Redes Neurais no MATLAB

Redes SOM

- Self Organizing Maps –

-SOM Toolbox –

http://www.cis.hut.fi/projects/somtoolbox/

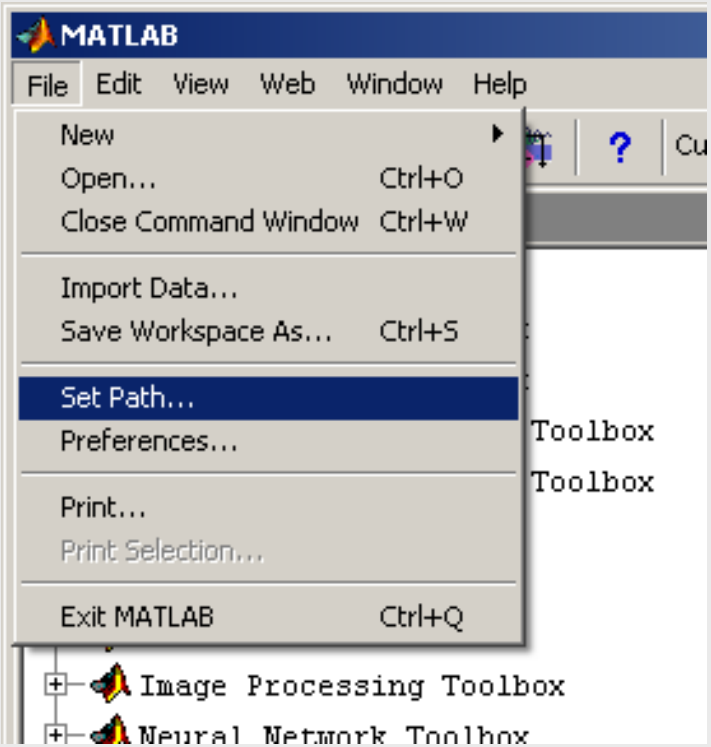
Instalação

- O SOM Toolkit está disponível no link:

http://www.cis.hut.fi/projects/somtoolbox/package/somtoolbox2_Mar_17_2005.zip

- Descompacte o arquivo e no Matlab inclua a pasta na lista de diretórios do Matlab.

Instalação



- clique em Add Folder

- escolha a pasta na qual foi descompactado o Toolbox

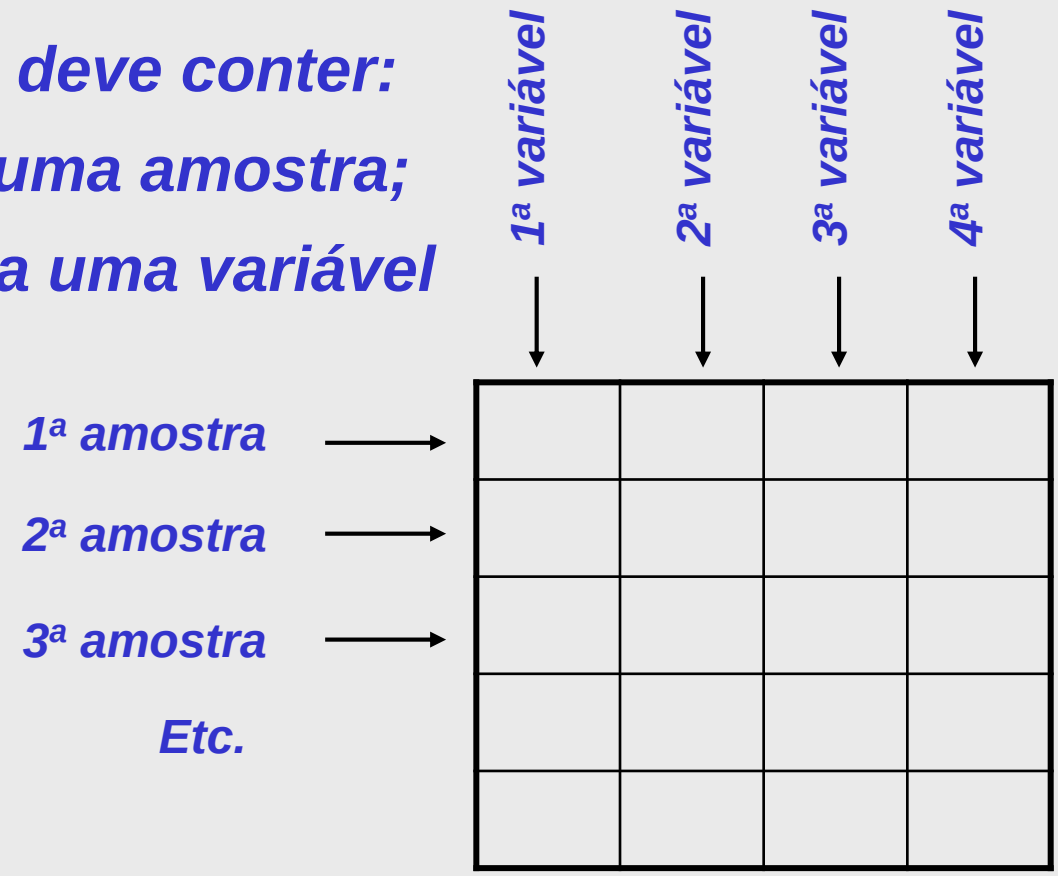
- clique em Save

Formato da Base de Dados

A base de dados deve conter:

- em cada linha uma amostra;

