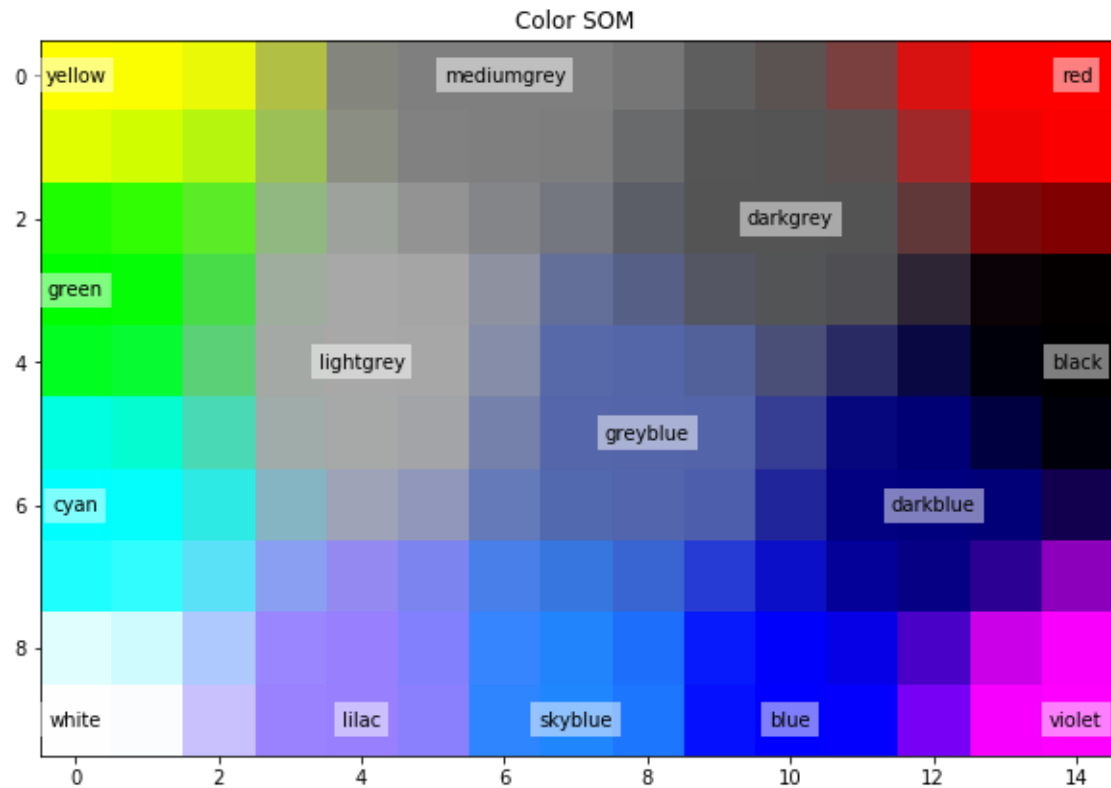
Testes

- ❖ $m = 20$
- ❖ $n = 30$
- ❖ $\text{dim} = 3$
- ❖ $n_iterations = 400$
- ❖ $\alpha = 0.5$
- ❖ **$\sigma = 5$**
- ❖ $\text{decay_in_time} = \text{True}$



