Table of Contents

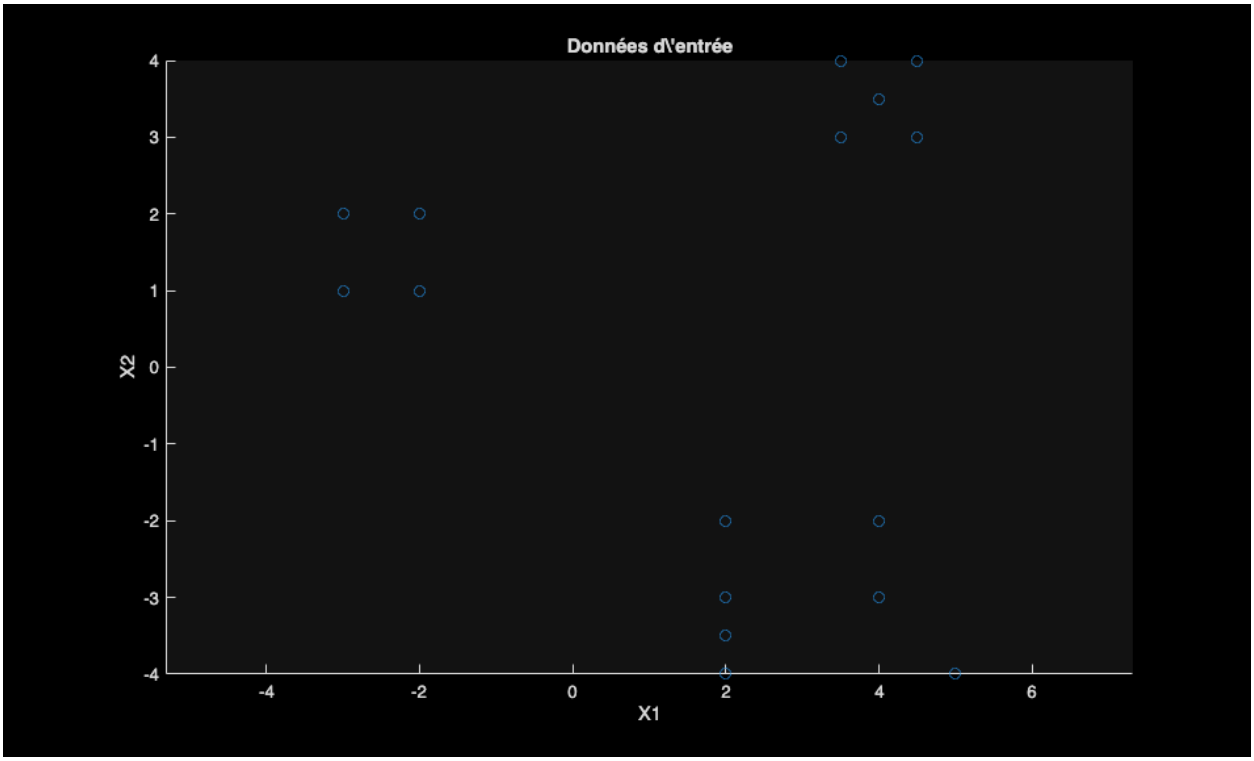
Lab 3 SOM - Réseau SOM de Kohonen 1
Carte Linéaire 1
Visualisation de l'évolution des poids 2
Visualisation des prototypes 3
Conclusion (Question 4) 5
La réduction de dimension 5
Position des individus sur la carte SOM 6
Conclusion sur la réduction de dimension 6
LA PRÉSERVATION TOPOLOGIQUE 7
Calcul des distances entre prototypes 8
Le principe de voisinage 10
Manip 1 10
Affichage du neurone gagnant 12
Q4 13
Le rayon d'apprentissage 13
Autre manip : radius apprentissage nul 16
La fonction d'apprentissage 18
Visualisation de l'alpha 19
Reconnaissance de formes avec le SOM 20
3. SOM avec carte 3x3 avec d1 à d9 21
SOM avec carte 5x5 avec d1 à d9 22
SOM avec carte 10x10 avec d1 à d9 23

