

Cartes Topologiques (suite)

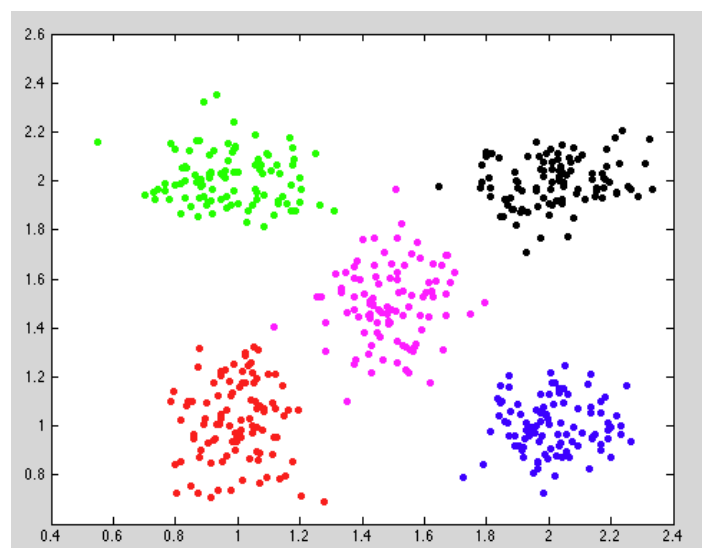
1 Problème de classification

Les données

- Charger le fichier [RCP208Data5Classes.zip](http://deptinfo.cnam.fr/new/spip.php?article851) depuis :
<http://deptinfo.cnam.fr/new/spip.php?article851>
Vous pouvez charger directement ce fichier en cliquant sur le lien suivant :
<http://deptinfo.cnam.fr/new/spip.php?pdoc6299>
- Décompresser ce fichier (sous linux : `unzip RCP208Data5Classes.zip`)
☞ Vous obtenez [Data2.mat](#) qui contient 4 variables :
 - *D* : les données
 - *classes* : les classes des données
 - *cnames* : les noms des variables
 - *labs* : les étiquettes des classes ou labels
- **Visualisation des données** : utiliser les instructions suivantes (sous matlab ou octave):

```
load Data2.mat
i1=find(classes==1);
i2=find(classes==2);
i3=find(classes==3);
i4=find(classes==4);
i5=find(classes==5);
figure
plot(D(i1,1),D(i1,2),'r.')
hold on
plot(D(i2,1),D(i2,2),'g.')
plot(D(i3,1),D(i3,2),'b.')
plot(D(i4,1),D(i4,2),'k.')
plot(D(i5,1),D(i5,2),'m.')
```

- On obtient la figure suivante :



Apprentissage

Initialisation de la structure de la carte topologique

```
>> addpath somtoolbox
>> sData = som_data_struct(D,'name','Donnees2','labels',labs,'comp_names',cnames);
Taper sData.comp_names pour voir son contenu
>> sData.comp_names → sData.comp_names contient cnames
Taper sData.labels pour voir son contenu
>> sData.labels → sData.labels contient labs
```

⇒ En utilisant les instructions vues dans la fiche de TP précédente, entraîner une carte

Labellisation des neurones de la cartes en utilisant le vote majoritaire

- Comme on le sait maintenant, à l'issu de l'apprentissage, chaque observation $\mathbf{z}_i$ est affectée à un neurone c selon la fonction d'affectation $\chi : c = \chi(\mathbf{z}_i)$.
 - On projette alors l'ensemble des données labellisées (étiquetées), chaque neurone c en récupère une partie.
 - Le neurone c est donc associé au sous-ensemble P_c constitué, pour partie, des données labélisées qui lui ont été affectées.
- ☞ Dès lors, pour déterminer la classe du neurone c , on peut utiliser (par exemple) un **vote majoritaire** qui consiste à attribuer au neurone c la classe qui apparaît le plus souvent parmi le sous-ensemble P_c . (Remarque : ici, dans notre exemple, toutes les données sont labellisées)
- Donc on va utiliser **le vote majoritaire** pour attribuer une classe à chaque neurone c . Le principe étant que le neurone prendra l'étiquette de la classe la plus représentée dans son sous ensemble P_c . Il est à noter qu'il est possible qu'un neurone n'ait capté aucune donnée (révélant ainsi des zones de l'espace mal connues) → dans ce cas aucune classe n'est attribuée au neurone.

A la fin de l'apprentissage, taper les instructions suivantes une à une pour les comprendre

Labellisation des neurones de la cartes en utilisant le vote majoritaire

```
>> Labl = cell(size(sMap.codebook,1),1);
>> Labl → pour voir le contenu de Labl
>> for ii = 1:size(sMap.codebook,1), Labl{ii} = num2str(ii); end
>> Labl → pour voir le nouveau contenu de Labl
>> sMap = som_label(sMap,'clear','all')
>> sMap.labels → pour voir son contenu
>> sMap = som_label(sMap,'add','all', Labl);
>> sMap.labels → pour voir son nouveau contenu
>> sMap = som_autolabel(sMap,sData,'vote'); → labellisation des neurones de la carte en utilisant
                                             le vote majoritaire
>> sMap.labels → pour voir son nouveau contenu
```

Visualisation du résultat obtenu par vote majoritaire

```
>> figure
>> som_show(sMap,'empty','numéro de chaque neurone et son étiquette');
>> som_show_add('label',sMap,'subplot',1);
```

On obtient la figure ci-dessous

numéro (ou indice) du neurone	n° de chaque neurone et son label obtenu par vote majoritaire						
	① deux	8 deux	15 deux	22	29 un	36 un	43 un
	2 deux	9 deux	16 deux	23 cinq	30 un	37 un	44 un
	3 deux	10 deux	17 cinq	24 cinq	31 cinq	38 un	45 un
	4 quatre	11 cinq	18 cinq	25 cinq	32 cinq	39 cinq	46 trois
	5 quatre	12 quatre	19 cinq	26 cinq	33 cinq	40 trois	47 trois
	6 quatre	13 quatre	20 quatre	27 cinq	34 trois	41 trois	48 trois
	7 quatre	14 quatre	21 quatre	28	35 trois	42 trois	49 trois

- Dans cette figure, chaque neurone contient deux informations :
→ son numéro (son indice dans la carte) et
→ son étiquette (ou label) obtenue suite au vote majoritaire.
- Pour plus de détails sur **som_label**, **som_autolabel** et **som_show**
→ utiliser la commande **help**

```
>> help som_label
>> help som_autolabel
>> help som_show
```

→ voir aussi documentation (techrep.pdf) **Manual and tutorial (April 2000)**

2 CAH sur les neurones d'une carte topologique

Reprendre les données **Datat2.mat**

A la fin de l'apprentissage, regrouper les référents de la carte en 5 classes en utilisant la CAH et comparer les résultats obtenus selon l'indice d'agrégation utilisé :

- indice du lien minimum (**single linkage**)
- indice du lien maximum (**complete linkage**)
- indice du lien moyen (**average linkage**)
- distance entre les centroïdes (**centroid method**)
- indice de Ward (**Ward method**)

Indications

On peut utiliser la fonction **som_clinkage.m** de la <<somtoolbox>>. Cette fonction permet de faire 10 tests en fonction des arguments **linkage** (indice d'agrégation) et **connect**. En effet, cette fonction propose :

- 5 choix pour l'argument **linkage** : **'single'**, **'complete'**, **'average'**, **'centroid'** et **'ward'**
- 2 choix pour l'argument **connect** (regroupements autorisés) : **'neighbors'** et **'any'**

'neighbors' : tient compte de la carte (du graphe).

→ On ne peut regrouper que deux neurones (groupes) adjacents sur la cartes.

'any' : ne tient pas compte de la carte (CAH classique)

→ donc $5 \times 2 = 10$ possibilités.

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

```
>> help som_cllinkage
```

Vous pouvez utiliser la commande Matlab **cluster** comme suit :

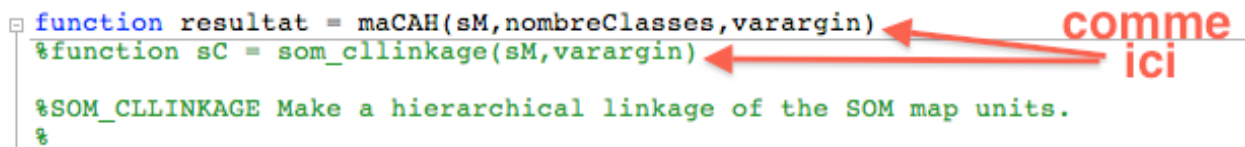
```
>> indice_agregation = 'single'; → 'single', 'average', 'complete', 'centroid' ou 'ward'
>> liens = 'neighbors'; → 'neighbors' ou 'any'
>> nbClasses = 5;
>> sC = som_cllinkage(sMap,'linkage',indice_agregation,'connect',liens);
>> LesClasses=cluster(sC.tree,nbClasses);
```

🔧 Si vous ne disposez pas de la toolbox contenant la commande **cluster**, vous pouvez créer une fonction **maCAH.m** comme suit :

Indication pour créer une fonction **maCAH.m**

1. Copier la fonction **som_cllinkage.m** de la <<somtoolbox>> dans un fichier **maCAH.m**
→ **Il ne faut pas modifier la fonction som_cllinkage.m** de la <<somtoolbox>>
2. Ouvrir le fichier **maCAH.m**
3. Remplacer (dans le fichier **maCAH.m**) la première ligne → **function sC = som_cllinkage(sM,varargin)**
par → **function resultat = maCAH(sM,nombreClasses,varargin)**