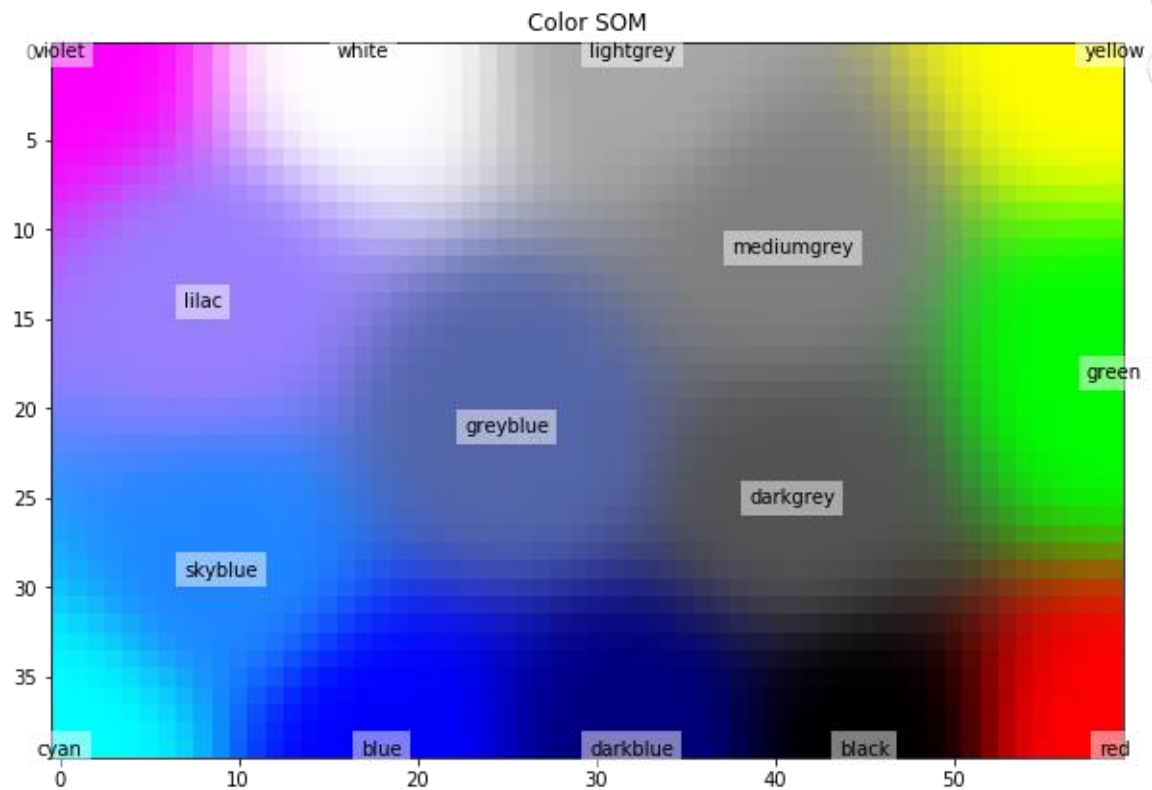
Testes

- ❖ $m = 10$
- ❖ $n = 15$
- ❖ $\text{dim} = 3$
- ❖ $n_iterations = 400$
- ❖ $\alpha = 0.5$
- ❖ $\sigma = \text{None}$
- ❖ $\text{decay_in_time} = \text{True}$



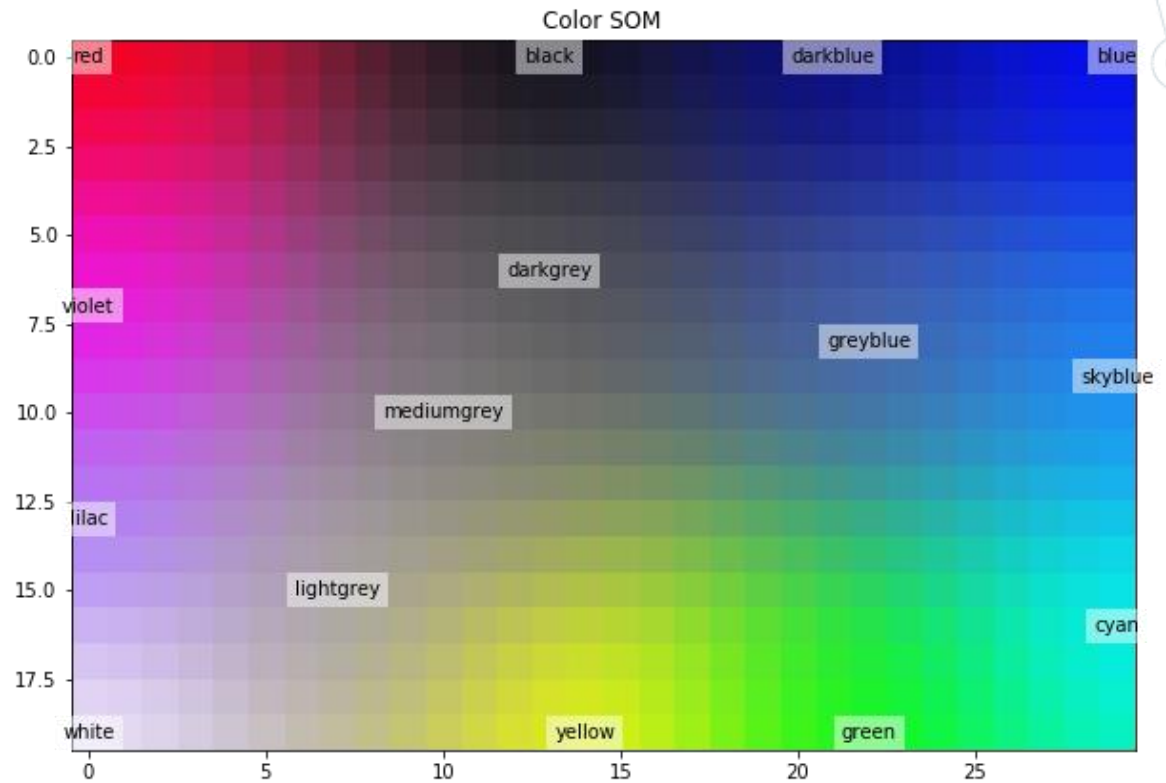
Testes

- ❖ **m = 40**
- ❖ **n = 60**
- ❖ **dim = 3**
- ❖ **n_interations = 400**
- ❖ **alpha = 0.5**
- ❖ **sigma = None**
- ❖ **decay_in_time = True**



