

▼ Mapas Auto organizáveis de Kohonen

1. Title: Relative CPU Performance Data

2. Source Information -- Creators: Phillip Ein-Dor and Jacob Feldmesser -- Ein-Dor: Faculty of Management; Tel Aviv University; Ramat-Aviv;

Tel Aviv, 69978; Israel

-- Donor: David W. Aha (aha@ics.uci.edu) (714) 856-8779

-- Date: October, 1987

3. Past Usage:

1. Ein-Dor and Feldmesser (CACM 4/87, pp 308-317) -- Results: -- linear regression prediction of relative cpu performance -- Recorded 34% average deviation from actual values
2. Kibler,D. & Aha,D. (1988). Instance-Based Prediction of Real-Valued Attributes. In Proceedings of the CSCSI (Canadian AI) Conference. -- Results: -- instance-based prediction of relative cpu performance -- similar results; no transformations required
 - Predicted attribute: cpu relative performance (numeric)

4. Relevant Information: -- The estimated relative performance values were estimated by the authors using a linear regression method. See their article (pp 308-313) for more details on how the relative performance values were set.

5. Number of Instances: 209

6. Number of Attributes: 10 (6 predictive attributes, 2 non-predictive,

1 goal field, and the linear regression's guess)

Falha no salvamento automático. Este arquivo foi atualizado remotamente ou em outra guia.

[Mostrar diferenças](#)

1. vendor name: 30 !(adviser, amdahl,apollo, basf, bti, burroughs, c.r.d, cambex, cdc, dec, dg, formation, four-phase, gould, honeywell, hp, ibm, ipl, magnuson, microdata, nas, ncr, nixdorf, perkin-elmer, prime, siemens, sperry, sratus, wang)(campo não utilizado, string)
2. !Model Name: many unique symbols (campo não utilizado, tipo misto)
3. MYCT: machine cycle time in nanoseconds (integer)
4. MMIN: minimum main memory in kilobytes (integer)
5. MMAX: maximum main memory in kilobytes (integer)
6. CACH: cache memory in kilobytes (integer)

7. CHMIN: minimum channels in units (integer)
8. CHMAX: maximum channels in units (integer)
9. PRP: published relative performance (integer)
10. ERP: estimated relative performance from the original article (integer)

8. Missing Attribute Values: None

Disponível em: <https://www.kaggle.com/balajisriraj/relative-cpu-performance-data?select=machine.names>

```
print("9. Class Distribution: the class value (PRP) is continuously valued.")
print("|PRP Value Range:\t\t|Number of Instances in Range: |")
print("|0-20\t\t\t\t|31\t\t\t\t|")
print("|21-100\t\t\t\t|121\t\t\t\t|")
print("|101-200\t\t\t\t|27\t\t\t\t|")
print("|201-300\t\t\t\t|13\t\t\t\t|")
print("|301-400\t\t\t\t|07\t\t\t\t|")
print("|401-500\t\t\t\t|04\t\t\t\t|")
print("|501-600\t\t\t\t|02\t\t\t\t|")
print("|above 600\t\t\t\t|04\t\t\t\t|")
```

```
9. Class Distribution: the class value (PRP) is continuously valued.
|PRP Value Range:      |Number of Instances in Range: |
|0-20                  |31                             |
|21-100                |121                            |
|101-200               |27                             |
|201-300               |13                             |
|301-400               |07                             |
|401-500               |04                             |
|501-600               |02                             |
|above 600             |04                             |
```

```
print("Summary Statistics:")
print("\t|Min \t\t|Max \t\t|Mean\t\t|SD\t\t|PRP Correlation\t|")
print("MCYT:\t|17\t\t|1500\t\t|203.8\t\t|260.3\t\t|-0.3071\t\t|")
print("MMIN:\t|64\t\t|32000\t\t|2868.0\t\t|3878.7\t\t|0.7949\t\t|")
```

Falha no salvamento automático. Este arquivo foi atualizado remotamente ou em outra guia.