- em cada coluna uma variável



Formato da Base de Dados

- *A base de dados é um arquivo em formato texto com colunas separadas por espaço.*
- *A primeira linha indica o número de variáveis*
- *Os nomes das variáveis devem ser precedidos por #n*

```
4
#n SepalL SepalW PetalL PetalW
5.1 3.5 1.4 0.2 Setosa
4.9 3.0 1.4 0.2 Setosa
4.7 3.2 1.3 0.2 Setosa
4.6 3.1 1.5 0.2 Setosa
...
```

Funções de Distância

Distância de Minkowski

$$d(\mathbf{x},\mathbf{y}) = \left[\sum_{k=1}^{n} |x_k - y_k|^p\right]^{1/p}, p \geq 1$$

Exemplo para 2 dados com 2 atributos usando distância de Minkowski para p=2:

- $\mathbf{x} = (x_1, x_2)$
- $\mathbf{y} = (y_1, y_2)$

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Funções de Distância - Casos particulares

Distância Manhattan – p=1

$$d(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{n} |x_k - y_k|$$

Distância Euclidiana – p=2

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

Criando uma Rede SOM

- **A criação de uma rede SOM deve seguir os seguintes passos:**

1. **Carregar a base de dados**
2. **Normalizá-la**
3. **Treinar o mapa**
4. **Visualizar o mapa**
5. **Analisar os resultados**

Criando uma Rede SOM

1. Carregar a base de dados

```
sD = som_read_data('iris.data');
```

Criando uma Rede SOM

sD = som_read_data('iris.data');

- *data read ok*

<sD 1x1 struct>:

type 'som_data'

> data <150x4 double> 0.1000 7.9000

labels <150x1 cell>

name 'iris.data'

comp_names <4x1 cell>

comp_norm <4x1 cell>

label_names [ ]

Criando uma Rede SOM

2. Normalizá-la

```
sD = som_normalize(sD,'range');
```

Opções: var, range, log, logistic, histD, histC

Criando uma Rede SOM

3. Treinar o mapa

Duas opções:

- **parâmetros automáticos**

```
sM = som_make(sD);
```

Criando uma Rede SOM

```
sM = som_make(sD);
```

Determining map size...
map size [13, 5]
Initialization...
Training using batch algorithm...
Rough training phase...
Training: 0/ 0 s
Training: 0/ 0 s
…...
Finetuning phase...
Training: 0/ 0 s
Training: 0/ 0 s
…....
Training: 0/ 0 s
Final quantization error: 0.093
Final topographic error: 0.027

Criando uma Rede SOM

3. Treinar o mapa

- Especificando os parâmetros

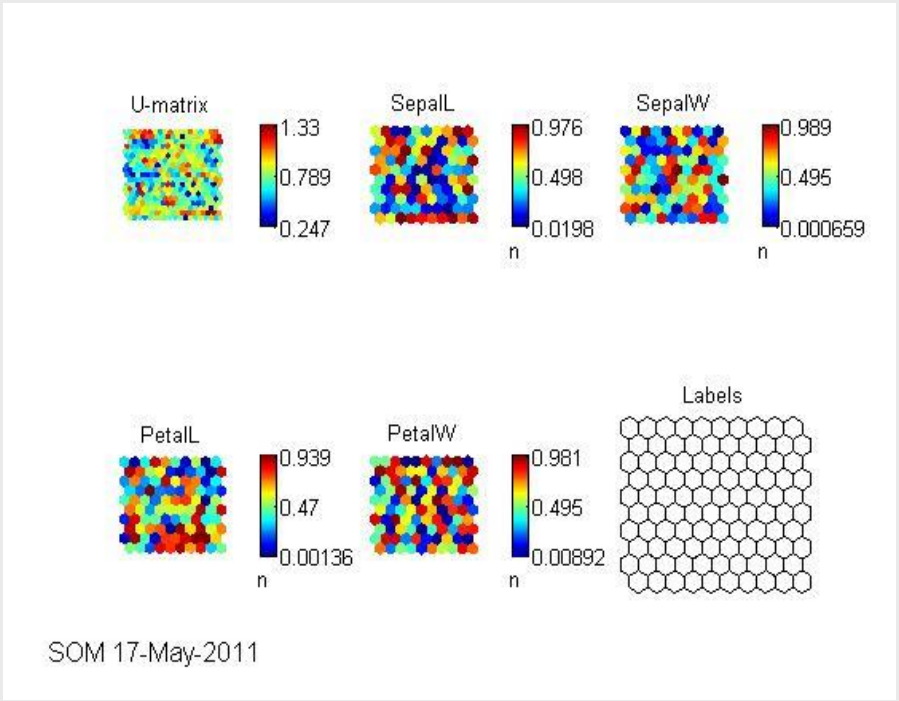
```
sM = som_randinit(sD,'msize', [10 10]);
```

Inicializa aleatoriamente o mapa com 10x10 neurônios

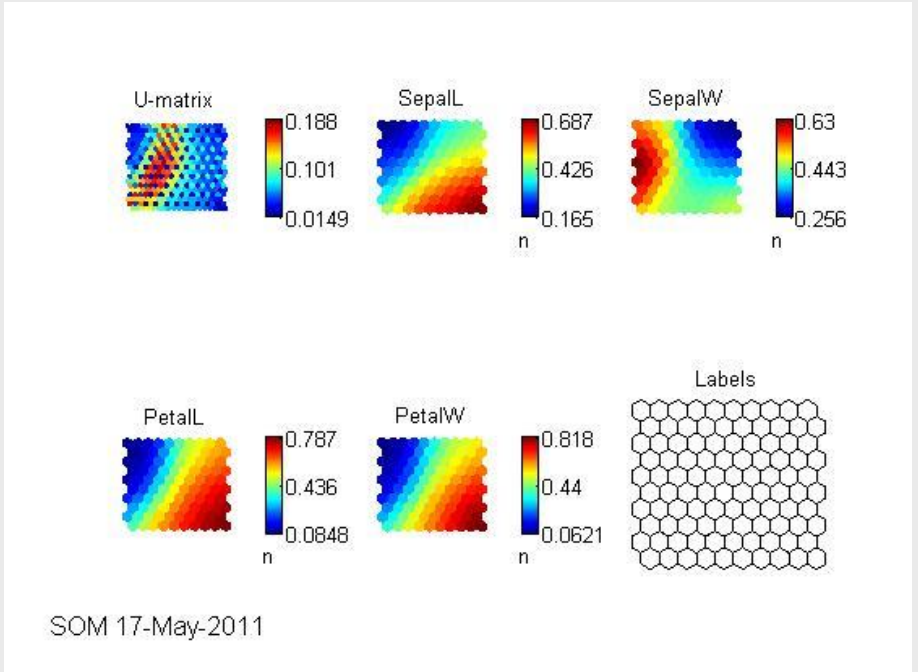
```
sM = som_seqtrain(sM,sD,'radius',[4 1]);
```