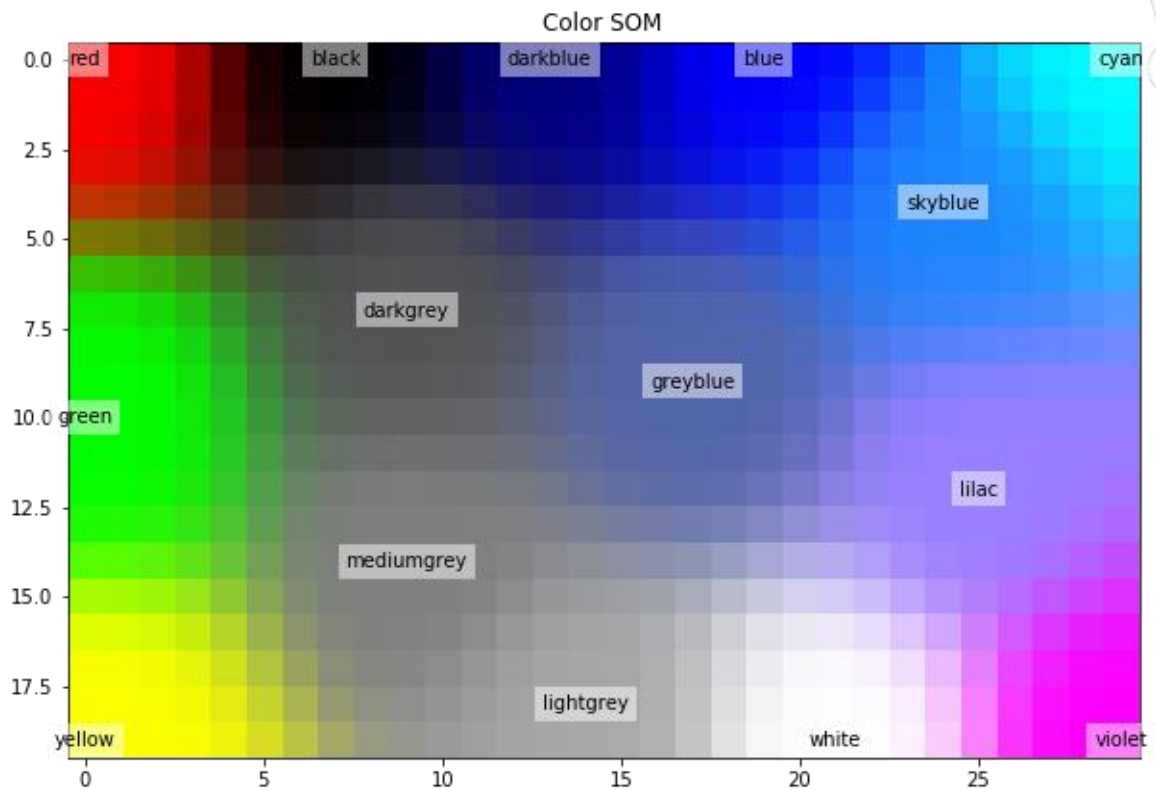
Testes

- ❖ $m = 20$
- ❖ $n = 30$
- ❖ $\text{dim} = 3$
- ❖ **$n_iterations = 100$**
- ❖ **$\alpha = 0.1$**
- ❖ $\text{sigma} = \text{None}$
- ❖ $\text{decay_in_time} = \text{True}$