[Mostrar diferenças](#)

```
print("CHMAX:\t|0\t\t|176\t\t|18.2\t\t|26.0\t\t|0.6052\t\t|")
print("PRP:\t|6\t\t|1150\t\t|105.6\t\t|160.8\t\t|1.0000\t\t|")
print("ERP:\t|15\t\t|1238\t\t|99.3\t\t|154.8\t\t|0.9665\t\t|")
```

Summary Statistics:

	Min	Max	Mean	SD	PRP
MCYT:	17	1500	203.8	260.3	-0.3
MMIN:	64	32000	2868.0	3878.7	0.79
MMAX:	64	64000	11796.1	11726.6	0.86
CACH:	0	256	25.2	40.6	0.66
CHMIN:	0	52	4.7	6.8	0.60
CHMAX:	0	176	18.2	26.0	0.60

PRP:	6	1150	105.6	160.8	1.00
ERP:	15	1238	99.3	154.8	0.96



▼ Vamos começar!

Após baixar no UCI Machine Learning Repository o arquivo *machine.csv* contendo os dados sobre os resultados de uma análise dos Dados relativos de desempenho da CPU de diversos fabricantes, vamos preparar nosso ambiente com as bibliotecas necessárias e depois importaremos os dados!

Importando as bibliotecas

```
!pip install minisom
# instalar a biblioteca <minisom> *no Anaconda Prompt digite <> pip install minisom
from minisom import MiniSom
import pandas as pd
import numpy as np
```

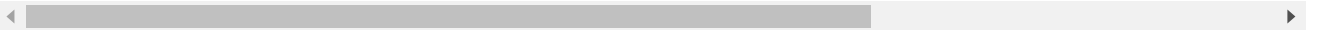
```
Requirement already satisfied: minisom in /usr/local/lib/python3.7/dist-packa
```



▼ Carregando os dados

```
from google.colab import drive
drive.mount('/gdrive')
```

```
Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.
```



```
base = pd.read_csv('/gdrive/My Drive/DATA/machine.csv')
```

Falha no salvamento automático. Este arquivo foi atualizado remotamente ou em outra guia.

[Mostrar diferenças](#)

```
base.describe()
```

▼ Camada de entrada e rótulos para validação

```
X = base.iloc[:,1:7].values  
y = base.iloc[:,0].values
```

▼ Normalização dos dados

```
from sklearn.preprocessing import MinMaxScaler  
normalizador = MinMaxScaler(feature_range = (0,1))  
X = normalizador.fit_transform(X)
```

▼ Construção do SOM

O algoritmo de Kohonen foi desenvolvido por Teuvo Kohonen em 1982, sendo considerado relativamente simples e com a capacidade de organizar dimensionalmente dados complexos em grupos (clusters), de acordo com suas relações. Este método solicita apenas os parâmetros


Falha no salvamento automático. Este arquivo foi atualizado remotamente ou em outra guia.

[Mostrar diferenças](#)
indeterminados. 

```
som = MiniSom(x = 10, y = 10, input_len = 6, sigma = 1.0, learning_rate = 0.9, random_weights_init(X)  
som.train_random(data = X, num_iteration = 100000)
```

▼ Topologia da Rede

Os SOMs têm duas camadas, a primeira é a camada de entrada e a segunda é a camada de saída ou o mapa de características. Ao contrário de outros tipos de RNA, SOM não tem função

de ativação em neurônios, passamos pesos diretamente para a camada de saída sem fazer nada. Cada neurônio em um SOM é atribuído a um vetor de peso com a mesma dimensionalidade d do espaço de entrada. 