Treina sequencialmente o mapa com vizinhança inicial de 4 até uma vizinhança final de 1

Criando uma Rede SOM



som_randinit(sD,'msize', [10 10]);



som_seqtrain(sM,sD,'radius',[4 1]);

Criando uma Rede SOM

D (struct) Dados de treinamento; estrutura dos dados

(matrix) matriz de treinamento, tamanho é dlen x dim

[argID, (string) Veja a seguir. Os valores ambíguos (*) podem ser dados sem o argumento precedente (argID).

.....

'radius' (scalar) raio para treinamento inicial

'mask' (vector) máscara de pesquisa BMU, tamanho dim x 1

'msize' (vector) tamanho do mapa

'alpha_ini' (scalar) taxa de aprendizagem inicial

*'trainlen_type' *(string) é o número de 'samples' ou 'epochs'*

*'train' *(struct) estrutura de treinamento, parâmetros para treinamento*

*'topol' *(struct) estrutura da topologia*

'som_topol','sTopo l' = 'topol'

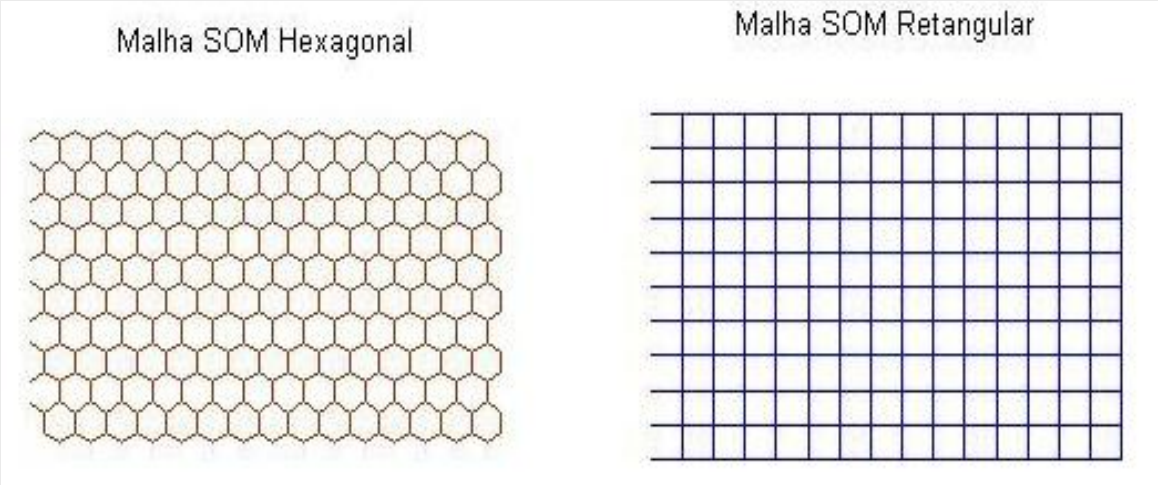
*'lattice' *(string) tipo de rede, 'hexa' or 'rect'*

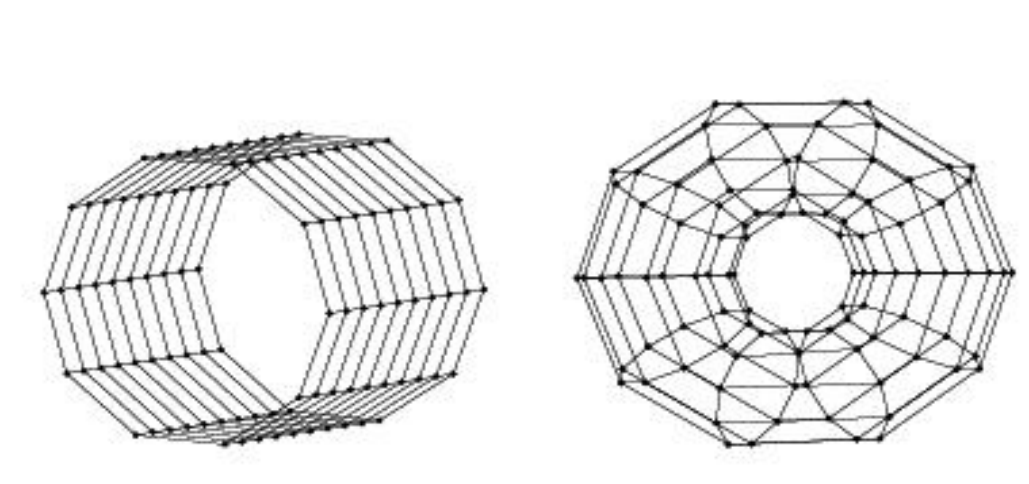
*'shape' *(string) tipo de mapa, plano=> 'sheet', cilíndrico=> 'cyl' ou toroidal => 'toroid'*

