

Cartes Topologiques : séance 1

Chargement de la SOM Toolbox

- Cliquez sur le lien suivant :

<http://www.cis.hut.fi/projects/somtoolbox/download/>

pour arriver à la page suivante :

Downloads and update logs
SOM Toolbox 2.0 downloadables

Note: Some advanced visualizations/GUIs do not work in the newest MATLABs (6.5-)

Item (date)	Files
SOM Toolbox 2.0 (Mar 17 2005)	somtoolbox2_Mar_17_2005.zip [417kb]
On-line (HTML) manual (Aug 2003)	somtoolbox_doc_Aug_11_2003.zip [821kB]
Manual and tutorial (April 2000)	PDF [3Mb] Both PS and PDF (zipped) [760kB]

- Chargez les 2 fichiers suivants:
 1. [somtoolbox2_Mar_17_2005.zip](#) (voir la figure ci-dessus)
 2. La documentation : **Manual and tutorial (April 2000)** (voir la figure ci-dessus)
- Décompresser le fichier **somtoolbox2_Mar_17_2005.zip**
(sous linux : `unzip somtoolbox2_Mar_17_2005.zip`)
- Vous obtenez alors un répertoire **somtoolbox** contenant les fonctions du logiciel

Utilisation de SOM Toolbox avec octave

- Octave est compatible avec Matlab et libre/open-source
- Normalement, la version modifiée (pour octave) est disponible à l'adresse suivante :

<https://github.com/limosek/somtoolbox>

- Une autre possibilité (pour utiliser SOM Toolbox avec **octave**) est de procéder comme suit :
 - Créer un répertoire **somtoolboxOctave**
 - Copier tous les fichiers contenus dans le répertoire **somtoolbox** dans **somtoolboxOctave**
 - Faire les changements suivants dans le répertoire **somtoolboxOctave**

1. Dans les fichiers suivants, remplacer tous les **'&'** en **'&&'** et tous les **'|'** en **'||'**

som_batchtrain.m
som_bmus.m
som_connection.m
som_grid.m
som_map_struct.m
som_neighborhood.m
sompak_train.m
som_randinit.m
som_seqtrain.m
som_set.m
som_topol_struct.m
som_train_struct.m
som_unit_coords.m
som_unit_dists.m
som_unit_neighs.m
vis_valuetype.m

2. Commenter la ligne 525 du fichier **som_batchtrain.m** pour la désactiver (la transformer en commentaire en ajoutant **%** au début de la ligne). Sinon, dans le terminal, le programme commence à afficher les données et attend qu'on appuie sur **'f'** pour continuer.

Donc la ligne

```
fprintf(1,'\rTraining: %3.0f/ %3.0f s',elap_t,tot_t)
```

devient

```
%fprintf(1,'\rTraining: %3.0f/ %3.0f s',elap_t,tot_t)
```

Autres modifications (nouvelles)

Si vous utilisez octave, voici les changements à faire pour corriger d'autres erreurs.

- Dans **som_randinit.m** → **ligne 202** il ne faut pas changer le **'&'** en **'&&'**

```
inds = find(~isnan(D(:,i)) & ~isinf(D(:,i)));
```
- Dans **som_unit_neighs.m** → **ligne 160** il ne faut pas changer le **'&'** en **'&&'**

```
inds = find(Ud(i,:)<1.01 & Ud(i,:)>0); % allow for rounding error
```
- Dans **som_cllinkage.m** → **ligne 266** il faut remplacer **now** par **date()**

Comme ci-dessous

```
name = sprintf('Clustering of %s at %s',data_name,datestr(datenum(now),0));  
name = sprintf('Clustering of %s at %s',data_name,datestr(datenum(date(),0)));
```

- Ensuite, il faut utiliser **somtoolboxOctave** à la place de **somtoolbox**

Approximation de fonctions de densité

Les données

Utiliser les instructions suivantes pour générer les données Data1

```
n=300;  
D = rand(n,2);  
save Data1 D
```

Vous pouvez taper (sous Matlab ou Octave) les instructions suivantes (en noir, après le caractère `>>`)

`>>` signifie que Matlab attend vos instructions

```
>> n=300;  
>> D = rand(n,2);  
>> save Data1 D
```

Apprentissage

On va utiliser les données Data1 pour vérifier visuellement si les référents de la carte topologique obtenue par apprentissage approximent la densité de probabilité de ces données.