som._activation_map

```
array([[1.48968206, 1.48225112, 1.15111759, 1.11452342, 0.8800262 ,
        1.08182237, 0.55618359, 0.20812675, 0.32926933, 0.37520645],
       [1.72686516, 1.30180502, 1.18012029, 0.90195367, 0.59874851,
        0.58328802, 0.31182765, 0.14395938, 0.00781558, 0.10008395],
       [1.16158423, 0.88667286, 1.21734866, 0.96839211, 0.57713607,
        0.61277302, 0.32433471, 0.24965323, 0.19221017, 0.20618071],
       [0.73995827, 0.76475656, 0.81286129, 0.77032672, 0.57657216,
        0.39312039, 0.24475952, 0.26252313, 0.25468574, 0.26190252],
       [0.59240704, 0.59591055, 0.59408408, 0.50095966, 0.40239507,
        0.29399272, 0.29226125, 0.24733244, 0.29314603, 0.30100478],
       [0.50224131, 0.40347793, 0.37970669, 0.3532823 , 0.32864615,
        0.28915972, 0.32653945, 0.33505582, 0.34900676, 0.33967796],
       [0.513352 , 0.42446887, 0.37540415, 0.33651193, 0.35266733,
        0.33091742, 0.33238958, 0.34441958, 0.38696558, 0.38302693],
       [0.46927256, 0.44441414, 0.43522344, 0.34832582, 0.38534478,
        0.35781172, 0.35012997, 0.36212068, 0.38990872, 0.36534155],
       [0.50142862, 0.40696925, 0.44508263, 0.38822721, 0.39789045,
        0.37951051, 0.33888548, 0.32502967, 0.36659958, 0.37155862],
       [0.44319607, 0.35029818, 0.43016504, 0.42385779, 0.35489146,
        0.36156426, 0.34949445, 0.3705257 , 0.36661888, 0.38020656]])
```

▼ Pesos do SOM

som._weights

```
9.18339234e-02, 1.44559431e-01],
[2.45260250e-01, 2.49285747e-01, 2.01114236e-01, 4.12466581e-02,
 5.47496728e-02, 1.05922296e-01],
[1.23127304e-01, 2.20906762e-01, 1.24145539e-01, 1.04447770e-01,
 6.73355073e-02, 1.01295088e-01],
[6.20569457e-02, 2.48511258e-01, 8.41314099e-02, 2.07029905e-02,
```

Falha no salvamento automático. Este arquivo foi atualizado remotamente ou em outra guia.

[Mostrar diferenças](#)

```
[1.20516375e-01, 1.24905977e-01, 3.79764186e-04, 1.96780552e-02,
 3.11150006e-02, 2.40990710e-02],
[5.95445658e-02, 6.53292138e-02, 3.84452165e-03, 2.10287630e-02,
 1.90781893e-02, 2.52358848e-02],
[3.60596075e-03, 7.97568117e-03, 9.81011969e-04, 1.91061194e-02,
 1.25358584e-02, 3.06151508e-03],
[1.40812522e-02, 5.67982477e-02, 1.49889998e-02, 1.91852501e-02,
 2.32348537e-02, 5.57684122e-02]],
[[6.06365075e-02, 4.99478736e-01, 1.24620469e-01, 2.04257783e-02,
 1.18118893e-01, 1.12184374e-01],
[1.25221262e-01, 3.46998030e-01, 1.85345067e-01, 5.62279201e-02,
 1.18640192e-01, 1.23031859e-01],
[1.25285360e-01, 2.49211480e-01, 2.47617503e-01, 2.28009681e-02,
 4.54292673e-02, 1.11792317e-01],
```

```
[1.23003578e-01, 2.48744978e-01, 9.48382294e-02, 1.97265263e-02,
 3.43113076e-02, 6.15847117e-02],
[3.63523425e-02, 2.41199242e-01, 6.63134167e-02, 1.96542753e-02,
 1.97128315e-02, 1.05157020e-01],
[3.94927054e-02, 1.85904260e-01, 5.55871983e-02, 2.26995390e-02,
 1.59095521e-02, 4.51406234e-02],
[5.57894881e-02, 1.25907620e-01, 3.23317084e-02, 5.51693017e-02,
 3.63548500e-02, 3.62854011e-02],
[2.87600719e-02, 1.81154856e-01, 3.36737829e-02, 5.52681804e-02,
 6.49653241e-02, 5.49172366e-02],
[2.00213958e-02, 7.94662580e-02, 5.79755614e-02, 1.95246051e-02,
 2.48922678e-02, 2.66429816e-02],
[1.74913155e-02, 6.29584092e-02, 8.92941991e-04, 1.91513720e-02,
 1.81377834e-02, 1.30496253e-02]],

[[6.06212432e-02, 4.99322806e-01, 1.25088258e-01, 1.92307718e-02,
 3.05756173e-01, 1.26770246e-01],
[6.25780066e-02, 2.60487855e-01, 2.44386761e-01, 2.27504612e-02,
 2.12331250e-01, 1.19398626e-01],
[9.86749215e-02, 2.48802662e-01, 2.49998030e-01, 9.55368819e-02,
 4.96680107e-02, 7.54130347e-02],
[6.25640772e-02, 1.29044796e-01, 2.46831519e-01, 2.06461228e-02,
 3.58030694e-02, 4.98144126e-02],
[5.97289211e-02, 1.24289852e-01, 1.25383123e-01, 1.96718998e-02,
 4.89822821e-02, 6.40699005e-02],
[2.83288672e-02, 1.24529416e-01, 1.03799460e-01, 4.57232644e-02,
 3.37703913e-02, 2.92387240e-02],
[5.16257739e-02, 1.24128583e-01, 4.61210143e-02, 1.94413321e-02,
 3.49871455e-02, 5.00544698e-02],
[7.32580957e-03, 1.24192719e-01, 7.12104555e-04, 1.93791774e-02,
 2.15983548e-02, 1.03254104e-02],
[2.46998019e-02, 6.23294092e-02, 1.88560133e-02, 1.91439274e-02,
 2.19224847e-02, 1.71414364e-02],
[2.31796291e-02, 3.14681216e-02, 6.33664471e-05, 1.99957852e-02,
 1.10010001e-02, 1.00000000e-02]]]
```