Criando uma Rede SOM - Topologia

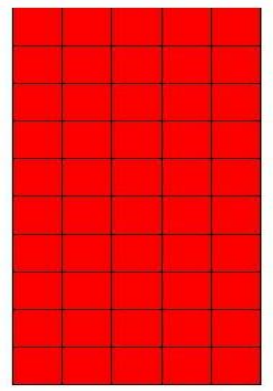
som_cplane('hexa',[10 15],'none')
title('Malha SOM Hexagonal')
.....

som_cplane('rect',[10 15],'none')
title('Malha SOM Retangular')
.....





som_cplane('hexa',[10 5],'r')



Informações do Mapa

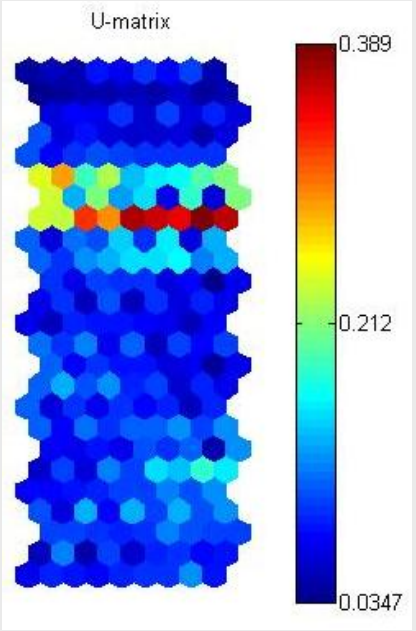
```
>>som_info(sM)

Struct type                    : som_map
Map name                       : SOM 17-May-2011
Input dimension                : 4
Map grid size                  : 13 x 5
Lattice type (rect/hexa)       : hexa
Shape (sheet/cyl/toroid)       : sheet
Neighborhood type              : gaussian
Mask                           : 1 1 1 1
Training status                : initialized, trained 2 times
```

Criando uma Rede SOM

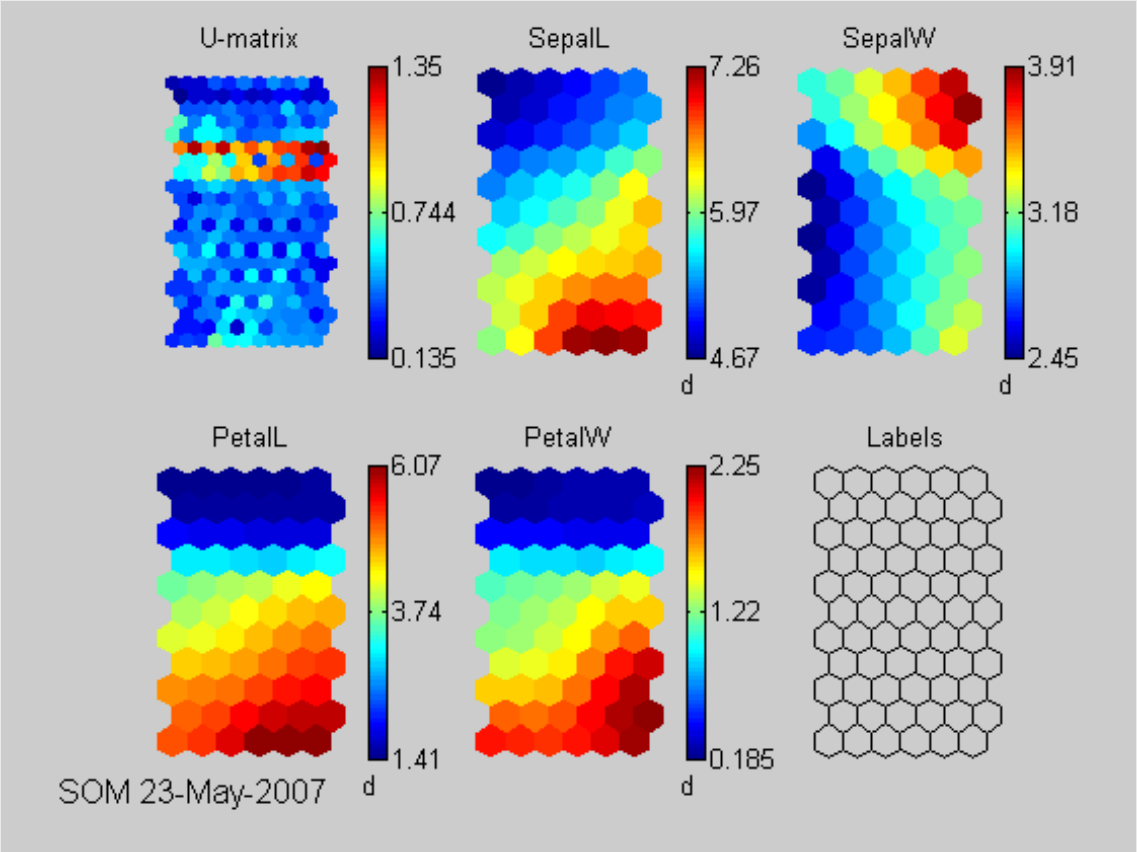
4. Visualizar o mapa

```
som_show(sM,'umat','all','comp',1:4,
   'empty','Labels','norm','d');
```



Criando uma Rede SOM

5. Analisar os resultados



Criando uma Rede SOM

5. Analisar os resultados

- **Na matriz U, nota-se que as 3 primeiras linhas do SOM formam claramente um cluster.**
- **Através dos planos das componentes, observa-se que este cluster possui:**
  - **Pétalas de largura e comprimento pequenos**
  - **Folhas de pequeno comprimento**
  - **Folhas de largura relativamente alta**