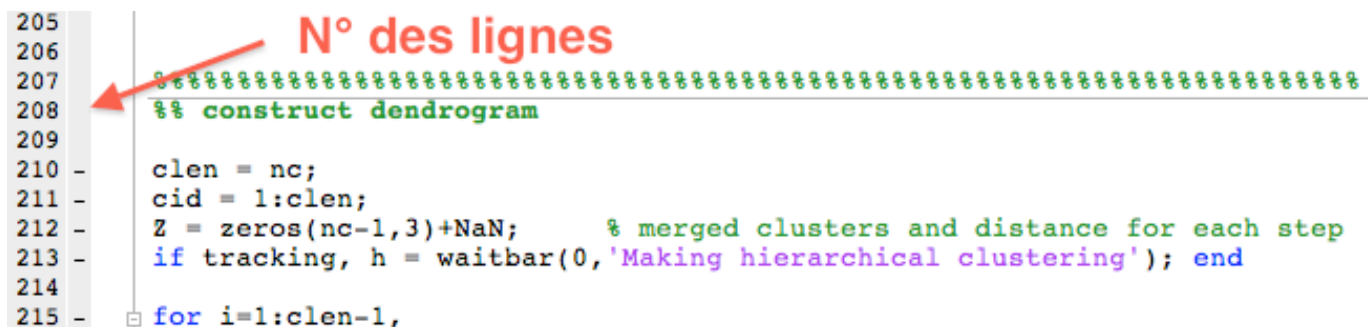
Comme indiqué sur la figure suivante :



```
function resultat = maCAH(sM,nombreClasses,varargin)
%function sC = som_cllinkage(sM,varargin)

%SOM_CLLINKAGE Make a hierarchical linkage of the SOM map units.
%
```

4. Chercher dans **maCAH.m** les instructions suivantes (elles se trouvent vers la fin) :

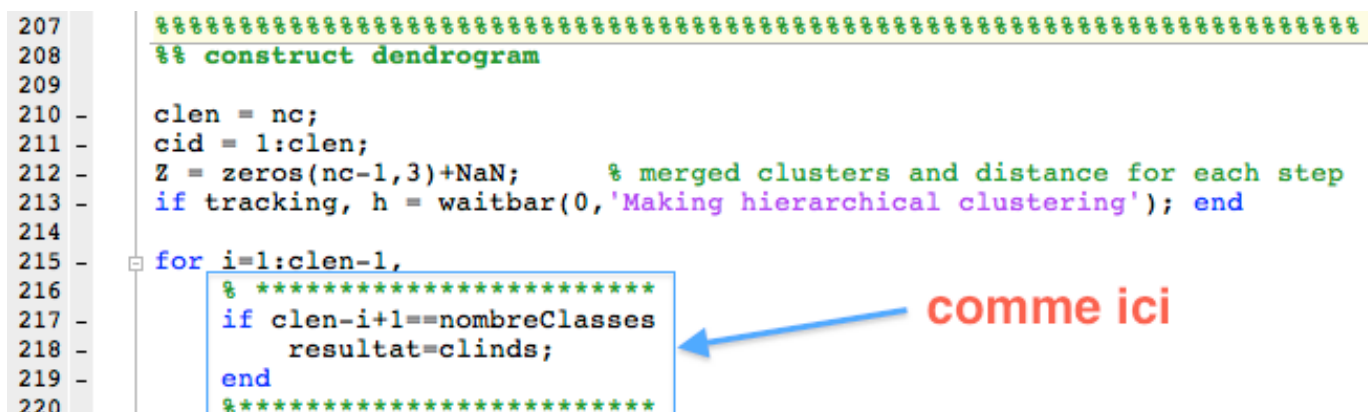


```
205
206
207 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
208 %% construct dendrogram
209
210 - clen = nc;
211 - cid = 1:clen;
212 - Z = zeros(nc-1,3)+NaN; % merged clusters and distance for each step
213 - if tracking, h = waitbar(0,'Making hierarchical clustering'); end
214
215 - for i=1:clen-1,
```

5. Ajouter (juste après ces instructions) les instructions suivantes :

```
if clen-i+1==nombreClasses
    resultat=clinds;
end
```

Comme indiqué sur la figure suivante :



```
207 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
208 %% construct dendrogram
209
210 - clen = nc;
211 - cid = 1:clen;
212 - Z = zeros(nc-1,3)+NaN; % merged clusters and distance for each step
213 - if tracking, h = waitbar(0,'Making hierarchical clustering'); end
214
215 - for i=1:clen-1,
216     % *****
217     if clen-i+1==nombreClasses
218         resultat=clinds;
219     end
220     % *****
```

- Tester la fonction **maCAH.m** comme suit :

On suppose que :

→ **sMap** est une carte obtenue par apprentissage

→ **msize** = [7 7] donc la carte contient 49 neurones

```
>> indice_agregation='ward'; → 'single', 'average', 'complete', 'centroid' ou 'ward'
>> liens = 'neighbors'; → 'neighbors' ou 'any'
>> nbClasses = 5;
>> LesClasses = maCAH(sMap,nbClasses,'linkage',indice_agregation,'connect',liens);
```

- Taper **LesClasses** pour voir son contenu.

```
>> LesClasses
```

```
LesClasses =
    [ 8x1 double]
    [10x1 double]
    [12x1 double]
    [10x1 double]
    [ 9x1 double]
```

Les référents de la carte sont regroupés en 5 classes :

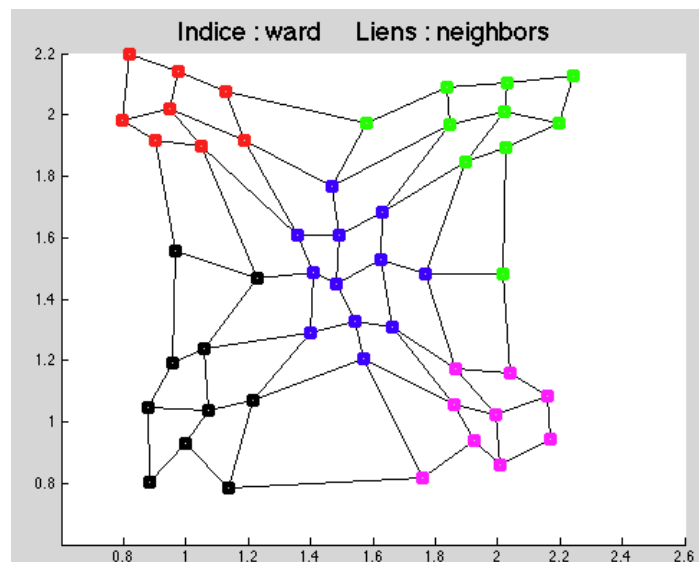
- ☞ La **classe 1** contient **8 neurones**
- ☞ La **classe 2** contient **10 neurones**
- ☞ La **classe 3** contient **12 neurones**
- ☞ La **classe 4** contient **10 neurones**
- ☞ La **classe 5** contient **9 neurones**

- Taper **LesClasses{1}** pour voir **les indices des 8 neurones de la classe 1**

```
>> LesClasses1
```

```
ans =
     1
     2
     9
     8
    15
    16
     3
    10
```

- Taper **LesClasses{2}**, **LesClasses{3}**, **LesClasses{4}** et **LesClasses{5}**
- On peut visualiser ces 5 classes sur la carte, comme le montre la figure suivante :
→ Pour la combinaison : **indice_agregation** = **'ward'** et **liens** = **'neighbors'**

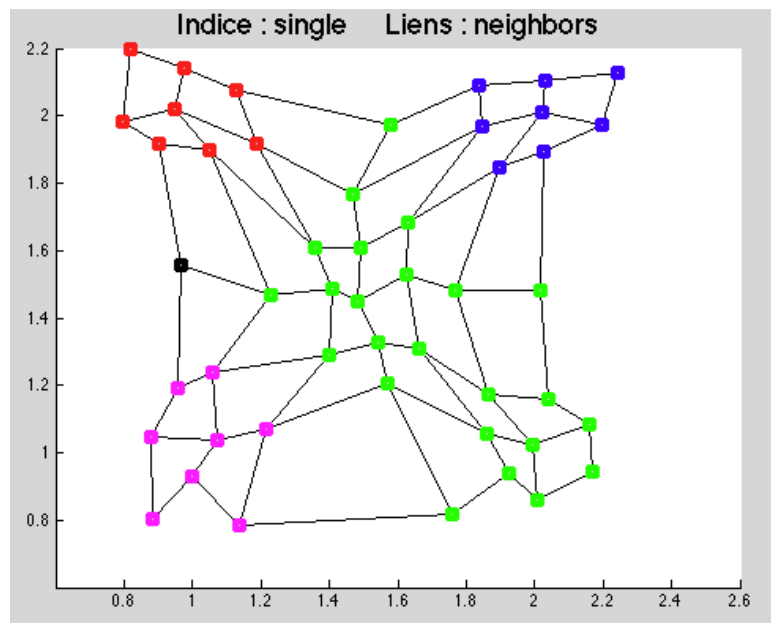


- Voici les instructions que j'ai utilisé pour visualiser les 5 classes :

```
figure
MesCouleurs={'ro', 'go', 'bo', 'ko', 'mo'};
som_grid(sMap,'Coord',sMap.codebook,'MarkerColor',[0.75 0.75 0.75],'LineColor',[0 0 0])
hold on
for j=1:nbClasses
    plot(sMap.codebook(LesClasses{j},1),sMap.codebook(LesClasses{j},2),MesCouleurs{j},...
        'LineWidth',3,'MarkerSize',6)
end
title(['Indice : ' indice_agregation '      Liens : ' liens'],'FontSize', 18)
```