▼ Visualização dos Dados

▼ Visualização de quantas vezes determinado neurônio foi ativado

Falha no salvamento automático. Este arquivo foi atualizado remotamente ou em outra guia.

[Mostrar diferenças](#)

com a mesma dimensionalidade do espaço de entrada. No exemplo abaixo, em cada neurônio da camada de saída teremos um vetor com dimensão n . Calculamos a distância entre cada neurônio (neurônio da camada de saída) e os dados de entrada, e o neurônio com a menor distância será o vencedor da competição.

A métrica euclidiana d_j é comumente usada para medir distâncias:

$$d_j = \sum_{i=1}^n (x_i - w_{ij})^2$$



```
q = som.activation_response(X)
q

array([[ 0.,  2.,  0.,  2.,  0.,  1.,  1.,  0.,  1.,  1.],
       [ 1.,  0.,  0.,  0.,  1.,  0.,  1.,  2.,  2.,  1.],
       [ 1.,  1.,  1.,  0.,  0.,  2.,  0.,  2.,  1.,  1.],
       [ 1.,  3.,  1.,  1.,  1.,  0.,  1.,  2.,  3.,  4.],
       [ 4.,  1.,  1.,  1.,  1.,  2.,  2.,  2.,  4.,  2.],
       [ 3.,  1.,  2.,  0.,  2.,  2.,  1.,  7.,  4.,  3.],
       [ 1.,  1.,  3.,  2.,  5.,  1.,  5.,  4.,  5.,  4.],
       [ 1.,  1.,  2.,  3.,  4.,  2.,  2.,  2.,  2.,  2.],
       [ 1.,  0.,  1.,  2.,  1.,  3.,  3.,  1.,  2.,  9.],
       [ 2.,  1.,  3.,  2.,  3.,  4.,  7.,  9.,  5., 11.]])
```

▼ MID - distância média entre neurônios

```
from matplotlib.pyplot import pcolor, colorbar, plot, plt
pcolor(som.distance_map().T)
# MID - mean inter neuron distance
colorbar()
```