Exécuter les instruction suivantes (en noir, après le caractère `>>` qui signifie que Matlab attend vos instructions) une à une pour comprendre les différentes étapes de l'apprentissage.

- **Ajouter le chemin d'accès aux fonctions de la somtoolbox (ou somtoolboxOctave)**

```
>> addpath somtoolbox
```

→ si vous utilisez octave alors : `addpath somtoolboxOctave`
→ à modifier selon l'emplacement de somtoolbox (ou somtoolboxOctave)

- **Charger les données**

```
>> load Data1
```

→ à modifier selon l'emplacement de Data1

Définition de la structure des données

```
>> sData = som_data_struct(D,'name','donnees1');
```

→ Pour voir les différents champs de cette structure, taper :

```
>> sData
```

→ Pour plus de détails sur la commande `som_data_struct`, taper :

```
>> help som_data_struct
```

Initialisation de la structure de la carte topologique

```
>> msize = [6 6]; → choix d'une carte 2D de 6×6  
>> insize = size(sData.data,2); → dimension des données d'apprentissage
```

Rappels sur Matlab (et octave)

Si M est une matrice contenant n exemples, chacun de dimension m alors
→ `size(M,1)` donne le nombre de lignes de M (donc n , le nombre d'exemples)
→ `size(M,2)` donne le nombre de colonnes de M (donc m , la dimension des exemples)

```
>> lattice = 'rect'; → choix du maillage
```

→ voir Figure 1 ci-dessous

- maillage rectangulaire : `lattice = 'rect'`
- maillage hexagonale : `lattice = 'hexa'`

```
>> shape = 'sheet'; → choix de la forme de la structure du graphe
```

→ voir Figure 2 ci-dessous

- en forme de plan : `shape = 'sheet'`
- en forme de cylindre : `shape = 'cyl'`
- en forme de tore : `shape = 'toroid'`

→ Voir aussi la documentation (techrep.pdf) Manual and tutorial (April 2000) page 8

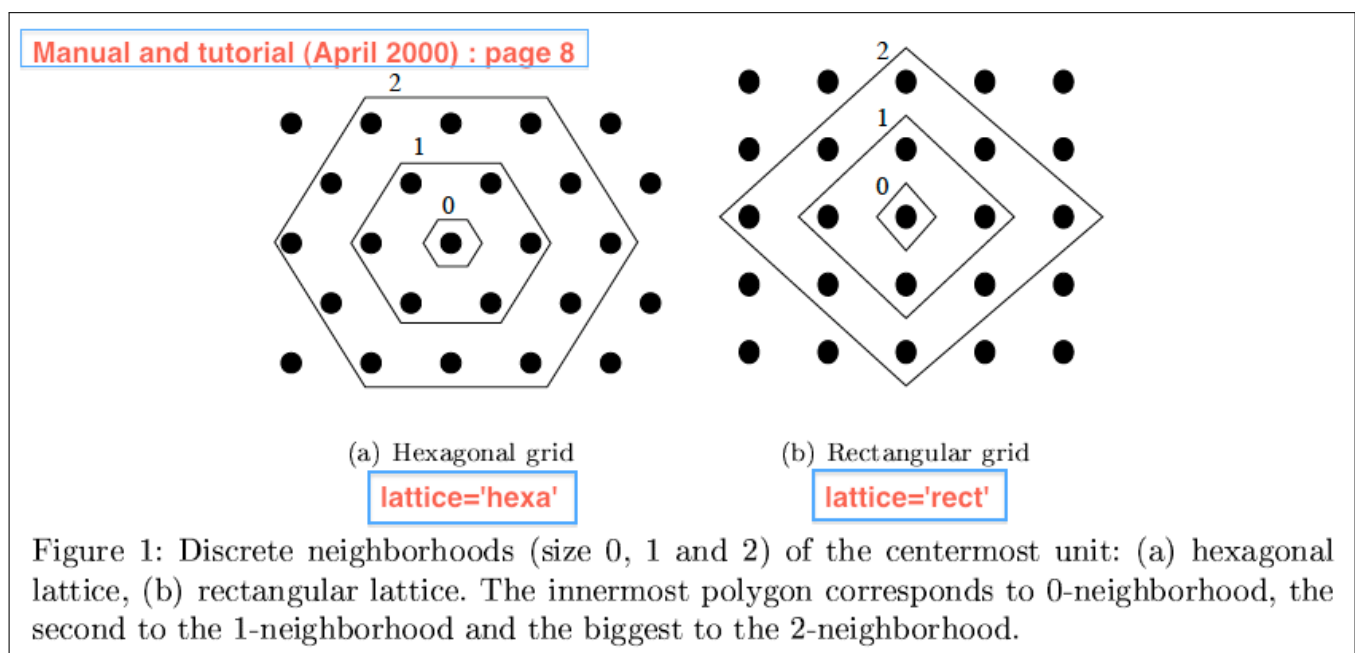
```
>> sMap = som_map_struct(insize,'msize',msize, lattice, shape);
```