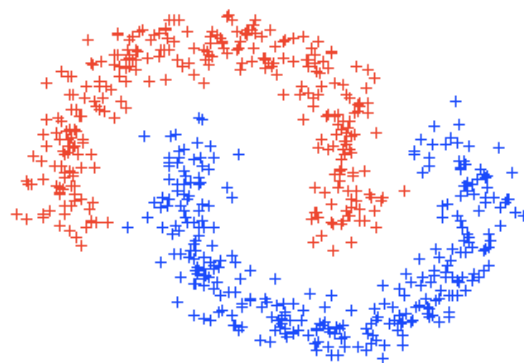
Remarque : les résultats de la CAH dépendent de la carte **sMap** obtenue par apprentissage

- Voici ce que donne la combinaison : **indice_agregation** = **'single'** et **liens** = **'neighbors'**



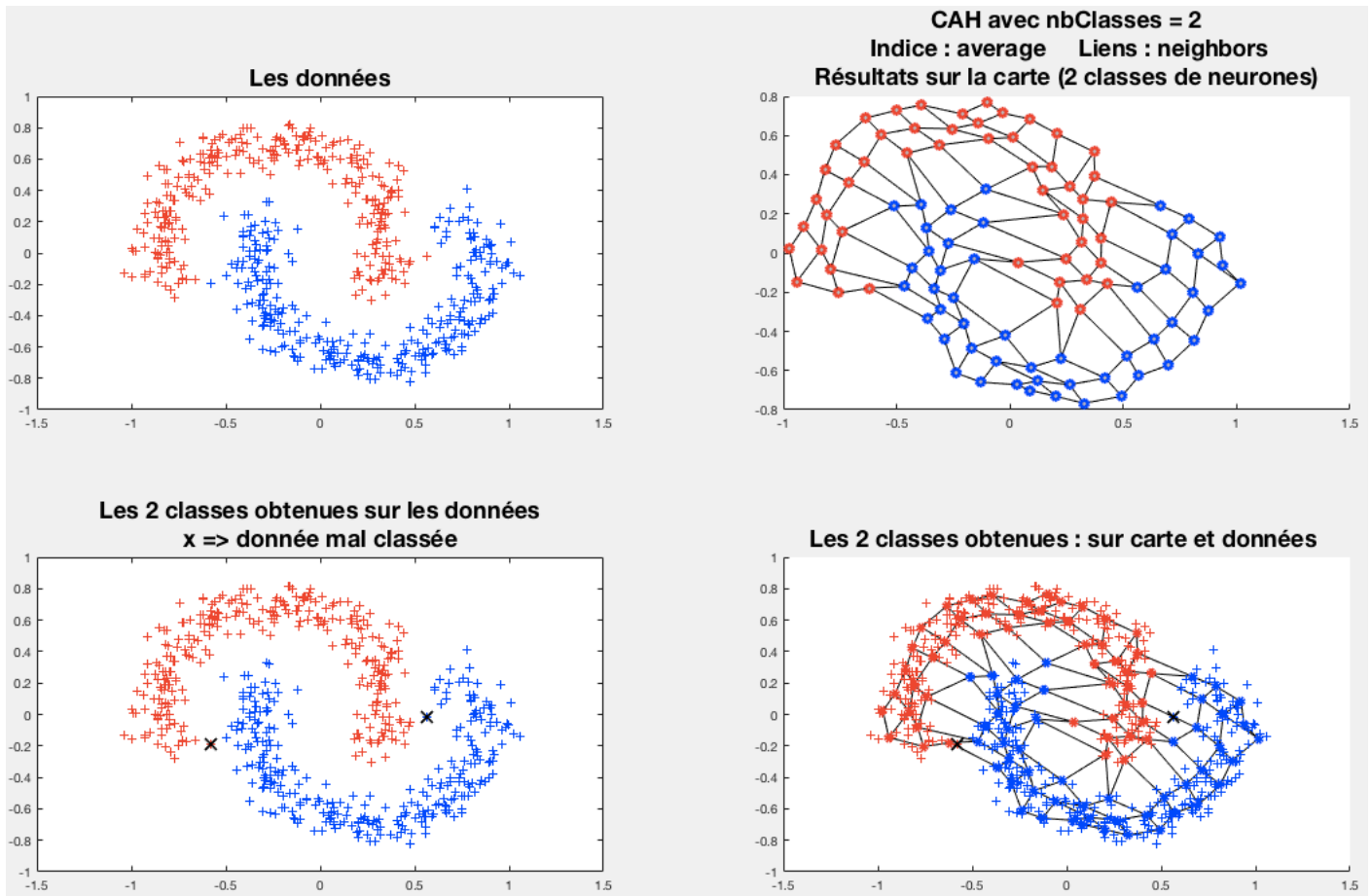
Quelques résultats sur un autre exemple (celui du cours) : 2 demi-cercles (2 classes)

L'objectif est de classer les données en 2 classes (rouge et bleu sur la figure ci-dessous) en utilisant les cartes topologiques et la CAH.

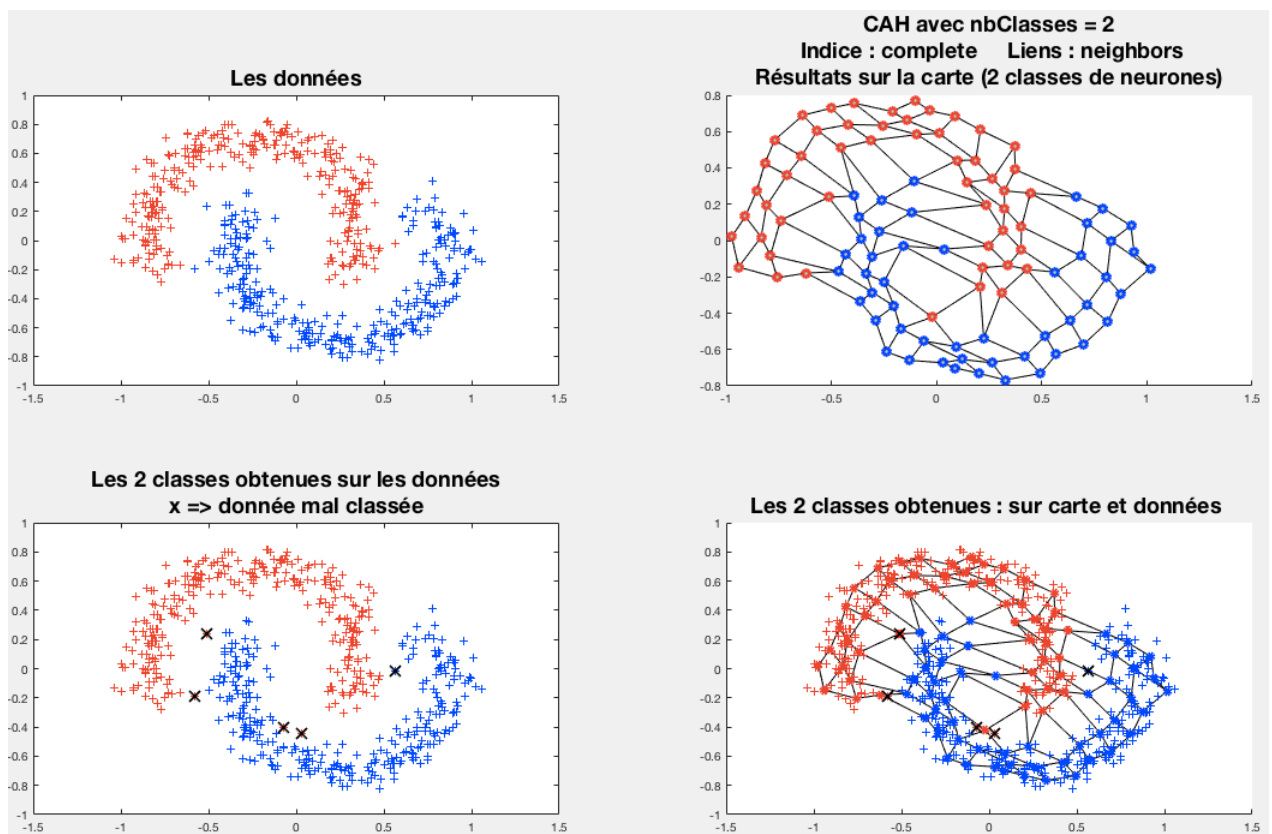


Voici les 2 combinaisons qui donnent de bons résultats (obtenus avec une carte topologique suivie d'une CAH avec nbClasses=2).