▼ **Erro de salvamento automático**

Falha no salvamento automático. Este arquivo foi atualizado remotamente ou em outra guia.
[Mostrar diferenças](#)

```
for i, x in enumerate(X):
    print(i)
    print(x)
    w = som.winner(x)
    print(w)

(7, 3)
189
[0.24849699 0.24924925 0.25          0.23076923 0.13636364 0.15909091]
(6, 1)
190
[0.24849699 0.37437437 0.125          0.15384615 0.09090909 0.14597902]
(7, 1)
```

```

191
[0.24849699 0.4994995 0.25      0.23076923 0.09090909 0.21153846]
(4, 1)
192
[0.24849699 0.4994995 0.5      0.46153846 0.18181818 0.34877622]
(3, 2)
193
[0.06062124 0.12412412 0.125    0.09615385 0.15909091 0.05594406]
(3, 6)
194
[0.06062124 0.4994995 0.09375   0.11538462 0.14772727 0.09440559]
(7, 0)
195
[0.06062124 0.4994995 0.1875    0.5         0.29545455 0.17657343]
(4, 2)
196
[0.06062124 0.4994995 0.4375    1.          0.59090909 0.26311189]
(0, 3)
197
[0.12324649 0.4994995 0.4375    1.          0.59090909 0.34178322]
(0, 3)
198
[0.24849699 1.          0.375     0.23076923 1.          0.79458042]
(0, 1)
199
[0.24849699 1.          0.5      0.23076923 1.          1.          ]
(0, 1)
200
[0.0061999 0.06156156 0.         0.01923077 0.01704545 0.00524476]
(8, 9)
201
[0.01402806 0.06156156 0.         0.01923077 0.01704545 0.00699301]
(8, 9)
202
[0.0061999 0.06156156 0.         0.01923077 0.01704545 0.01048951]
(8, 9)
203
[0.01402806 0.06156156 0.         0.01923077 0.01704545 0.01311189]
(8, 9)
204
[0.02930862 0.12412412 0.         0.01923077 0.04545455 0.03146853]
(6, 6)
205

```

Falha no salvamento automático. Este arquivo foi atualizado remotamente ou em outra guia.

[Mostrar diferenças](#)

```

206
[0.06062124 0.12412412 0.         0.03846154 0.07954545 0.04020979]
(6, 6)
207
[0.01402806 0.12412412 0.125     0.          0.          0.05332168]
(9, 5)

```

```

w = som.winner(X[148]) #Características do vinho da 3 linha do dataset ativa o neu
print(w)

```

```

(4, 4)

```

```

import matplotlib.patches as mpatches

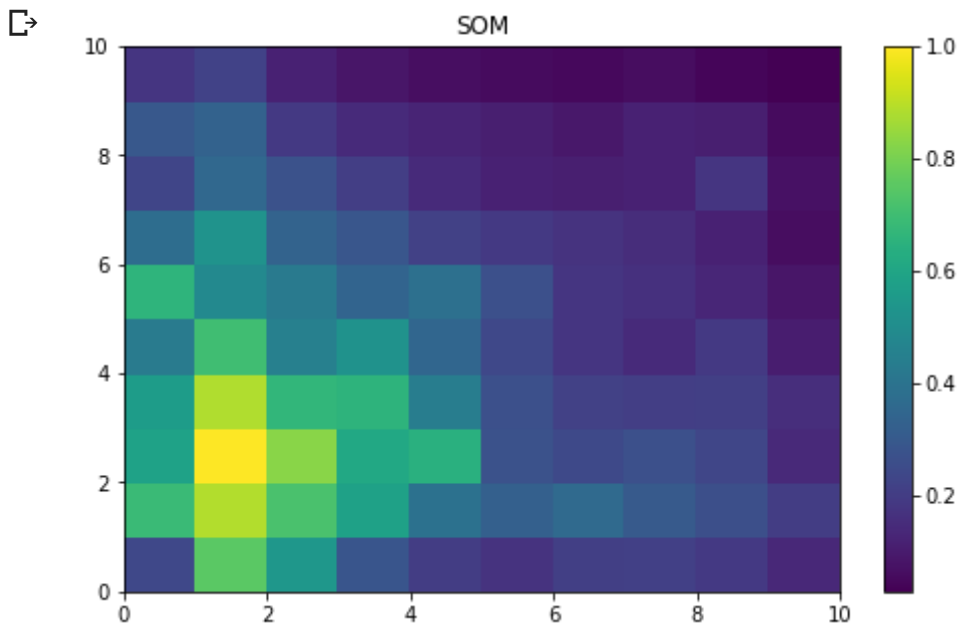
```



```
plt.figure(figsize=(8,5))
plt.title('SOM')
plt.pcolor(som.distance_map().T) # distance map as background
plt.colorbar()

for cnt,xx in enumerate(X):
    w = som.winner(xx) # getting the winner

plt.axis([0,som.get_weights().shape[0],0,som.get_weights().shape[1]])
plt.show() # show the figure
```



Falha no salvamento automático. Este arquivo foi atualizado remotamente ou em outra guia.

[Mostrar diferenças](#)