Criando uma Rede SOM

5. Analisar os resultados

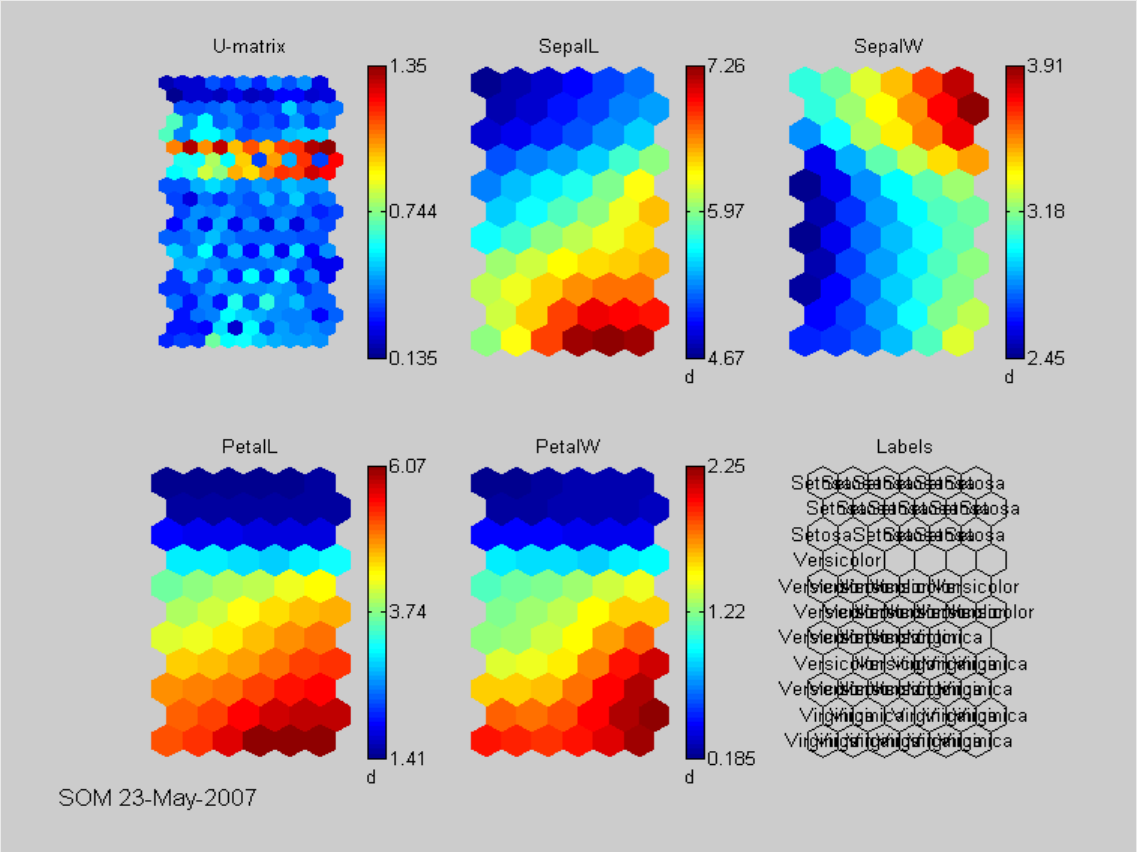
Pode-se colocar rótulos:

```
sM = som_autolabel(sM,sD,'vote');

som_show_add('label',sM,'subplot',6);
```

Criando uma Rede SOM

5. Analisar os resultados



Criando uma Rede SOM

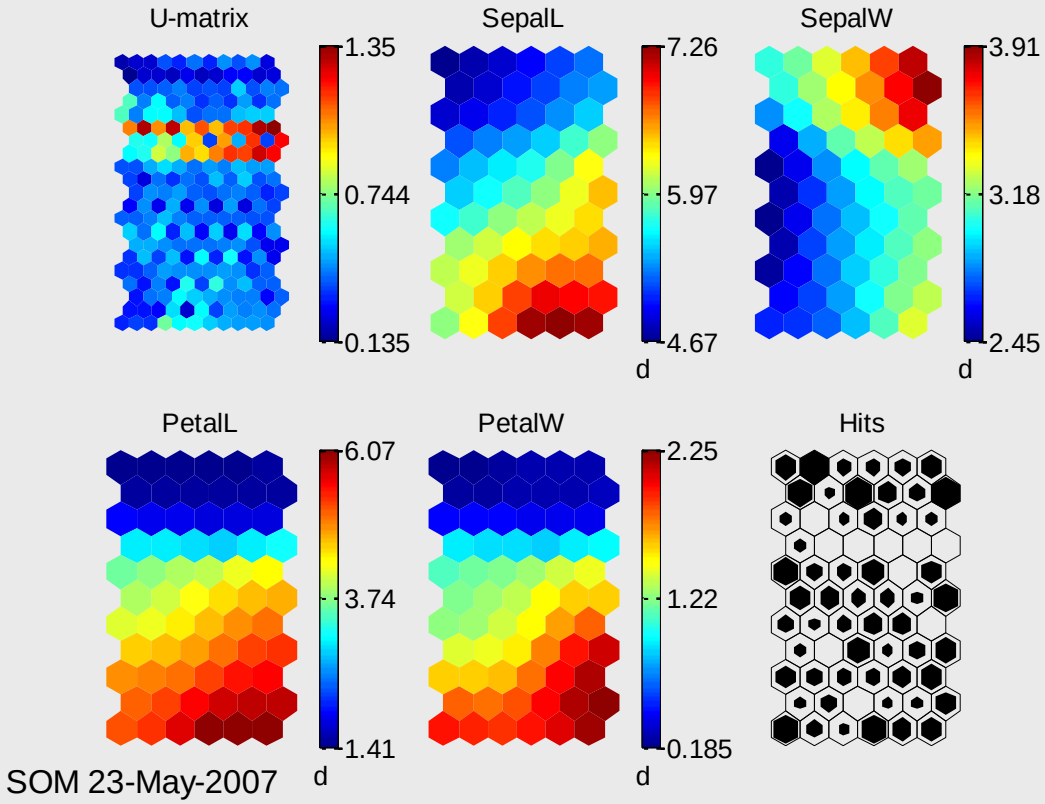
5. Analisar os resultados

Pode-se colocar o número de *hits*:

som_show_add('hit',som_hits(sM,sD),'subplot',6)