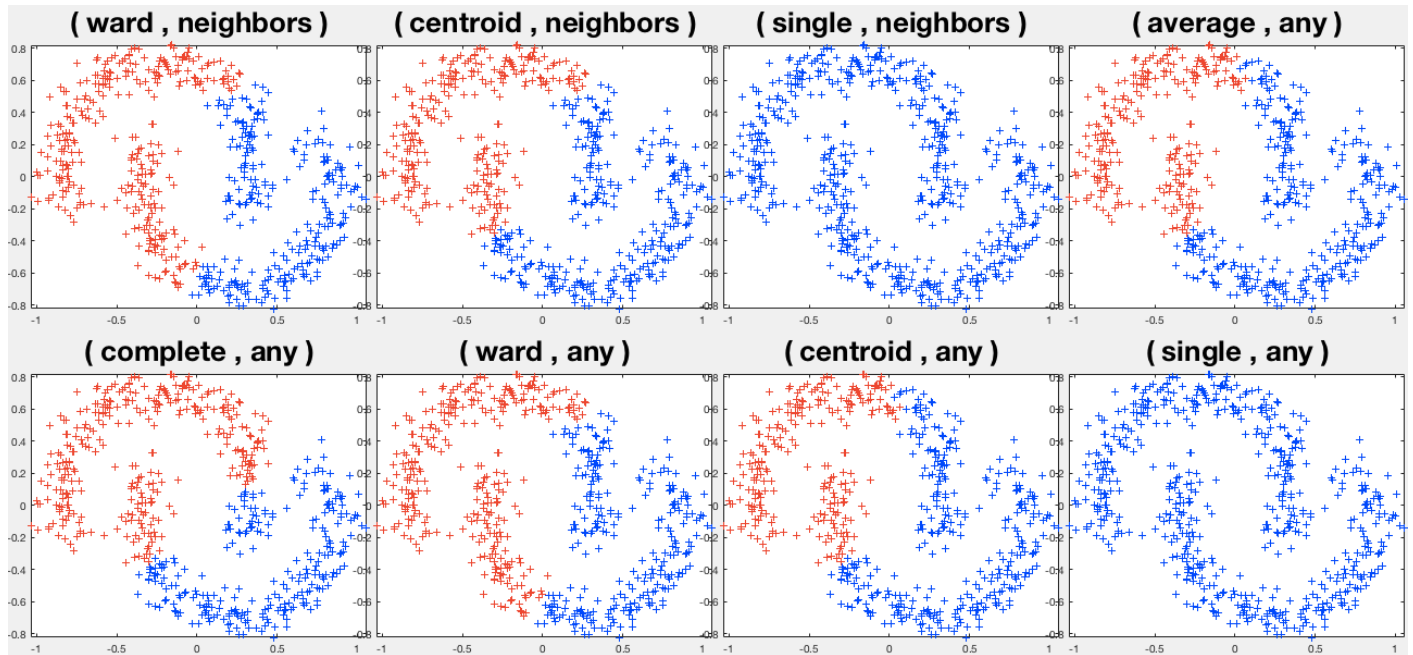
- Indice : average liens : neighbors



- Indice : complete liens : neighbors



Voici les résultats obtenus avec les autres combinaisons :



3 Problème réel : étude et classification des pixels d'une image sur l'Océan selon son spectre de couleur

L'étude de la couleur de l'océan aide à comprendre des processus de modélisation des constituants primaires de l'océan, qui sont à la base de la vie marine.

Le signal capté par un satellite en haut de l'atmosphère est fortement affecté par l'état de celle-ci. La présence d'aérosols -particules en suspension, sable, poussières- dans l'atmosphère modifie le spectre de couleur et oblige à traiter différemment les zones selon la prédominance d'un aérosol particulier.

Nous avons pour l'image d'étude, des étiquettes ou labels, proposées par un expert, permettant d'étiqueter la totalité de cette image. Ces labels nous indiquent si le pixel est nuage ou terre ou dans le cas contraire, quel type d'aérosols il contient. Il y a quatre types, mais dans la pratique, pour cette image, nous ne trouvons que trois : **aérosols désertiques**, correspondant pour la Méditerranée à des traînées de sable poussé par le vent saharien; **aérosols continentaux**, poussières et autres particules venues du continent; et **aérosols maritimes**, correspondant à l'atmosphère marine, relativement propre, se trouvant au-dessus des eaux claires.

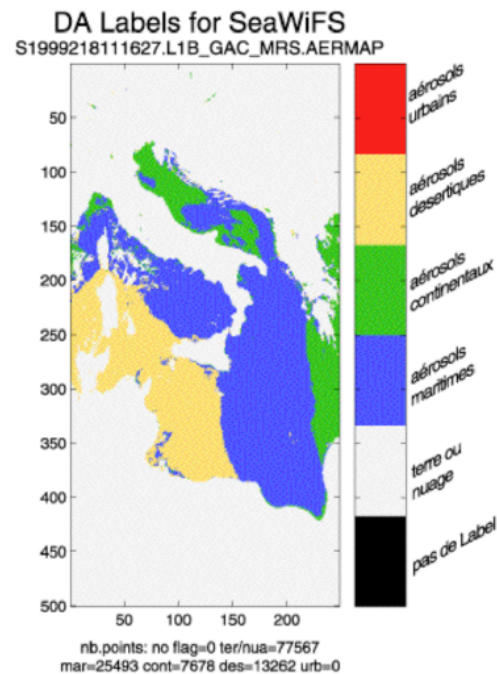
Les données

L'image d'étude est composée de 8 fichiers de données, contenant les réflectances mesurées pour 8 longueurs d'onde différentes :

L1Brad_rho_412.mat
L1Brad_rho_443.mat
L1Brad_rho_490.mat
L1Brad_rho_510.mat
L1Brad_rho_555.mat
L1Brad_rho_670.mat
L1Brad_rho_765.mat
L1Brad_rho_865.mat



Vue en vraies couleurs de la zone d'étude.



Étiquettes données par un expert selon leur contenu en aérosols.

La réflectance (ρ) est la quantité de radiation solaire réfléchi par un point. Sa valeur est dans l'intervalle $[0,1]$. Les 8 longueurs d'onde vont du 412 nm (correspondant à longueur d'onde de la couleur bleue) à 865 nm (correspondant au proche infrarouge). Les fichiers de réflectances ont été enregistrés en format Matlab binaire et contiennent chacun une matrice de 500 pixels de hauteur par 248 pixels de large. Vous les chargerez en utilisant la fonction Matlab **load**. La matrice stockée dans chaque fichier est déjà nommée : la commande

load L1Brad_rho_412 charge la variable **rho_412** qui est contenue dans le fichier **L1Brad_rho_412.mat**

Il y a un fichier **Labels_Aer_Map.mat** d'étiquettes données par un expert qui labellisent chacun des pixels de l'image. Les labels inclus signifient selon la valeur : **1** → correspond à un pixel sur terre ou nuage.

2 → aérosols maritimes

3 → aérosols continentaux

4 → aérosols désertiques

Théoriquement, il peut y avoir 0 (pas de label) ou 5 (aérosols urbain), mais ce n'est pas le cas sur cette image. Il y a également une série de fichiers contenant des flags ou masques proposés par le fournisseur des données. Un flag qui peut nous intéresser ici est celui du masque Terre. En utilisant ce masque vous pouvez enlever les points sur terre. **L2Flags LAND.mat** → masque terre (si pixel==1 alors terre)

Les données sont disponible à l'adresse suivant : <http://deptinfo.cnam.fr/new/spip.php?article1005>

Pour charger les données, il faut charger le fichier [Data_Classif_Ocean.tar.zip](#)

Vous pouvez charger directement ces données en cliquant sur le lien suivant :

<http://deptinfo.cnam.fr/new/spip.php?pdoc6569>

Ensuite il faut décompresser ce fichier :

→ sous linux : **unzip Data_Classif_Ocean.tar.zip** puis **tar xvf Data_Classif_Ocean.tar**

Vous obtenez alors un répertoire [Data_Classif_Ocean](#) contenant les données (**10 fichiers**).

Exemple de lecture des réflectances et des labels

Ci-dessous 2 versions :

- la première version contient des commentaires
- la deuxième version permet de faire "copier coller" (j'ai utilisé la commande verbatim sous latex).

Version 1 : avec les commentaires.

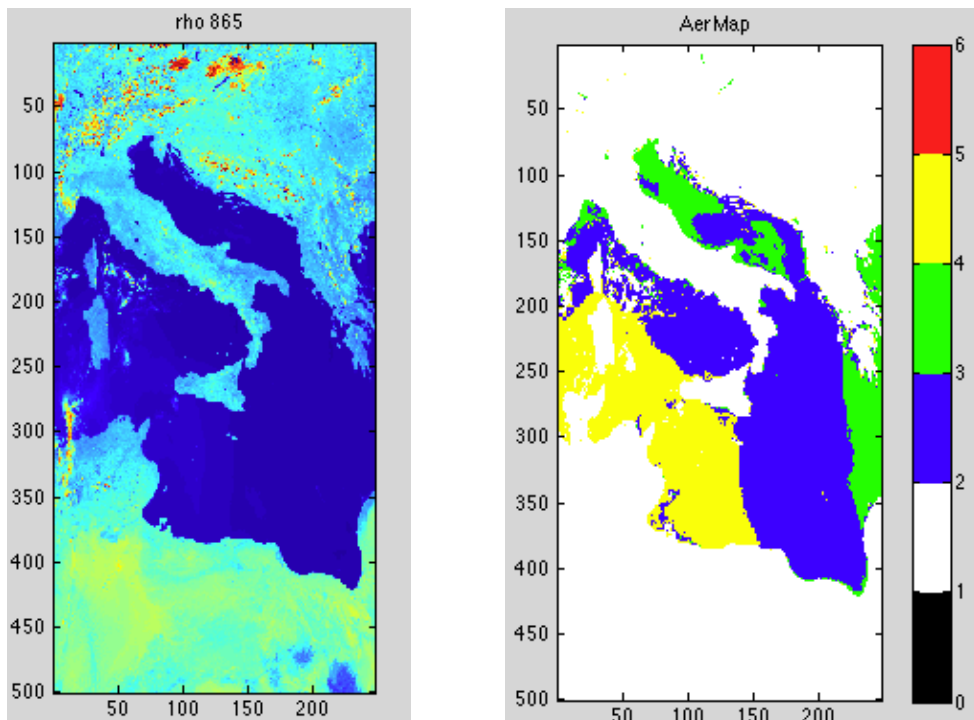
```
% Chemin d'accès aux données : à modifier selon l'emplacement de Data_Classif_Ocean
addpath Data_Classif_Ocean
% Exemple de lecture des réflectances
load L1Brad_rho_865;
figure
imagesc(rho_865);
axis image;
title('rho 865')
% Exemple de lecture des labels
load Labels_Aer_Map.mat
MesCouleurs = [0 0 0; 1 1 1 ; 0 0 1; 0 1 0; 1 1 0; 1 0 0];
figure
imagesc(Aer_Map,[0, 6]);
axis image;
colormap(MesCouleurs)
colorbar
title('Aer Map')
```