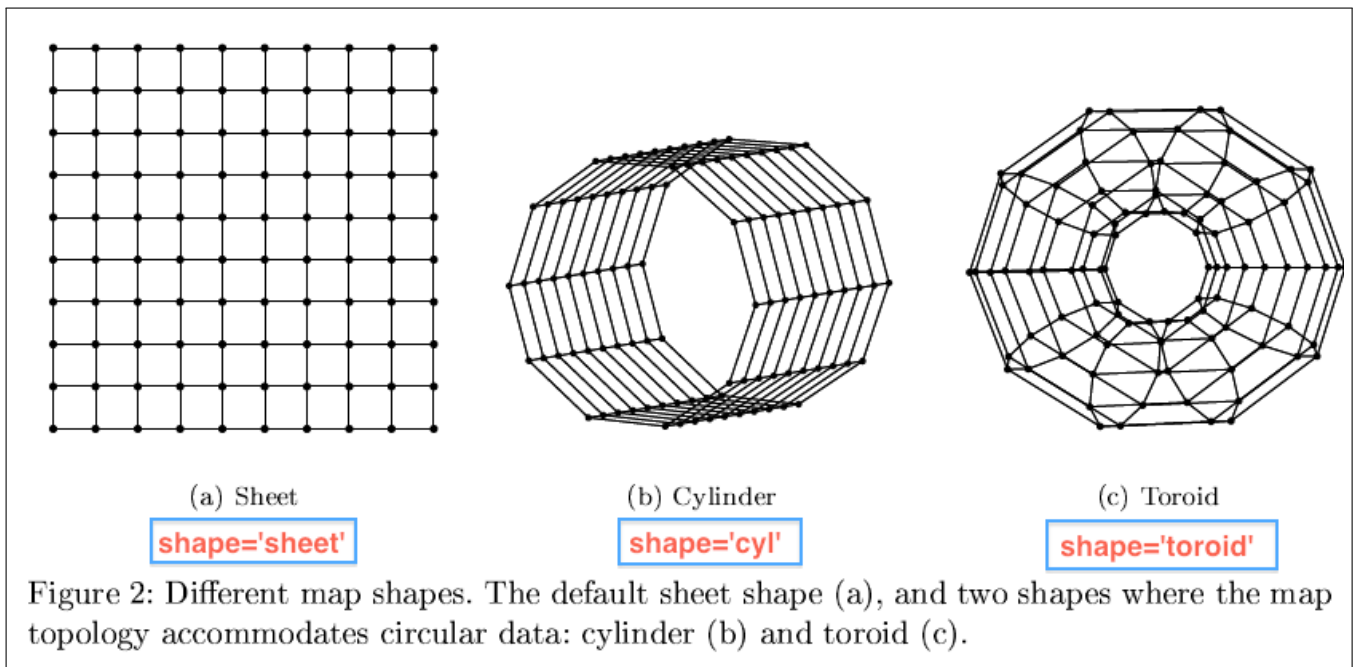
→ Pour voir les différents champs de cette structure, taper :

```
>> sMap
```

→ Pour plus de détails sur la commande `som_map_struct`, taper :

```
>> help som_map_struct
```





Initialisation des poids de la carte topologique (des vecteurs référents)

`>> sMap = som_randinit(sData, sMap);` → `som_randinit` : initialisation aléatoire

Visualisation des données et de la carte initiale

```
>> figure
>> plot(D(:,1),D(:,2),'b+');
>> hold on
>> som_grid(sMap,'Coord',sMap.codebook)
>> axis on
>> title('Données et structure de la grille');
```

On obtient la carte suivante :

Si la figure n'est pas très lisible, vous pouvez ajouter les paramètres suivants dans `som_grid` :

Pour les neurones (sommets) :