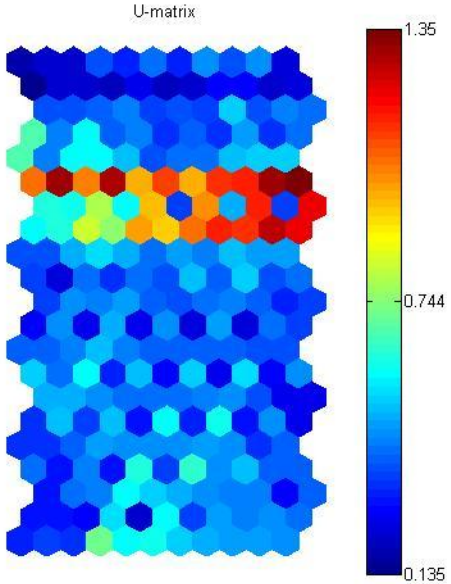
Criando uma Rede SOM

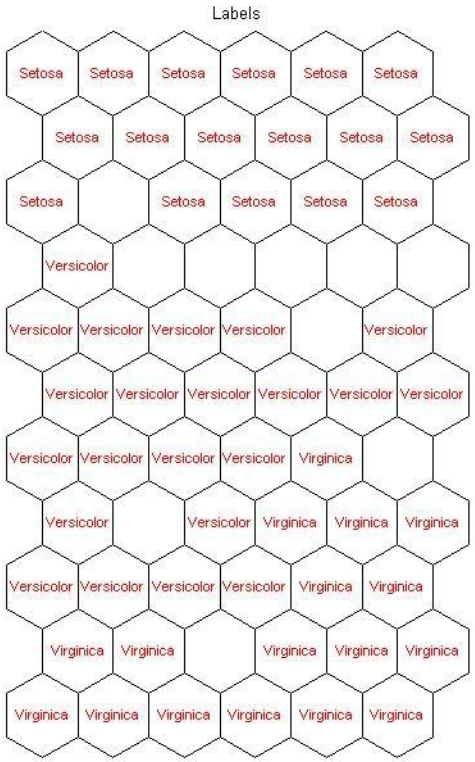
5. Analisar os resultados



Criando uma Rede SOM

5. Analisar os resultados





SOM 30-Mar-2011

K-means com Kohonen

[c,p,err,ind] = kmeans_clusters(sM,[n_max], [c_max])

onde:

Input:

sM – mapa;

[n_max] – número máximo de clusters;

[c_max)- número máximo de iterações;

Output:

***c** – centróides dos clusters;*

***p** – índices dos clusters;*

***err** – soma dos quadrados dos erros para cada valor de k;*

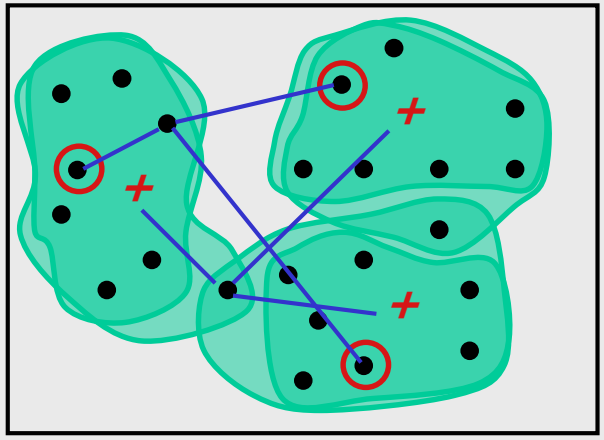
***ind** – valor do índice de Davies-Bouldin para cada cluster.*

[c, p, err, ind] = kmeans_clusters(sM);

[dummy,i] = min(ind);