Version 2 : utile pour faire "copier coller" (version 1 sans les commentaires : problème avec les accents)

→ il vaut mieux faire des "copier-coller" ligne par ligne
et réécrire le caractère ' (comme dans title('rho 865')) s'il est en rouge.

```
addpath Data_Classif_Ocean;
load L1Brad_rho_865;
figure
imagesc(rho_865);
axis image;
title('rho 865')
load Labels_Aer_Map.mat
MesCouleurs = [0 0 0; 1 1 1 ; 0 0 1; 0 1 0; 1 1 0; 1 0 0];
figure
imagesc(Aer_Map,[0, 6]);
axis image;
colormap(MesCouleurs)
colorbar
title('Aer Map')
```

On obtient les 2 figures suivantes :



Exemple d'apprentissage

- **Base d'apprentissage** : pour constituer la base d'apprentissage
→ utiliser une ligne sur 2 de l'image et
→ utiliser le masque **L2Flags_LAND.mat** pour enlever les points sur terre.
- **Apprentissage** : dans cette exemple les paramètres d'apprentissage sont initialisés de façon approximative. Donc il faut faire d'autres apprentissages en modifiant ces paramètres

Ci-dessous 2 versions : la première version contient des commentaires et la deuxième version permet de faire "copier coller" (j'ai utilisé la commande verbatim sous latex).

Version 1 : avec les commentaires.

```
1 % Chemin d'accès aux données : à modifier selon l'emplacement de Data_Classif_Ocean
2 - addpath Data_Classif_Ocean
3 % Chemin d'accès à somtoolbox: à modifier selon l'emplacement de somtoolbox
4 - addpath somtoolbox
5 % Base d'apprentissage : on va utiliser une ligne sur 2 de l'image et
6 % on enlève les points sur terre
7 - load L2Flags_LAND.mat;
8 lin_incr = 2; lignesApp = 1:lin_incr:size(LAND,1);
9 MasqueTerreApp = LAND(lignesApp,:);
10 - load Labels_Aer_Map.mat;
11 AerMapApp = Aer_Map(lignesApp,:);
12 valid_pixels = find(MasqueTerreApp == 0); % Pixels de la base d'apprentissage
13 AerMapApp = AerMapApp(valid_pixels);
14 D=[];
15 - load L1Brad_rho_412; x=rho_412(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
16 - load L1Brad_rho_443; x=rho_443(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
17 - load L1Brad_rho_490; x=rho_490(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
18 - load L1Brad_rho_510; x=rho_510(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
19 - load L1Brad_rho_555; x=rho_555(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
20 - load L1Brad_rho_670; x=rho_670(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
21 - load L1Brad_rho_765; x=rho_765(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
22 - load L1Brad_rho_865; x=rho_865(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
23 - NomLabels = {'nuage', 'marit', 'cont', 'desert'};
24 % labels des données d'apprentissage (nuage, marit, cont et desert)
25 - labs = cell(length(valid_pixels),1);
26 % labels des données d'apprentissage sous forme numérique (1, 2, 3 et 4)
27 - labsNumerique=zeros(length(valid_pixels),1);
28 - for i_flag = 1:4
29 -     j_flag = find(AerMapApp == i_flag);
30 -     if ~isempty(j_flag);
31 -         labsNumerique(j_flag)=i_flag;
32 -         for jj = j_flag'
33 -             labs{jj} = NomLabels{i_flag};
34 -         end
35 -     end
36 - end
37 - cnames={'r412nm','r443nm','r490nm','r510nm','r555nm','r670nm','r765nm','r865nm'}';
38 - sD = som_data_struct(D,'name','CouleurOcean','labels',labs,'comp_names',cnames);
39 % Exemple d'apprentissage
40 - n_data = size(sD.data,1); insize = size(sD.data,2);
41 % Choix et initialisation de la carte
42 - msize = [10 10]; lattice = 'rect'; shape = 'sheet';
43 - sM = som_map_struct(insize,'msize',msize, lattice, shape);
44 - sM = som_lininit(sD, sM); % sM=som_randinit(sD, sM);
45 % Apprentissage : Phase 1 (auto-organisation)
46 - figure
47 - subplot(1,2,1)
48 - epochs = 40; radius_ini = 2.5; radius_fin = 1; Neigh = 'gaussian';
49 - tr_lev = 2;
50 - [sM,sT] = som_batchtrain(sM, sD,'trainlen',epochs,...
51 -     'radius_ini',radius_ini,'radius_fin',radius_fin, ...
52 -     'neigh',Neigh,'tracking',tr_lev);
53 - legend('Phase 1 : auto-organisation')
54 %Apprentissage : Phase 2 (convergence)
55 - subplot(1,2,2)
56 - epochs = 60; radius_ini = 1; radius_fin = 0.5;
57 - [sM,sT] = som_batchtrain(sM, sD,'trainlen',epochs,...
58 -     'radius_ini',radius_ini,'radius_fin',radius_fin, ...
59 -     'neigh',Neigh,'tracking',tr_lev);
60 - legend('Phase 2 : convergence')
61 % Qualité de la carte
62 - [qe,te]=som_quality(sM,sD);
63 - disp(['qe = ' num2str(qe)]); disp(['te = ' num2str(te)]);
```

```

addpath Data_Classif_Ocean
addpath somtoolbox
load L2Flags_LAND.mat;
lin_incr = 2; lignesApp = 1:lin_incr:size(LAND,1);
MasqueTerreApp = LAND(lignesApp,:);
load Labels_Aer_Map.mat;
AerMapApp = Aer_Map(lignesApp,:);
valid_pixels = find(MasqueTerreApp == 0);
AerMapApp = AerMapApp(valid_pixels);
D=[];
load L1Brad_rho_412; x=rho_412(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
load L1Brad_rho_443; x=rho_443(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
load L1Brad_rho_490; x=rho_490(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
load L1Brad_rho_510; x=rho_510(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
load L1Brad_rho_555; x=rho_555(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
load L1Brad_rho_670; x=rho_670(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
load L1Brad_rho_765; x=rho_765(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
load L1Brad_rho_865; x=rho_865(lignesApp,:); x=x(valid_pixels);D=[D,x]; clear x
NomLabels = {'nuage', 'marit', 'cont', 'desert'};
labs = cell(length(valid_pixels),1);
labsNumerique=zeros(length(valid_pixels),1);
for i_flag = 1:4
    j_flag = find(AerMapApp == i_flag);
    if ~isempty(j_flag);
        labsNumerique(j_flag)=i_flag;
        for jj = j_flag'
            labs{jj} = NomLabels{i_flag};
        end
    end
end
end
cnames={'r412nm','r443nm','r490nm','r510nm','r555nm','r670nm','r765nm','r865nm'}';
sD = som_data_struct(D,'name','CouleurOcean','labels',labs,'comp_names',cnames);
n_data = size(sD.data,1); insize = size(sD.data,2);
msize = [10 10]; lattice = 'rect'; shape = 'sheet';
sM = som_map_struct(insize,'msize',msize, lattice, shape);
sM = som_lininit(sD, sM);
figure
subplot(1,2,1)
epochs = 40; radius_ini = 2.5; radius_fin = 1; Neigh = 'gaussian';
tr_lev = 2;
[sM,sT] = som_batchtrain(sM, sD,'trainlen',epochs,...
    'radius_ini',radius_ini,'radius_fin',radius_fin, ...
    'neigh',Neigh,'tracking',tr_lev);
legend('Phase 1 : auto-organisation')
subplot(1,2,2)
epochs = 60; radius_ini = 1; radius_fin = 0.5;
[sM,sT] = som_batchtrain(sM, sD,'trainlen',epochs,...
    'radius_ini',radius_ini,'radius_fin',radius_fin, ...
    'neigh',Neigh,'tracking',tr_lev);
legend('Phase 2 : convergence')
[qe,te]=som_quality(sM,sD);
disp(['qe = ' num2str(qe)]); disp(['te = ' num2str(te)]);

```

Dans ce programme,

labs : contient les labels sous forme symbolique : 'nuage', 'marit', 'cont', 'desert'

labsNumerique : contient les mêmes labels que **labs** mais sous forme numérique : 1, 2, 3, 4
('nuage' → 1 'marit' → 2 'cont' → 3 'desert' → 4)