Lab 3 SOM - Réseau SOM de Kohonen

```
clear; close all; clc;
```

Carte Linéaire

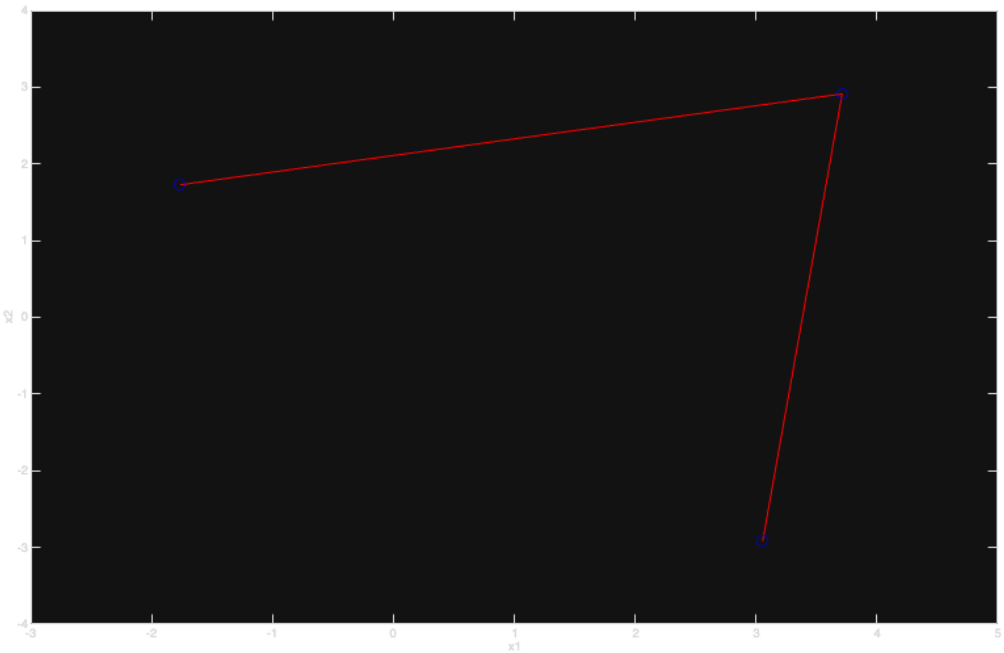
```
% Manipulation des données
S = load('samples1.mat');   % charge la structure
data = S.data;              % extrait la matrice 16×2


% Visualisation des données
figure;
scatter(data(:,1), data(:,2))
title('Données d\''entrée')
xlabel('X1')
ylabel('X2')
axis equal

% Taille du réseau
mapsize = [1 3];
alpha=geometric(1,.9,100); % taux d'apprentissage
```



Visualisation de l'évolution des poids

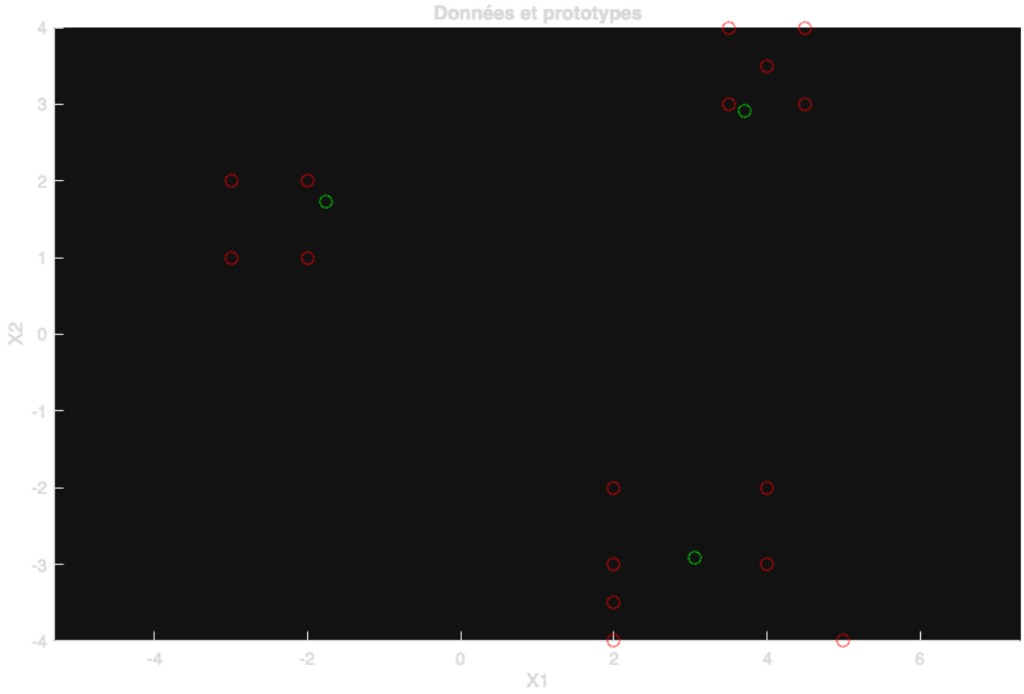
```
[c,p]=som(data,mapsize,'alpha',alpha,'protomap',1);
```

```
Nombre d'epoques : 100
```