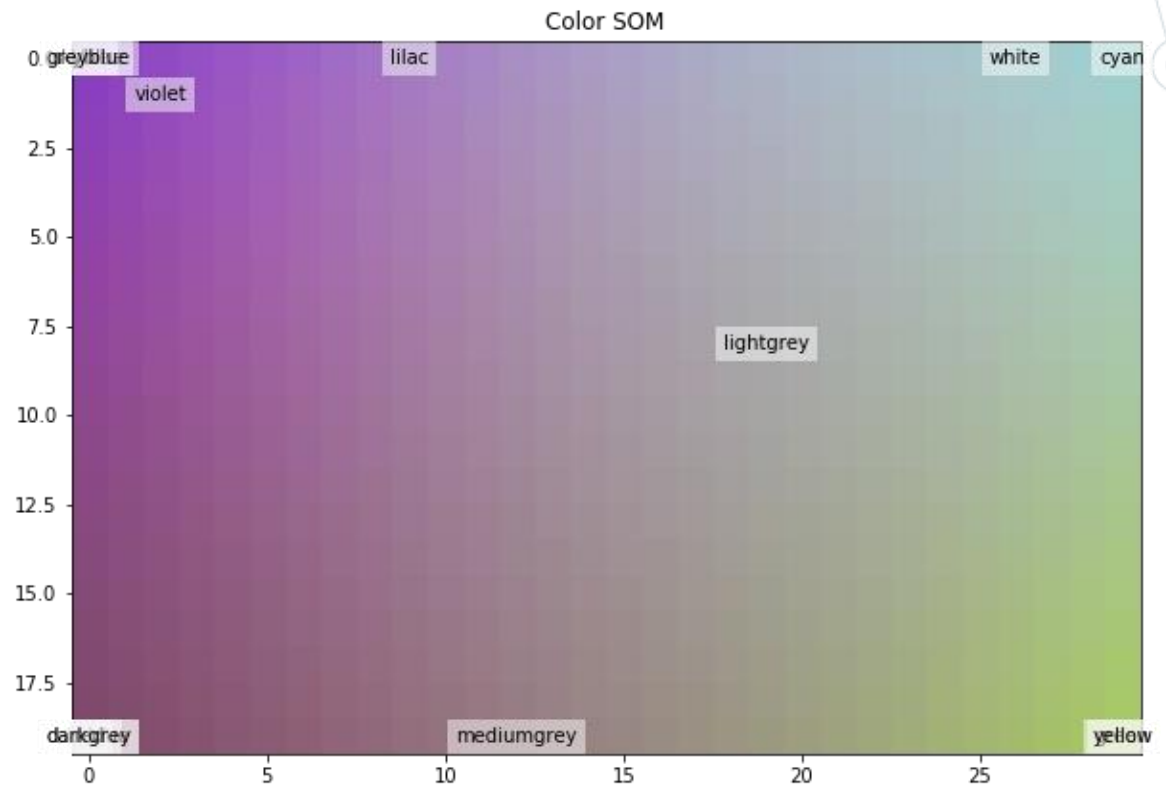
Testes

- ❖ $m = 20$
- ❖ $n = 30$
- ❖ $\text{dim} = 3$
- ❖ $\text{n_iterations} = 1000$
- ❖ $\alpha = 0.1$
- ❖ $\text{sigma} = \text{None}$
- ❖ $\text{decay_in_time} = \text{True}$



Testes

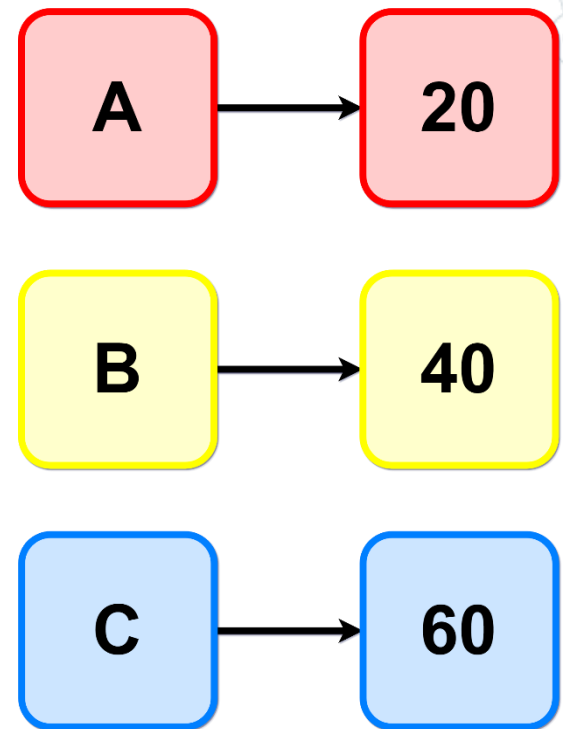
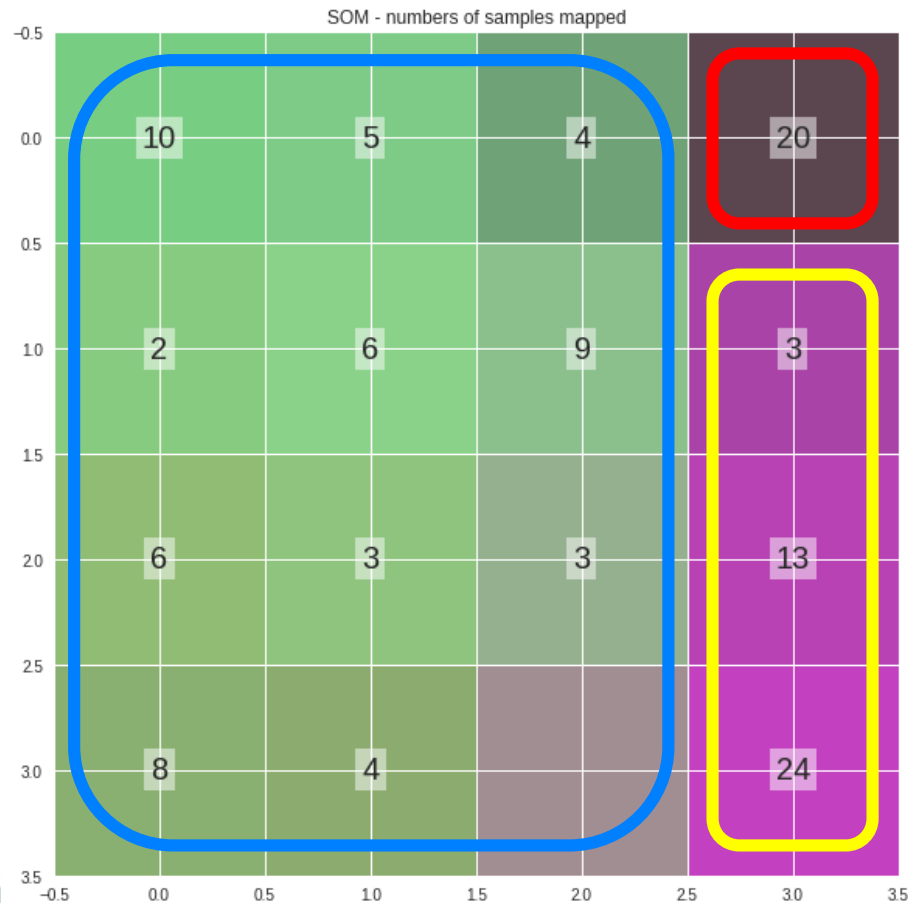
- ❖ $m = 20$
- ❖ $n = 30$
- ❖ $\text{dim} = 3$
- ❖ $n_iterations = 400$
- ❖ $\alpha = 0.5$
- ❖ $\sigma = \text{None}$
- ❖ **$\text{decay_in_time} = \text{False}$**