Voici les 4 premières valeurs de **labs** et **labsNumerique** :

```
>> labs(1:4)

ans =

    'cont'
    'marit'
    'cont'
    'desert'

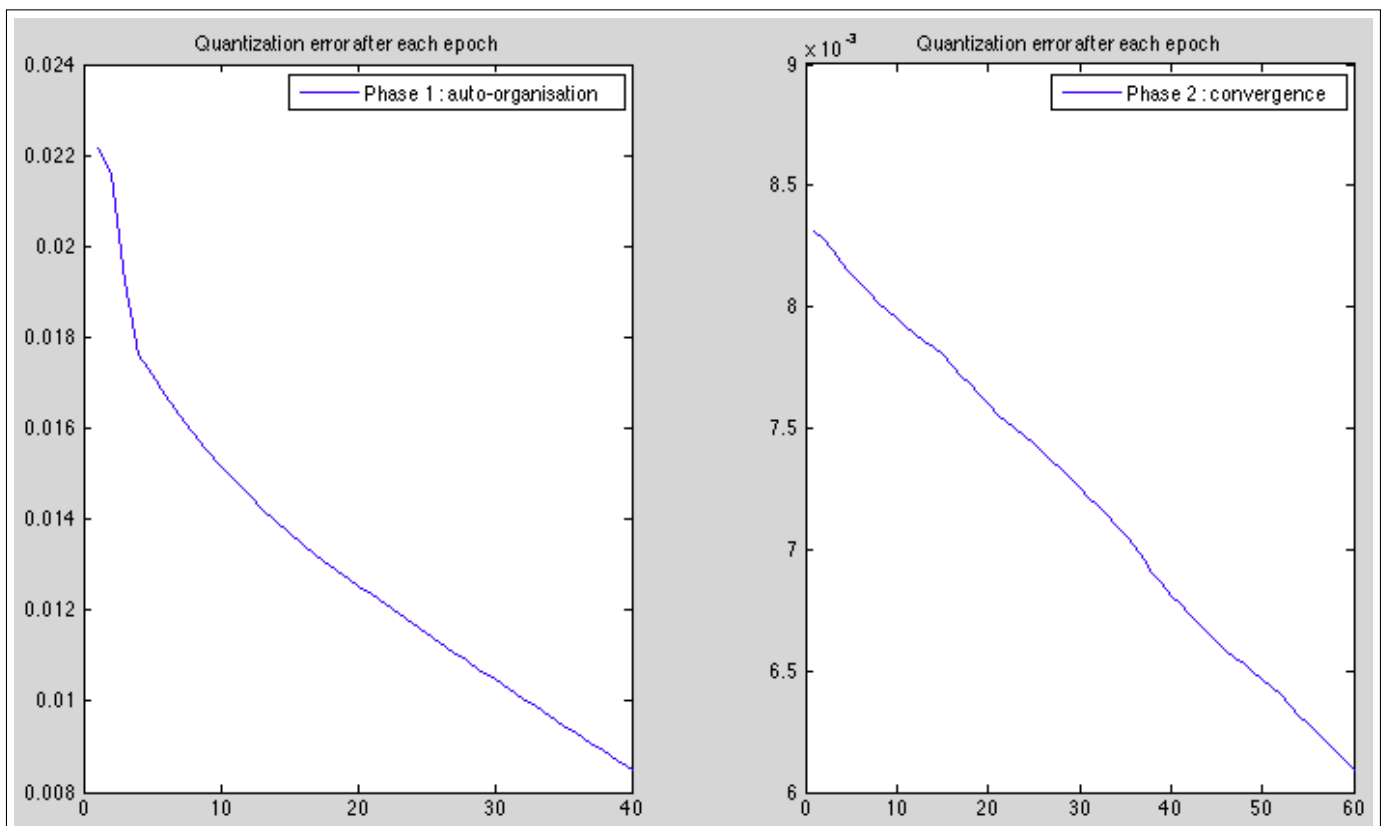
>> labsNumerique(1:4)

ans =

     3
     2
     3
     4

>>
```

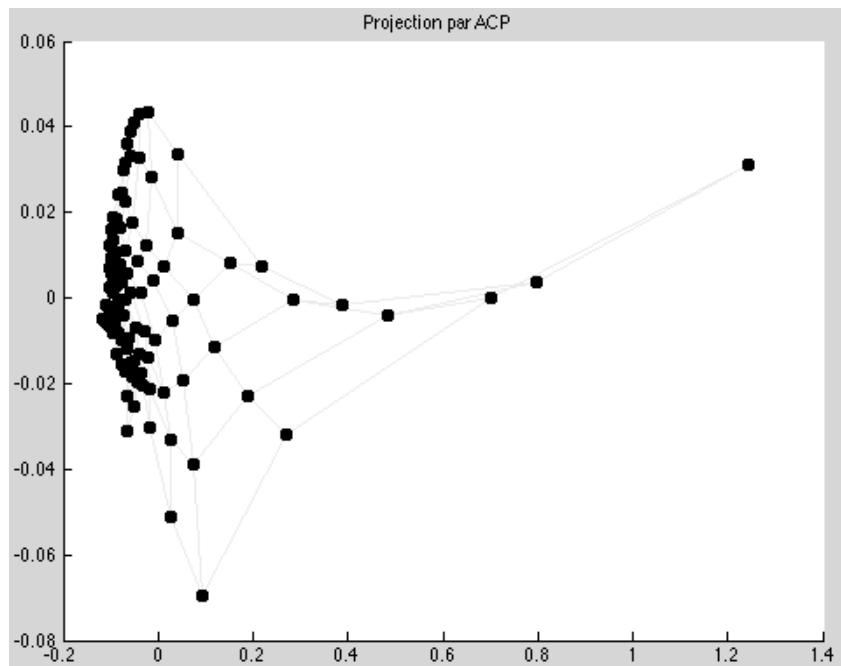
Voici les courbes d'apprentissage (on a utilisé l'instruction **subplot**)



Projection de la carte

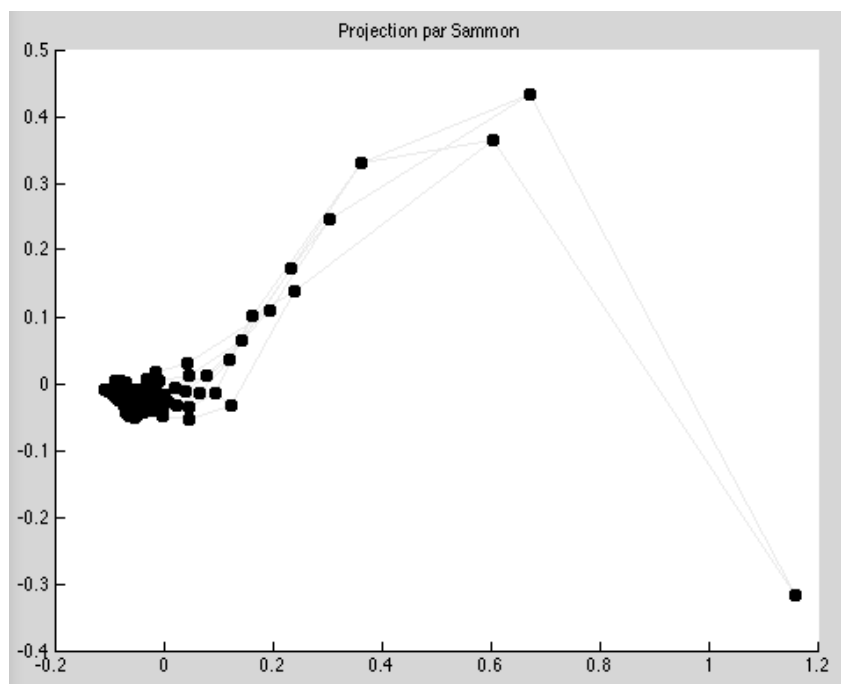
Projection de la carte par ACP : avec la commande `pcaproj`

```
PCAneurones=pcaproj(sM.codebook,2);  
figure  
som_grid(sM,'Coord',PCAneurones); axis on  
title('Projection par ACP');
```



Projection de la carte par Sammon : avec la commande `Sammon`

```
figure  
SammonNeurones=sammon(sM,2);  
som_grid(sM,'coord',SammonNeurones);axis on  
title('Projection par Sammon');
```



→ faire `help pcaproj`

→ faire `help sammon`

Voir ci-dessous la première page de l'article de **Sammon**

A Nonlinear Mapping for Data Structure Analysis

JOHN W. SAMMON, JR.

Abstract—An algorithm for the analysis of multivariate data is presented along with some experimental results. The algorithm is based upon a point mapping of N L -dimensional vectors from the L -space to a lower-dimensional space such that the inherent data "structure" is approximately preserved.

Index Terms—Clustering, dimensionality reduction, mappings, multidimensional scaling, multivariate data analysis, nonparametric, pattern recognition, statistics.

INTRODUCTION

THE purpose of this paper is to describe the nonlinear mapping algorithm (NLM) which has been found to be highly effective in the analysis of multivariate data. The analysis problem is to detect and identify "structure" which may be present in a list of N L -dimensional vectors. Here the word structure refers to geometric relationships among subsets of the data vectors in the L -space. Some examples of structure are hyperspherical and hyperellipsoidal clusters, and linear and certain nonlinear relationships among the vectors of some subset.

The algorithm is based upon a point mapping of the N L -dimensional vectors from the L -space to a lower-dimensional space such that the inherent structure of the data is approximately preserved under the mapping. The approximate structure preservation is maintained by fitting N points in the lower-dimensional space such that their interpoint distances approximate the corresponding interpoint distances in the L -space. We shall be primarily interested in mappings to 2- and 3-dimensional spaces since the resultant data configuration can easily be evaluated by human observations in 3 or less dimensions.

THE NONLINEAR MAPPING

Suppose that we have N vectors in an L -space designated X_i , $i = 1, \dots, N$ and corresponding to these we define N vectors in a d -space ($d = 2$ or 3) designated Y_i , $i = 1, \dots, N$. Let the distance¹ between the vectors X_i and X_j in the L -space be defined by $d_{ij}^* = \text{dist}[X_i, X_j]$ and the distance between the corresponding vectors Y_i and Y_j in the d -space be defined by $d_{ij} = \text{dist}[Y_i, Y_j]$.