Visualisation des prototypes

```
n = size(data,1);    % 16
m = size(p,1);       % 3 prototypes

% Couleur des données (rouge)
dataColor = repmat([1 0 0], n, 1);

% Couleur des prototypes (vert ou autre)
protoColor = repmat([0 1 0], m, 1);

% Matrice de couleurs complète
co = [dataColor ; protoColor];

scatter([data(:,1);p(:,1)], [data(:,2);p(:,2)], 50, co);
title('Données et prototypes')
xlabel('X1')
ylabel('X2')
axis equal
```




```
for classe = 1:3
    fprintf("Classe %d :\n", classe);

    % Trouver les individus appartenant à cette classe
    index = find(c == classe);
    fprintf("Indices des individus : ");
    disp(index');

    % Calcul du centre de gravité des individus de la classe
    centroid = mean(data(index, :), 1);
    fprintf("Centre de gravité : [%f, %f]\n", centroid(1), centroid(2));

    % Prototype associé
    prototype = p(classe, :);
    fprintf("Prototype du SOM : [%f, %f]\n\n", prototype(1), prototype(2));
end
```

```
Classe 1 :
Indices des individus :      1     2     3     4

Centre de gravité : [-2.500000, 1.500000]
Prototype du SOM : [-1.763249, 1.728503]

Classe 2 :
Indices des individus :      5     6     7     8     9

Centre de gravité : [4.000000, 3.500000]
Prototype du SOM : [3.711978, 2.913045]
```

```
Classe 3 :
Indices des individus :     10    11    12    13    14    15    16

Centre de gravité : [3.000000, -3.071429]
Prototype du SOM : [3.054778, -2.916299]
```

Conclusion (Question 4)

En comparant les centres de gravité des classes avec les prototypes du SOM, on remarque qu'ils sont assez proches pour chaque classe. Cela montre que le réseau SOM a bien appris à représenter les groupes de données : chaque neurone correspond à un ensemble cohérent de points.