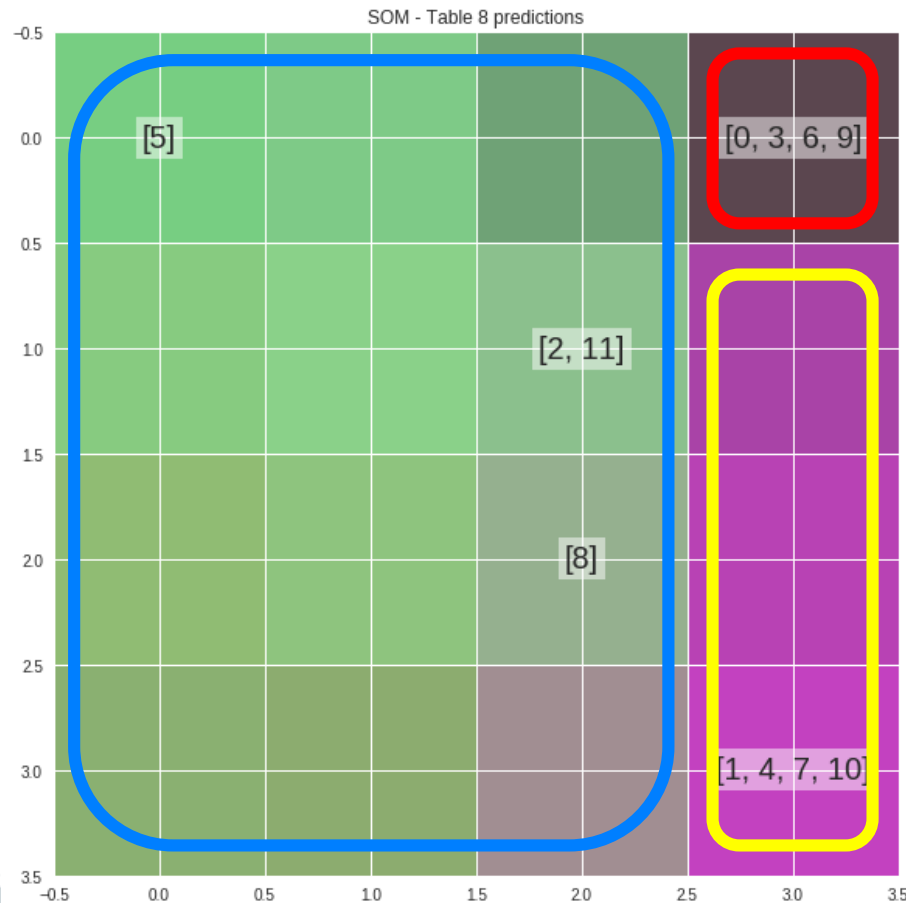
Problema proposto

- ❖ No processo de identificação de pneus:
 - ❖ Identificar eventuais similaridades e correlações entre as variáveis que fazem parte do processo de fabricação de pneus
- ❖ Mapa de Kohonen:
 - ❖ Com 16 neurônios
 - ❖ $\eta = 0,001$
 - ❖ Vizinhança unitária
 - ❖ 3 classes distintas
 - ❖ A – 20 amostras
 - ❖ B – 40 amostras
 - ❖ C – 60 amostras