Let us now randomly² choose an initial d -space configuration for the Y vectors and denote the configuration as follows:

$$Y_1 = \begin{bmatrix} y_{11} \\ \vdots \\ y_{1d} \end{bmatrix} \quad Y_2 = \begin{bmatrix} y_{21} \\ \vdots \\ y_{2d} \end{bmatrix} \quad \cdots \quad Y_N = \begin{bmatrix} y_{N1} \\ \vdots \\ y_{Nd} \end{bmatrix}$$

Next we compute all the d -space interpoint distances d_{ij} , which are then used to define an error E , which represents how well the present configuration of N points in the d -space fits the N points in the L -space, i.e.,

$$E = \frac{1}{\sum_{i < j} [d_{ij}^*]} \sum_{i < j}^N \frac{[d_{ij}^* - d_{ij}]^2}{d_{ij}^*} \quad (1)$$

Note that the error is a function of the $d \times N$ variables y_{pq} , $p = 1, \dots, N$ and $q = 1, \dots, d$. The next step in the NLM algorithm is to adjust the y_{pq} variables or equivalently change the d -space configuration so as to decrease the error. We use a steepest descent procedure to search for a minimum of the error (see Appendix I for further details).

SOME COMPUTER RESULTS

We have exercised the nonlinear mapping algorithm on several data sets in order to test and evaluate the utility of the program in detecting and identifying structure in data. Some of the results obtained for several different artificially generated data sets³ are reported for the case where $d = 2$. We have also run the algorithm on many real data sets and have achieved highly satisfactory results; however, for demonstration purposes it is useful to work with artificially generated data in order that we can compare our results with the known data structure. The test data sets were as follows.

1) *Straight Line Data*: These data consisted of nine points distributed along a line in a 9-dimensional space. The data points were spaced evenly along the line with an interpoint Euclidean distance of $\sqrt{9}$ units. The initial 2-space configuration was chosen randomly.

² For the purpose of this discussion it is convenient to think of the starting configuration as being selected randomly; however, in practice the initial configuration for the vectors is found by projecting the L -dimensional data orthogonally onto a d -space spanned by the d original coordinates with the largest variances.

³ One exception is data set 3 which is a classical data set. This data set was not artificially generated.

Manuscript received August 26, 1968; revised February 2, 1969. The author was with Rome Air Development Center, Griffiss AFB, Rome, N. Y. He is now with Computer Symbolic, Inc., Rome, N. Y.

¹ Any distance measure could be used; however, if we have no a priori knowledge concerning the data, we would have no reason to prefer any metric over the Euclidean metric. Thus, this algorithm uses the Euclidean distance measure.

Labelliser chaque neurone avec la classe et la cardinalité

```
[Bmus, Qerrors] = som_bmus(sM, sD);
Hits = som_hits(sM, sD);
NbClasses = length(NomLabels);
MatClass = zeros(size(sM.codebook,1),NbClasses);
for iDon = 1: size(sD.data,1)
    MatClass(Bmus(iDon),labsNumerique(iDon))=MatClass(Bmus(iDon),labsNumerique(iDon))+1;
end
sM = som_label(sM, 'clear','all');
for iCell = 1: size(sM.codebook,1)
    for iClass = 1: NbClasses
        if MatClass(iCell,iClass) ~= 0
            curname = NomLabels{iClass};
            cellLabels = [ curname(1:2) ' ' num2str(MatClass(iCell,iClass)) ];
            sM = som_label(sM,'add',iCell, cellLabels);
        end
    end
end
figure('Position',[80 80 600 600]);
h=som_show(sM,'empty','Cardinalite par classe');
hlbl=som_show_add('label',sM);
```

On obtient la figure suivante. Par exemple, le neurone encadré en rouge a capté :

- 1 pixel nuage
- 17 pixels aérosols maritimes
- 6 pixels aérosols continentaux
- 100 pixels aérosols désertiques

ma 669 co 3	nu 1 ma 359 co 3	ma 391	ma 293 co 1	ma 695	ma 627	ma 359	ma 176 co 21	ma 38 co 254	nu 1 co 163
ma 342 co 10 de 3	ma 417 co 7	ma 557 co 2	ma 558	ma 395 co 1	ma 321 co 1	ma 216 co 2	ma 110 co 85	ma 5 co 182	co 272
ma 457 co 21 de 2	ma 324 co 4	ma 437 co 7	ma 476 co 5	ma 402 co 7	ma 207 co 6	ma 187 co 26	ma 89 co 100	nu 1 ma 13 co 131	ma 1 co 331
ma 255 co 36 de 13	ma 278 co 1 de 89	ma 265 de 31	ma 296 co 101	ma 330 co 275	ma 140 co 271	ma 69 co 115	ma 80 co 50	ma 40 co 52	ma 3 co 313
ma 52 co 3 de 110	ma 91 de 141	ma 355 de 4	nu 3 ma 76 co 221	ma 211 co 134 de 1	ma 142 co 93 de 8	ma 72 co 13 de 13	ma 50 co 12 de 89	ma 26 co 59 de 9	nu 1 ma 11 co 187
ma 10 co 1 de 161	ma 90 de 270	ma 105 co 1 de 16	nu 2 ma 8 co 137	ma 70 co 28 de 19	ma 57 de 61	ma 24 co 2 de 251	nu 1 ma 17 co 6 de 100	ma 15 co 2 de 44	ma 15 co 23 de 8
ma 2 de 216	ma 67 de 240	ma 95 co 1 de 63	ma 31 co 4 de 125	ma 8 co 4 de 197	co 3 de 120	ma 2 co 2 de 143	nu 28 ma 9 co 6 de 109	nu 86 co 1	nu 86
ma 8 de 304	ma 10 de 297	ma 4 de 242	ma 4 de 183	ma 4 co 2 de 236	ma 3 co 2 de 221	nu 7 co 2 de 201	nu 111 ma 1 de 8	nu 82	nu 84
ma 17 de 114	ma 1 de 178	ma 3 de 484	ma 1 co 1 de 315	ma 3 co 3 de 84	co 1 de 147	nu 137 de 17	nu 76	nu 89	nu 56
ma 69 co 2 de 187	ma 15 co 4 de 46	ma 2 de 317	ma 5 co 1 de 234	nu 1 de 52	nu 32 de 9	nu 56	nu 31	nu 53	nu 75

Labellisation des neurones de la carte en utilisant le vote majoritaire sur chaque neurone

```
sM = som_label(sM, 'clear','all');
Labl = cell(size(sM.codebook,1),1);
for ii = 1:size(sM.codebook,1), Labl{ii} = num2str(ii); end
sM = som_label(sM,'add', 'all', Labl);
sM = som_autolabel(sM,sD,'vote');
figure('Position',[50 50 650 650]);
som_show(sM,'empty','numero du neurone et son label');
som_show_add('label',sM);
```

☞ Le neurone encadré en rouge porte maintenant le label **aérosols désertiques** car il a capté en majorité des aérosols désertiques : 100 pixels

numero du neurone et son label									
1 marit	11 marit	21 marit	31 marit	41 marit	51 marit	61 marit	71 marit	81 cont	91 cont
2 marit	12 marit	22 marit	32 marit	42 marit	52 marit	62 marit	72 marit	82 cont	92 cont
3 marit	13 marit	23 marit	33 marit	43 marit	53 marit	63 marit	73 cont	83 cont	93 cont
4 marit	14 marit	24 marit	34 marit	44 marit	54 cont	64 cont	74 marit	84 cont	94 cont
5 desert	15 desert	25 marit	35 cont	45 marit	55 marit	65 marit	75 desert	85 cont	95 cont
6 desert	16 desert	26 marit	36 cont	46 marit	56 desert	66 desert	76 desert	86 desert	96 cont
7 desert	17 desert	27 marit	37 desert	47 desert	57 desert	67 desert	77 desert	87 nuage	97 nuage
8 desert	18 desert	28 desert	38 desert	48 desert	58 desert	68 desert	78 nuage	88 nuage	98 nuage
9 desert	19 desert	29 desert	39 desert	49 desert	59 desert	69 nuage	79 nuage	89 nuage	99 nuage
10 desert	20 desert	30 desert	40 desert	50 desert	60 nuage	70 nuage	80 nuage	90 nuage	100 nuage

Matrice de confusion et pourcentage de bonne classification

```
% neurone gagnant de chaque pixel
Neuronedechaquepixel= som_bmus(sM,sD);
% classification des données selon la carte
Classifparcarte=sM.labels(Neuronedechaquepixel,2);
% pixels classés de la même façon que l'expert
correct= find(strcmp(Classifparcarte,sD.labels)==1);
% pourcentage de bonne classification
pourcentage=length(correct)*100/length(sD.labels);
% matrice de confusion
nbreclasse=length(NomLabels);
confuse=zeros(nbreclasse,nbreclasse);
for i=1:nbreclasse
    for j=1:nbreclasse
        confuse(i,j)=length(find(strcmp(sD.labels,NomLabels(i))==1 ...
                                & strcmp(Classifparcarte,NomLabels(j))==1));
    end
end
% resultats
disp(' ')
disp(['pourcentage de bonne classification : ' num2str(pourcentage) '%'])
disp('matrice de confusion')
disp(num2str(confuse))
..
```

Version utile pour faire "copier coller"

```
Neuronedechaquepixel= som_bmus(sM,sD);
Classifparcarte=sM.labels(Neuronedechaquepixel,2);
correct= find(strcmp(Classifparcarte,sD.labels)==1);
pourcentage=length(correct)*100/length(sD.labels);
nbreclasse=length(NomLabels);
confuse=zeros(nbreclasse,nbreclasse);
for i=1:nbreclasse
    for j=1:nbreclasse
        confuse(i,j)=length(find(strcmp(sD.labels,NomLabels(i))==1 ...
                                & strcmp(Classifparcarte,NomLabels(j))==1));
    end
end
disp(' ')
disp(['pourcentage de bonne classification : ' num2str(pourcentage) '%'])
disp('matrice de confusion')
disp(num2str(confuse))
```