'`MarkerColor`' : pour la couleur
'`MarkerSize`' : pour la taille

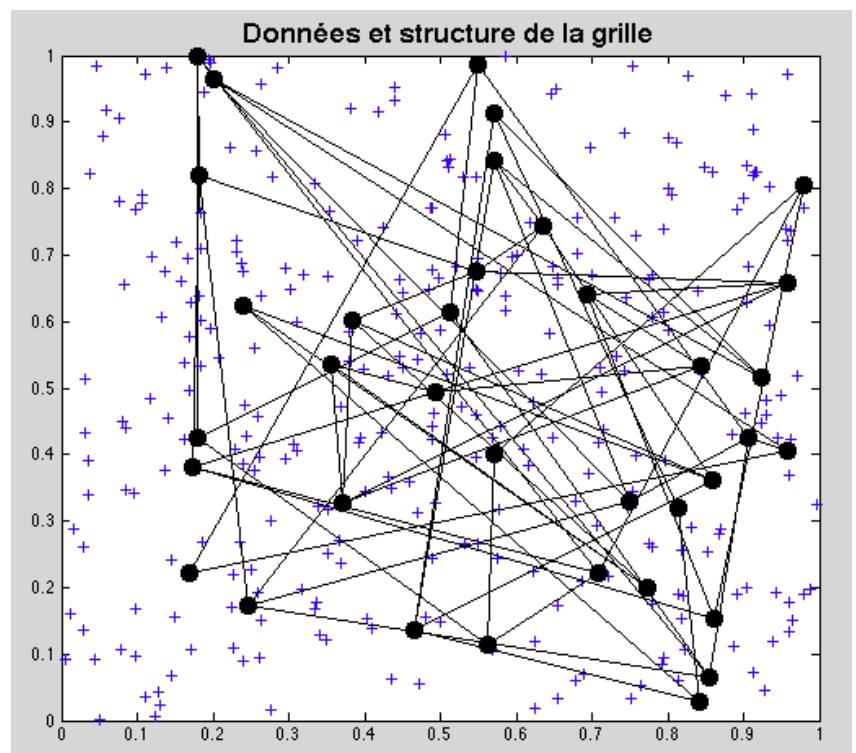
Pour les arêtes (lignes) :

'`LineColor`' : pour la couleur
'`LineWidth`' : pour la largeur

Exemple

Les sommets
couleur : rouge → `[1 0 0]`
taille : 7

Les lignes
couleur : bleu → `[0 0 1]`
largeur : 1



```
>> som_grid(sMap,'Coord',sMap.codebook,'MarkerColor',[1 0 0],'MarkerSize',7, 'LineColor',[0 0 1],'LineWidth',1)
```

Remarque : comme les poids de la carte (les vecteurs référents) sont initialisés aléatoirement, il y a beaucoup de neurones qui sont très proches au niveau du graphe (en utilisant la distance δ) mais très loin au niveau géométrique.

Essayer aussi l'initialisation **som_lininit** (qui tient compte des données) à la place de **som_randinit**

```
>> sMap = som_lininit(sData, sMap)
```

Pour plus de détails sur la commande **som_lininit**, taper :

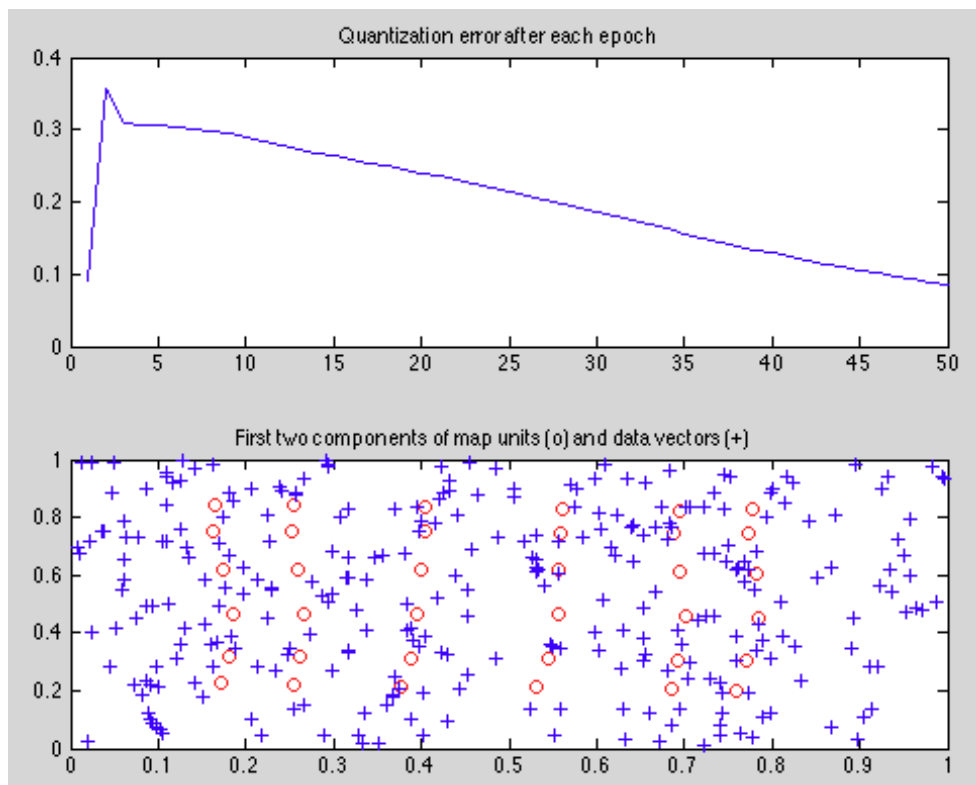
```
>> help som_lininit → pour voir les détails sur cette initialisation
```