Resultado: treino

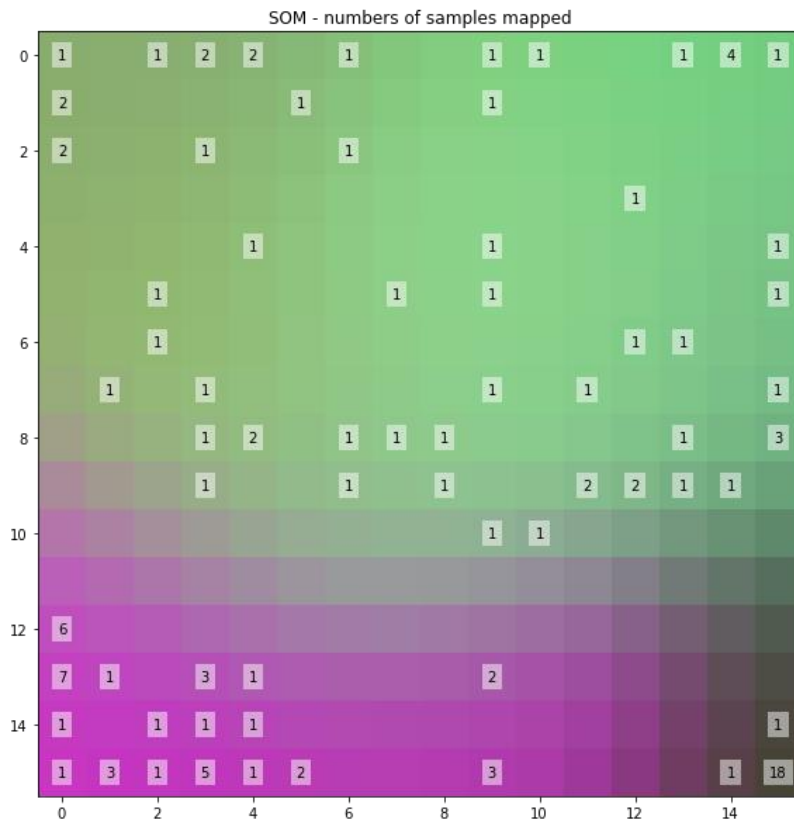
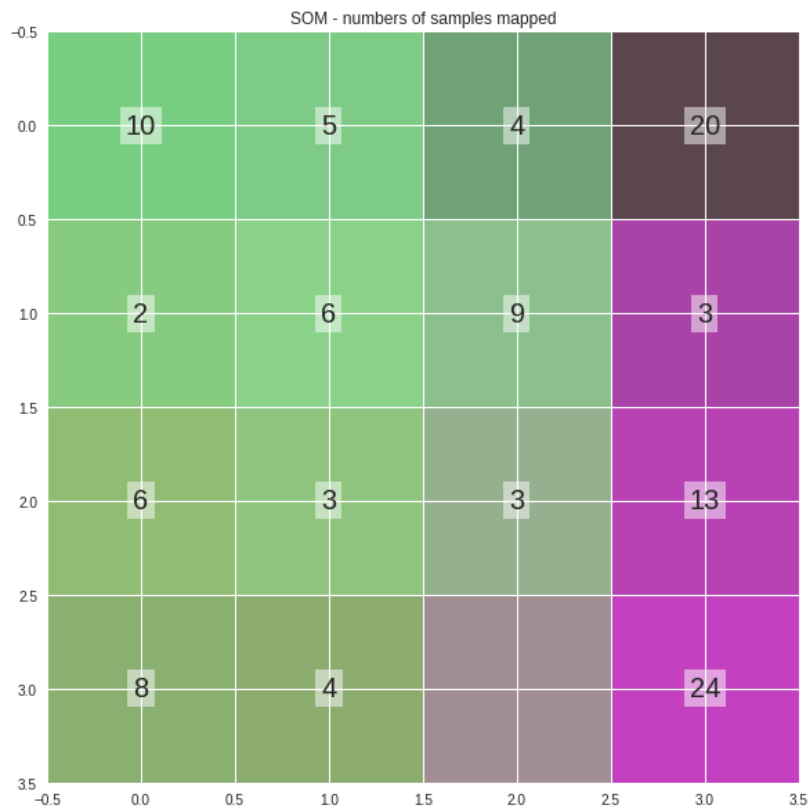