Algoritmo *K-means* - crisp

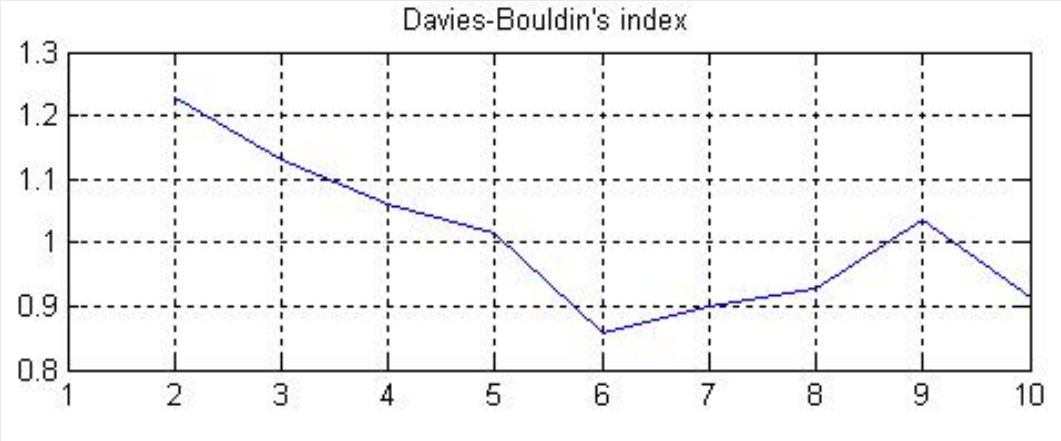
Exemplo k = 3

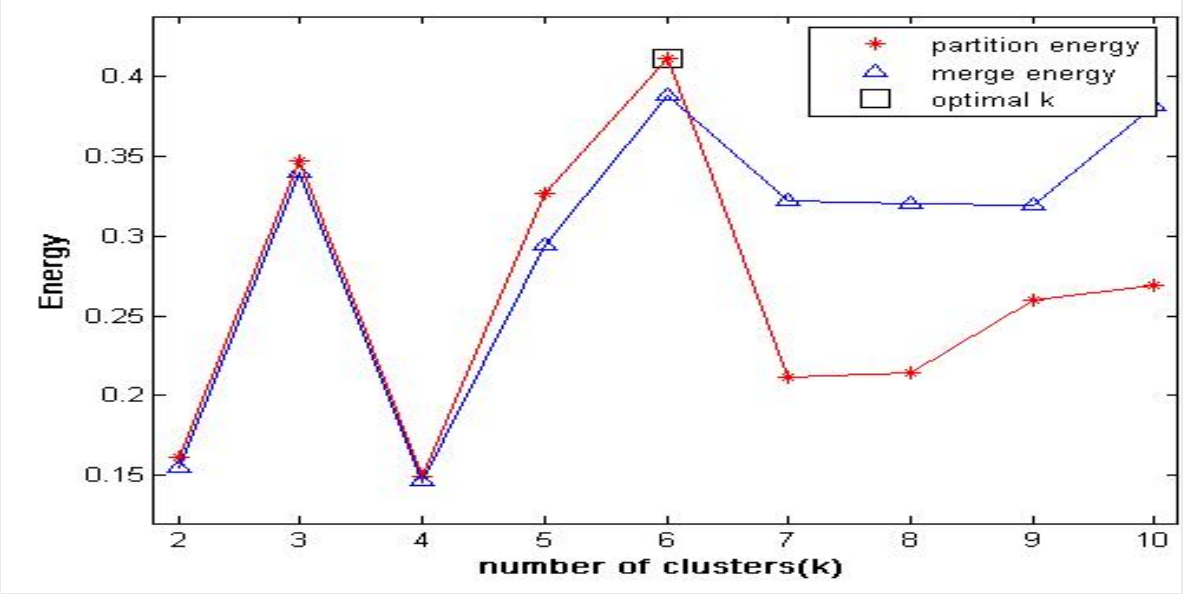


2ª Iteração

- Minimiza a soma dos quadrados das distâncias entre os indivíduos.

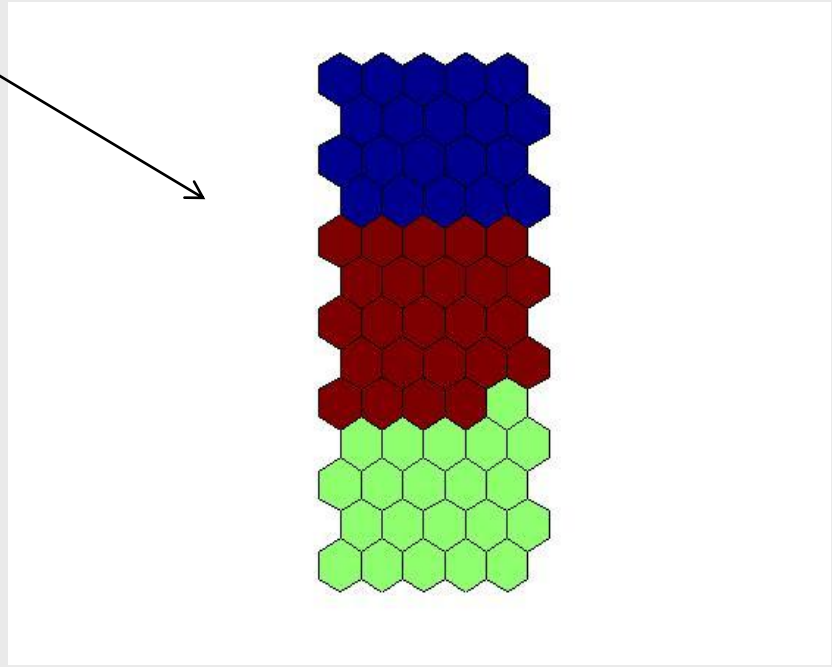
K-means com Kohonen

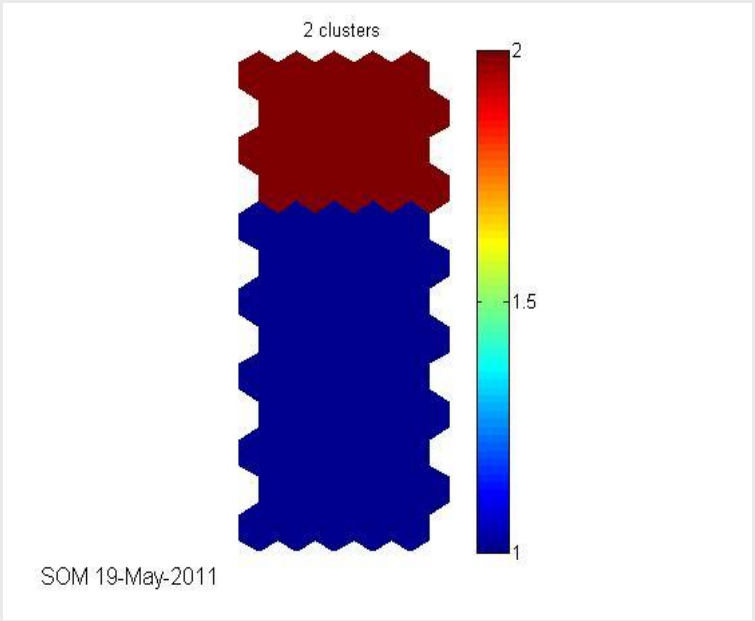




K-means com Kohonen

[c,p,err,ind] = kmeans_clusters(sM, 3, 10000);
som_show(sM,'color',{p{3},sprintf('%d clusters',3)});