Visualisation des données et de la carte initiale avec cette nouvelle initialisation

```
>> figure
>> plot(D(:,1),D(:,2),'b+');
>> hold on
>> som_grid(sMap,'Coord',sMap.codebook)
>> axis on
>> title('Données et structure de la grille');
```

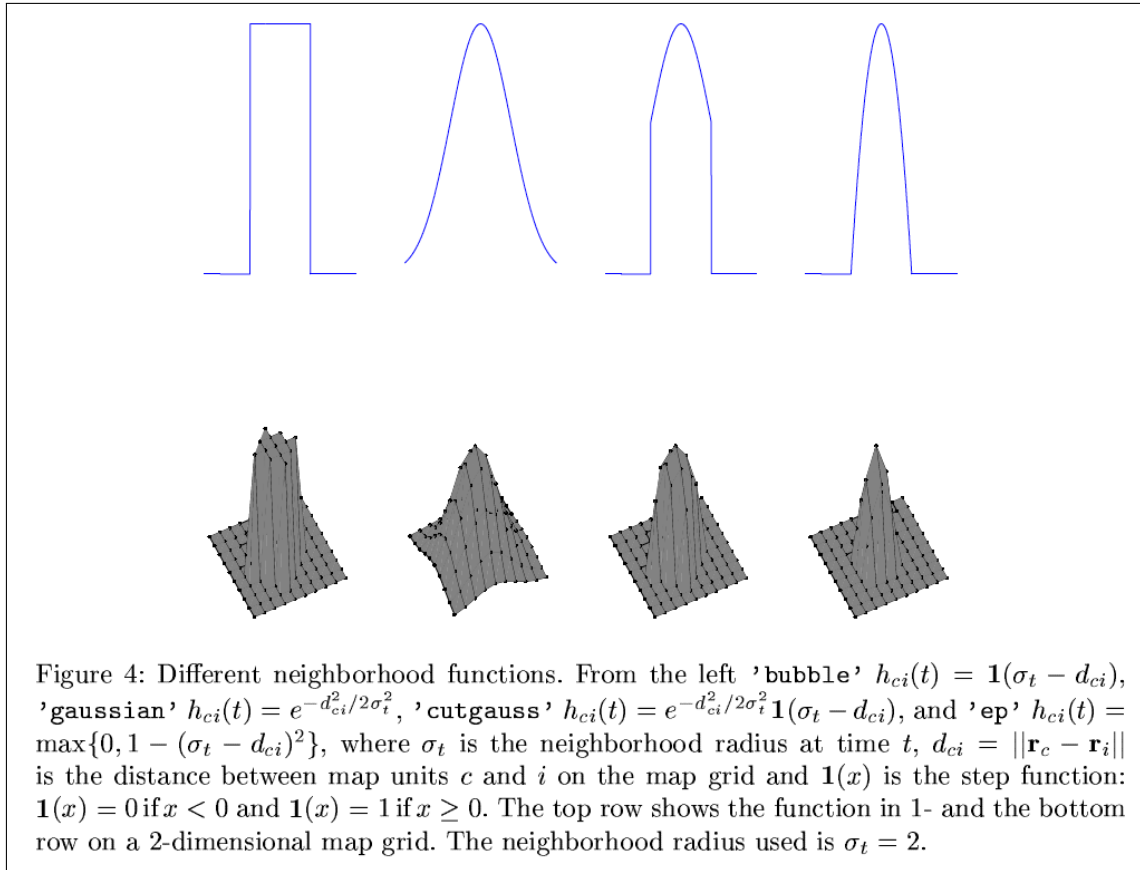
Entraînement de la carte : Phase 1 (Auto organisation)

```
>> figure
>> epochs = 50; → nombre d'itérations de la phase d'auto-organisation
>> radius_ini = 5; → valeur initiale de T (Tmax) (pour la phase d'auto-organisation)
>> radius_fin = 1; → valeur finale de T (Tmin) (pour la phase d'auto-organisation)
>> Neigh = 'gaussian'; → choix de la fonction de voisinage (voir figure 4 ci-dessous)
→ Neigh = 'gaussian', 'cutgauss', 'bubble' ou 'ep'
>> tr_lev = 3;
>> [sMap,sT] = som_batchtrain(sMap, sData,'trainlen',epochs, ... ← ne pas oublier ces 3 points
>> 'radius_ini',radius_ini,'radius_fin',radius_fin, 'neigh',Neigh,'tracking',tr_lev);
```