Resultado: teste

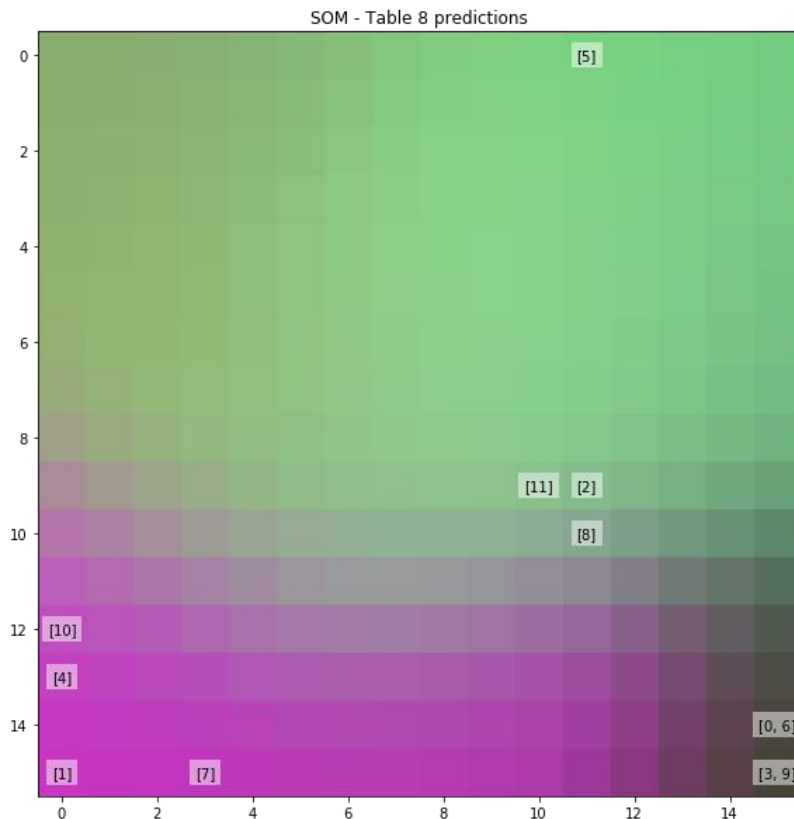
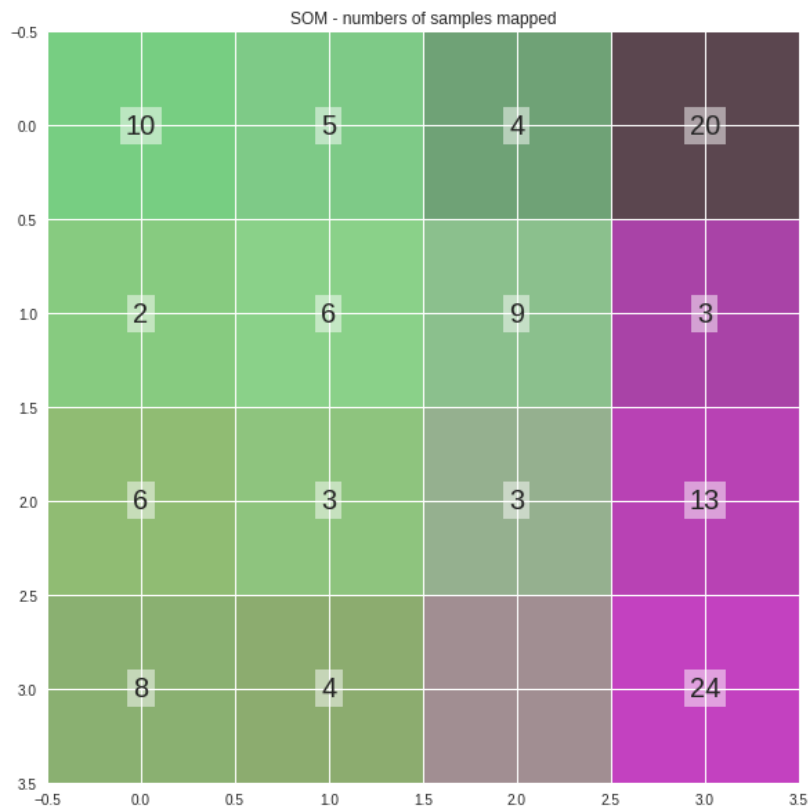


	x1	x2	x3
0	0.2471	0.1778	0.2905
1	0.8240	0.2223	0.7041
2	0.4960	0.7231	0.5866
3	0.2923	0.2041	0.2234
4	0.8118	0.2668	0.7484
5	0.4837	0.8200	0.4792
6	0.3248	0.2629	0.2375
7	0.7209	0.2116	0.7821
8	0.5259	0.6522	0.5957
9	0.2075	0.1669	0.1745
10	0.7830	0.3171	0.7888
11	0.5393	0.7510	0.5682