On obtient les résultats suivants :

```
pourcentage de bonne classification : 89.3468%
matrice de confusion
1054      1      8      37
      1 11529     534     674
      1   945    2811      63
      34   262      17    6219
>>
```

Si vous voulez visualiser les résultats sous forme de figure (pas vraiment important), vous pouvez utiliser la fonction **PloterMatrice** ci-dessous (j'ai modifié très légèrement la fonction **plotmat.m** de Netlab). Il faut écrire les instructions dans un fichier **PloterMatrice.m** (même nom que la fonction)

Fichier **PloterMatrice.m**

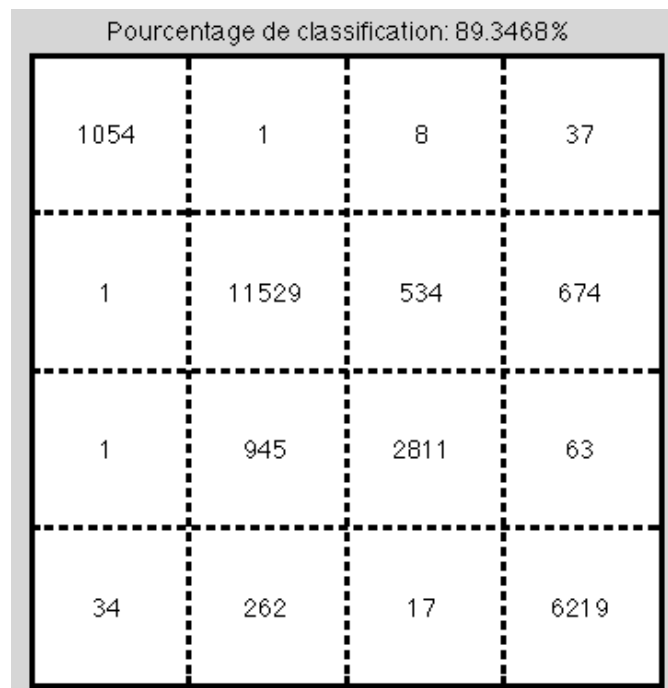
```
function PloterMatrice(M)
[m,n]=size(M);
for rowCnt=1:m,
    for colCnt=1:n,
numberString=num2str(M(rowCnt,colCnt));
text(colCnt-.5,m-rowCnt+.5,numberString, ...
    'HorizontalAlignment','center', ...
    'Color', 'k', ...
    'FontWeight','bold', ...
    'FontSize', 14);
    end;
end;

set(gca,'Box','on', ...
    'Visible','on', ...
    'xLim',[0 n], ...
    'xGrid','on', ...
    'xTickLabel',[], ...
    'xTick',0:n, ...
    'yGrid','on', ...
    'yLim',[0 m], ...
    'yTickLabel',[], ...
    'yTick',0:m, ...
    'DataAspectRatio',[1, 1, 1], ...
    'GridLineStyle',':', ...
    'LineWidth',3, ...
    'XColor','k', ...
    'YColor','k');
```

Ensuite vous pouvez l'utiliser comme suit :

```
figure
PloterMatrice(confuse)
title(['Pourcentage de classification: ' num2str(pourcentage) '%'], 'FontSize', 14);
```

On obtient la figure suivante :



Remarque : cette matrice de confusion porte uniquement sur les pixels de la base d'apprentissage.

Si on veut tester notre carte sur l'ensemble de l'image, on peut utiliser les instructions suivantes pour construire la base de test avant de calculer la matrice de confusion :

```
load L2Flags_LAND.mat; load Labels_Aer_Map.mat;
load L1Brad_rho_412; load L1Brad_rho_443; load L1Brad_rho_490; load L1Brad_rho_510;
load L1Brad_rho_555; load L1Brad_rho_670; load L1Brad_rho_765; load L1Brad_rho_865;
% Base de test
pixels_test = find(LAND == 0 & ...
    isnan(rho_412)==0 & isnan(rho_443)==0 & ...
    isnan(rho_490)==0 & isnan(rho_510)==0 & ...
    isnan(rho_555)==0 & isnan(rho_670)==0 & ...
    isnan(rho_765)==0 & isnan(rho_865)==0 ...
);
AerMap_test = Aer_Map(pixels_test);
Dtest=[rho_412(pixels_test),rho_443(pixels_test),rho_490(pixels_test), ...
    rho_510(pixels_test),rho_555(pixels_test),rho_670(pixels_test), ...
    rho_765(pixels_test),rho_865(pixels_test)];
NomLabels = {'nuage', 'marit', 'cont', 'desert'};
labs = cell(length(pixels_test),1);
for i_flag = 1:4
    j_flag = find(AerMapApp == i_flag);
    if ~isempty(j_flag);
        labsNumerique(j_flag)=i_flag;
        for jj = j_flag'
            labs{jj} = NomLabels{i_flag};
        end
    end
end
end
Neuronedechaquepixel= som_bmus(sM,Dtest);
Classifparcarte=sM.labels(Neuronedechaquepixel,2);
Classifparexpert=transpose(NomLabels(AerMap_test));
correct= find(strcmp(Classifparcarte,Classifparexpert)==1);
pourcentage=length(correct)*100/length(AerMap_test);
nbreclasses=length(NomLabels);
confuse=zeros(nbreclasses,nbreclasses);
for i=1:nbreclasses
    for j=1:nbreclasses
        confuse(i,j)=length(find(strcmp(Classifparexpert,NomLabels(i))==1 ...
            & strcmp(Classifparcarte,NomLabels(j))==1));
    end
end
end
disp(' ')
disp(['pourcentage de bonne classification : ' num2str(pourcentage) '%'])
disp('matrice de confusion')
disp(num2str(confuse))
figure
PloterMatrice(confuse)
title(['Pourcentage de classification: ' num2str(pourcentage) '%'], 'FontSize', 14);
```

On obtient la figure suivante :

Pourcentage de classification: 89.2851 %

2110	4	15	64
3	23062	1062	1328
2	1925	5582	135
61	542	46	12468

Remarque : dans le programme précédent (construction de la base de test) les instructions suivantes (lignes 6 à 9 dans le programme) servent à enlever les pixels qui ont une valeur **NaN** (Not a Number) sur l'une des 8 longueurs d'onde (on peut choisir de ne pas enlever ces pixels).

```

& ...
isnan(rho_412)==0 & isnan(rho_443)==0 & ...
isnan(rho_490)==0 & isnan(rho_510)==0 & ...
isnan(rho_555)==0 & isnan(rho_670)==0 & ...
isnan(rho_765)==0 & isnan(rho_865)==0 ...

```

Visualisation des pixels captés par un neurone (ou un groupe de neurones)

Pour visualiser les pixels captés par un neurone (par exemple le neurone numéro 1), on peut utiliser les instructions suivantes :

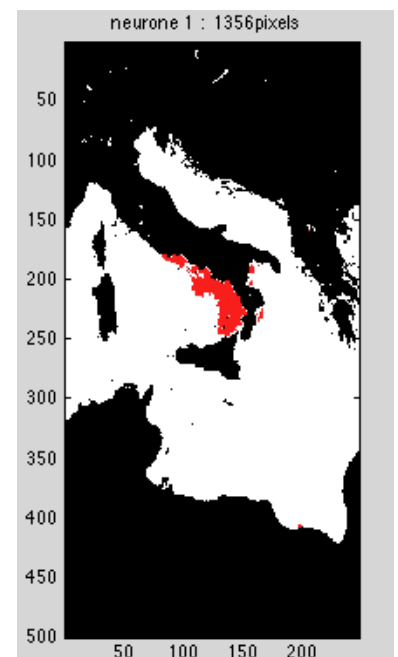
```

num_neurone=1;
[Bmus, Qerrors] = som_bmus(sM, Dtest);
pixels_captés_par_num_neurone=find(Bmus==num_neurone);
nbre_pixels_captés_par_num_neurone=length(pixels_captés_par_num_neurone);
% on visualise sur le masque terre
carte=LAND;
carte(pixels_test(pixels_captés_par_num_neurone))=0.5;
Palette = [ 1 1 1; 1 0 0; 0 0 0];
figure
imagesc(carte,[0 1])
colormap(Palette)
axis image
colorbar
title([ 'neurone ' num2str(num_neurone) ' : ' ...
        num2str(nbre_pixels_captés_par_num_neurone) 'pixels'])

```

On obtient la figure suivante :

- on a visualisé les résultats sur le masque terre
- les **pixels sur terre** sont montrés **en noir**
- les **pixels affectés au neurone d'indice 1** sont montrés **en rouge**



Visualisation des pixels mal classés

Pour visualiser les pixels classés par la carte de façon différente que l'expert, on peut utiliser les instructions suivantes :

```
pixelsmalclasses=find(strcmp(Classifparcarte,Classifparexpert)~=1);  
% on visualise sur la carte des labels  
carte=Aer_Map;  
carte(pixels_test(pixelsmalclasses))=5;  
MesCouleurs = [1 1 1 ; 0 0 1; 0 1 0; 1 1 0; 1 0 0];  
figure  
imagesc(carte)  
colormap(MesCouleurs)  
axis image  
title([num2str(length(pixelsmalclasses)) ' pixels mal classés : en rouge'])
```

On obtient la figure ci-dessous, on a visualisé les résultats sur la carte des labels.

→ **les pixels mal classés** sont montrés **en rouge**

→ **Maintenant vous pouvez faire d'autres apprentissages (faire varier les paramètres d'apprentissage) pour réduire leur nombre**

msize = ? lattice = ?

Phase 1 (auto-organisation) : epochs = ? radius_ini = ? radius_fin = ? Neigh = ?

Phase 2 (convergence) : epochs = ? radius_ini = ? radius_fin = ? Neigh = ?