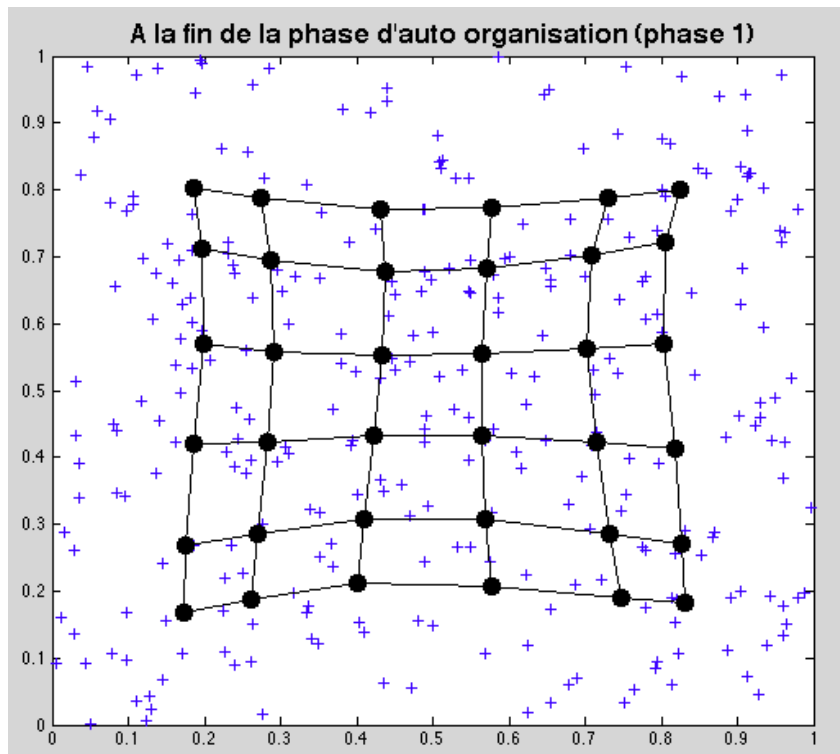
Exemple de fonction de voisinage : voir figure 4 suivante

Voir aussi la documentation ([techrep.pdf](#)) Manual and tutorial (April 2000) page 10