Resultado: comparações

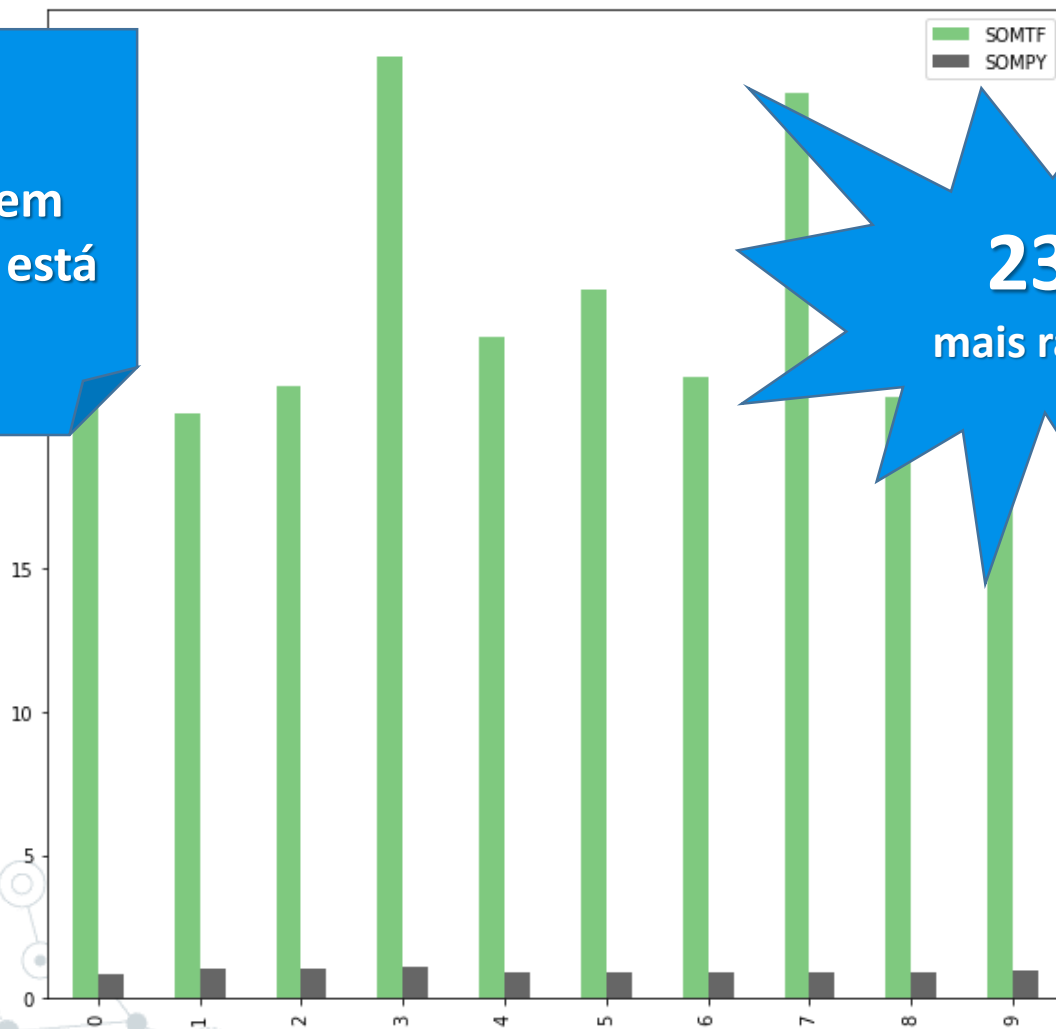


Resultado: comparações



Resultado: comparações

Conclusão:
Nosso código em
Tensorflow não está
otimizado!



23x
mais rápido

Referências

- ❖ Ivan Silva, Danilo Spatti and Rogério Fluzino "Redes Neurais Artificiais para engenharia e ciências aplicadas", Artliber, 2010
- ❖ <http://blog.yhat.com/posts/self-organizing-maps-1.html>
- ❖ <http://www.ai-junkie.com/ann/som/som1.html>
- ❖ <https://vahidmoosavi.com/2014/02/18/a-self-organizing-map-som-package-in-python-sompy/>
- ❖ <https://codesachin.wordpress.com/2015/11/28/self-organizing-maps-with-googles-tensorflow/>
- ❖ <http://www.decom.ufop.br/imobilis/self-organizing-maps/>
- ❖ <https://bop.unibe.ch/linguistik-online/article/view/788/1355>