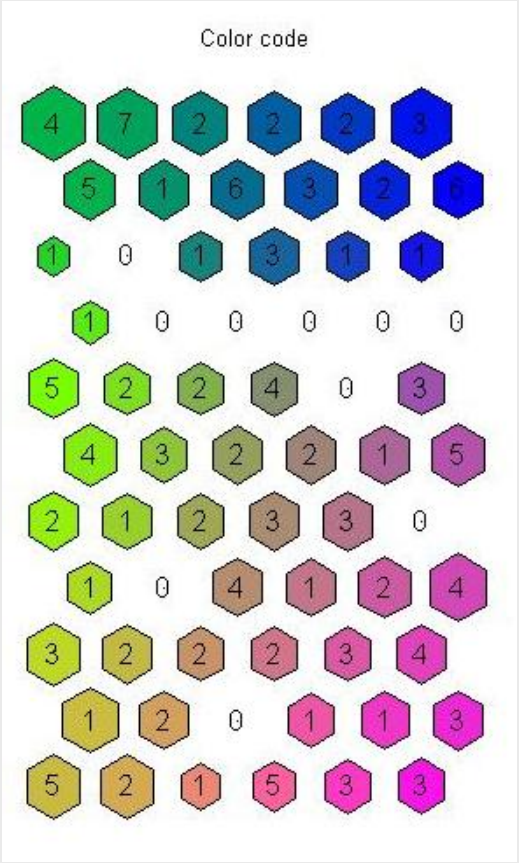




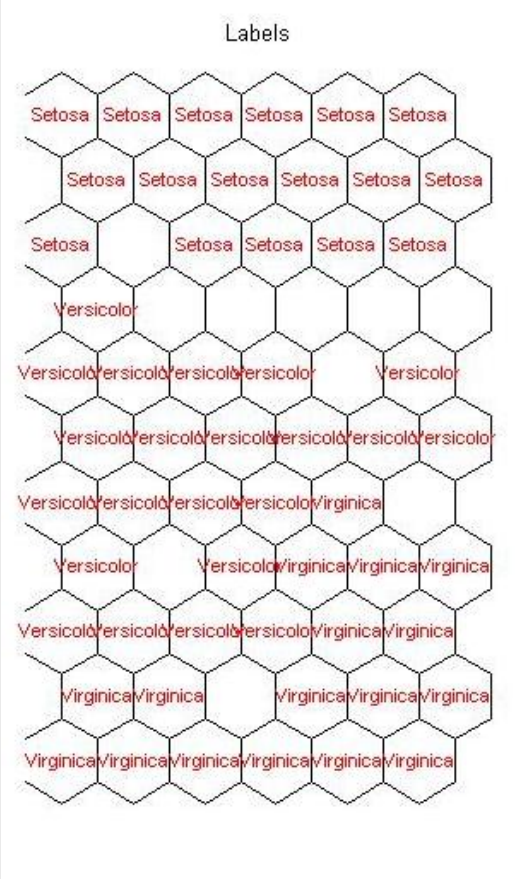
[c,p,err,ind] = kmeans_clusters(sM, 3, 1000);
[dummy,i] = min(ind);
som_show(sM,'color',{p{i},sprintf('%d clusters',i)});

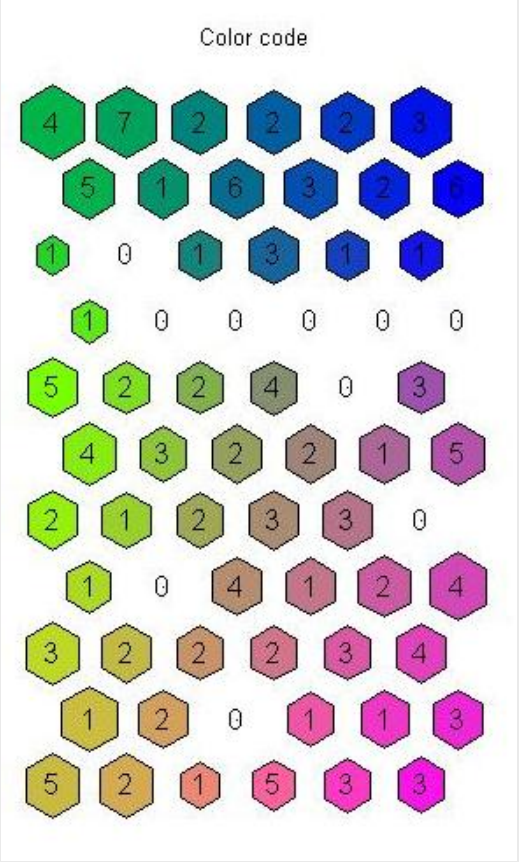
K-means com Kohonen

```
som_cplane(sM,Code,Dm);
hold on
som_grid(sM,'Label',cellstr(int2str(hits)),...
'Line','none','Marker','none','Labelcolor','k');
hold off
title('Color code')
```



K-means com Kohonen





Labels **K-means**

Dúvidas