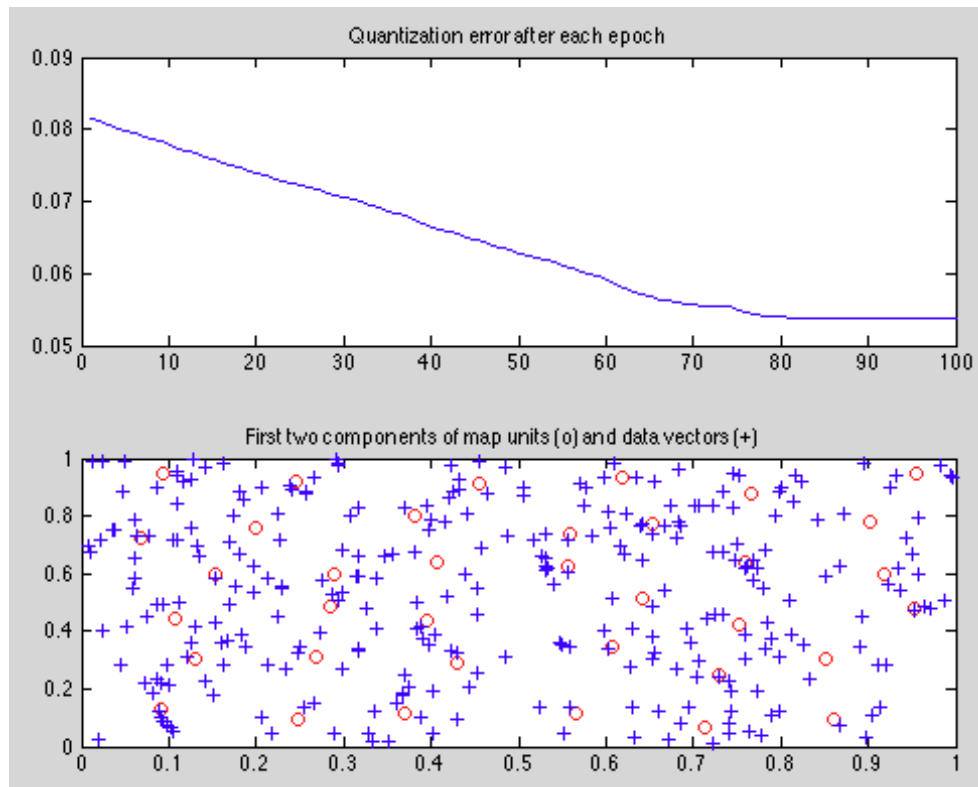
Visualisation des données et de la carte à la fin de la phase 1

```
>> figure
>> plot(D(:,1),D(:,2),'b+')
>> hold on
>> som_grid(sMap,'Coord',sMap.codebook), hold off, axis on
>> title('A la fin de la phase d'auto organisation (phase 1)');
```



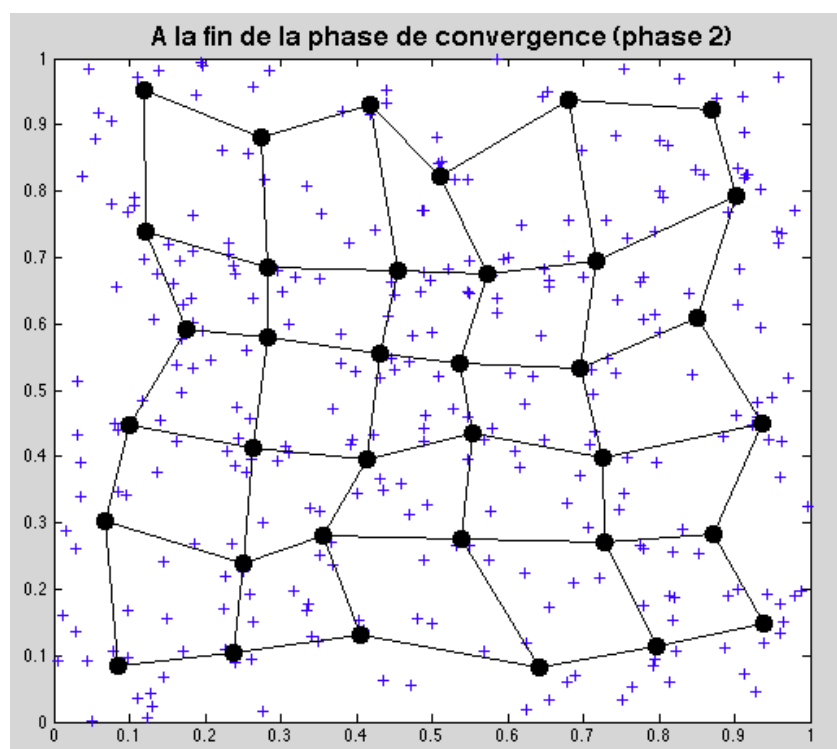
Entraînement de la carte : Phase 2 (Convergence)

```
>> epochs = 100; radius_ini = 1; radius_fin = 0.1;
>> figure
>> [sMap,sT] = som_batchtrain(sMap, sData,'trainlen',epochs, ... ← ne pas oublier ces 3 points
>> 'radius_ini',radius_ini,'radius_fin',radius_fin, 'neigh',Neigh,'tracking',tr_lev);
```



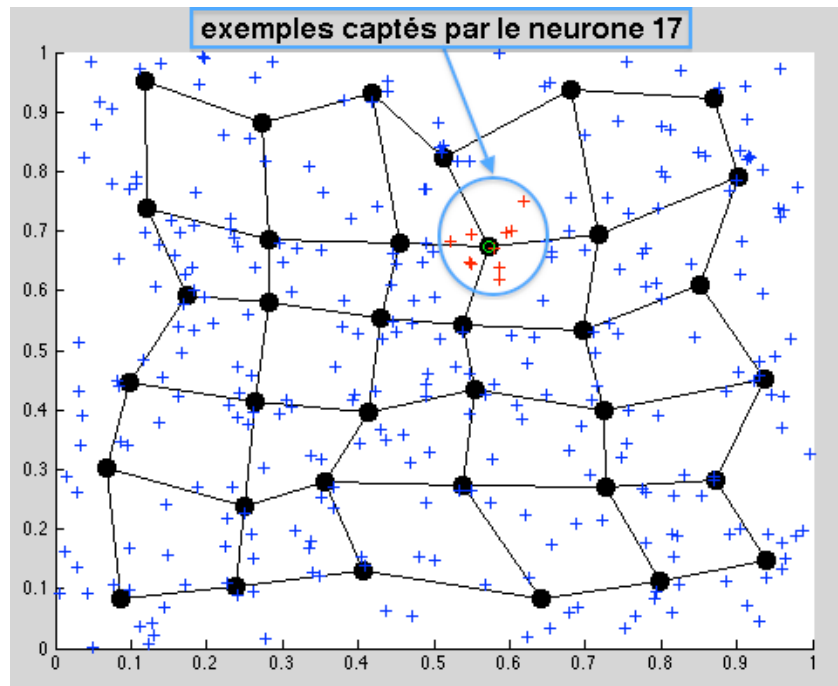
Visualisation des données et de la carte à la fin de la phase 2

```
>> figure
>> plot(D(:,1),D(:,2),'b+')
>> hold on
>> som_grid(sMap,'Coord',sMap.codebook)
>> title('A la fin de la phase de convergence (phase 2)');
```



Visualisation des exemples captés par un neurone

```
>> [Bmus, Qerrors] = som_bmus(sMap, sData);  
  
→Taper help som_bmus pour plus de détails  
→ Lire aussi page 42 de la documentation "Manual and tutorial (April 2000)"  
  
>> neur=17; % neurone choisi  
>> exemples_captés = find(Bmus == neur); % données captées par le neurone choisi  
>> figure  
>> som_grid(sMap,'Coord',sMap.codebook) % visualisation de la carte  
>> hold on  
>> plot(D(:,1),D(:,2),'b+') % données en bleu  
>> plot(D(exemples_captés,1),D(exemples_captés,2),'r+') % données captées par neur en rouge  
>> plot(sMap.codebook(neur,1),sMap.codebook(neur,2),'go') % neurone neur en vert  
>> title(['exemples captés par le neurone ' num2str(neur)]) % titre de la figure
```



Qualité de la carte

```
>> [qe_conv,te_conv]=som_quality(sMap,sData)
```

Pour plus de détails sur la commande **som_quality**, taper :

```
>> help som_quality
```

```
>> help som_quality
SOM_QUALITY Calculate the mean quantization and topographic error.

[qe,te] = som_quality(sMap, D)

qe = som_quality(sMap,D);
[qe,te] = som_quality(sMap,sD);

Input and output arguments:
sMap      (struct) a map struct
D          (struct) a data struct
           (matrix) a data matrix, size dlen x dim

qe         (scalar) mean quantization error
te         (scalar) topographic error

The issue of SOM quality is a complicated one. Typically two
evaluation criterias are used: resolution and topology preservation.
If the dimension of the data set is higher than the dimension of the
map grid, these usually become contradictory goals.

The first value returned by this function measures resolution and the
second the topology preservation.
qe : Average distance between each data vector and its BMU
te : Topographic error, the proportion of all data vectors
    for which first and second BMUs are not adjacent units.

NOTE: when calculating BMUs of data vectors, the mask of the given
      map is used.

For more help, try 'type som_quality' or check out the online documentation.
See also som\_bmus.

>>
```

BMU : Best-Matching Unit