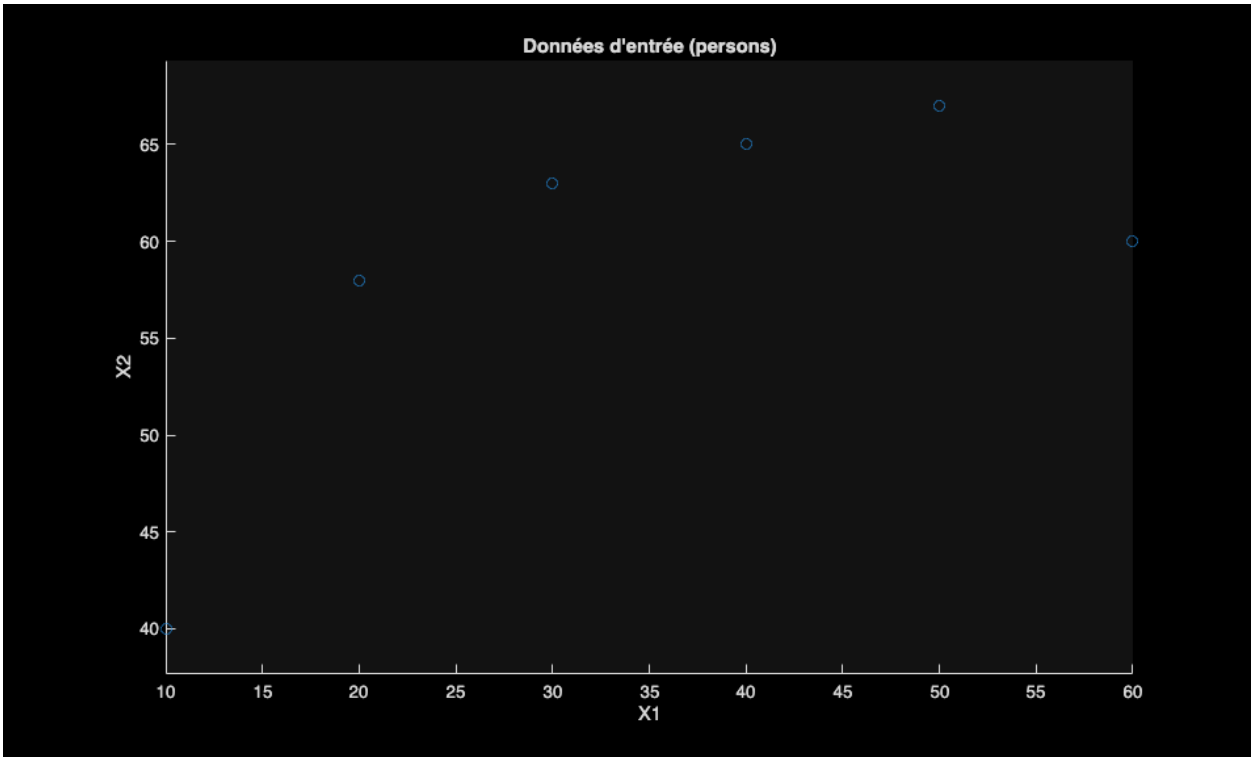
Les prototypes ne coïncident pas exactement avec les centroïdes, ce qui est normal. Le SOM cherche aussi à préserver la topologie, c'est-à-dire à garder un certain ordre entre les neurones sur la carte linéaire. À cause de cette contrainte, les prototypes peuvent être légèrement déplacés par rapport aux centres de gravité exacts.

En résumé : - Les prototypes représentent bien les classes apprises. - Les petites différences avec les centroïdes viennent de la préservation de l'ordre/topologie imposée par la carte SOM.

La réduction de dimension

```
% Charger les données
S = load("persons.mat");
persons = S.persons;  % Matrice 6x3 du TP

% Visualisation simple (projeter les 2 premières dimensions)
figure;
scatter(persons(:,1), persons(:,2))
title('Données d''entrée (persons)')
xlabel('X1')
ylabel('X2')
axis equal

alpha = geometric(1, .9, 100); % taux d'apprentissage

% Taille du réseau
mapsize = [1 8];
[c, p, pMap] = som(persons, mapsize, 'alpha', alpha);
```

```
Nombre d'epoques : 100
```



Position des individus sur la carte SOM

```
disp("Positions des individus sur la carte 1D :");

for i = 1:size(persons,1)
    classe = c(i);          % classe de l'individu i
    pos = pMap(classe, :);  % position (x,y) du prototype correspondant

    fprintf("Individu %d -> classe %d -> position [%d, %d]\n", ...
            i, classe, pos(1), pos(2));
end
```

```
Positions des individus sur la carte 1D :
Individu 1 -> classe 1 -> position [1, 5]
Individu 2 -> classe 2 -> position [1, 3]
Individu 3 -> classe 3 -> position [1, 2]
Individu 4 -> classe 4 -> position [1, 1]
Individu 5 -> classe 5 -> position [1, 7]
Individu 6 -> classe 6 -> position [1, 8]
```

Conclusion sur la réduction de dimension

La carte SOM 1D projette les individus dans un ordre cohérent avec leurs caractéristiques (âge, poids, occupation). Les individus jeunes apparaissent à gauche de la carte, tandis que les individus plus âgés sont projetés vers la droite.

Les individus 3 et 4 sont regroupés dans la même classe, ce qui est logique puisqu'ils ont des valeurs très proches. De même, les individus 5 et 6 apparaissent dans des classes voisines, représentant les personnes plus âgées.

Le SOM parvient donc à réduire la dimension tout en préservant les relations de voisinage entre les individus.

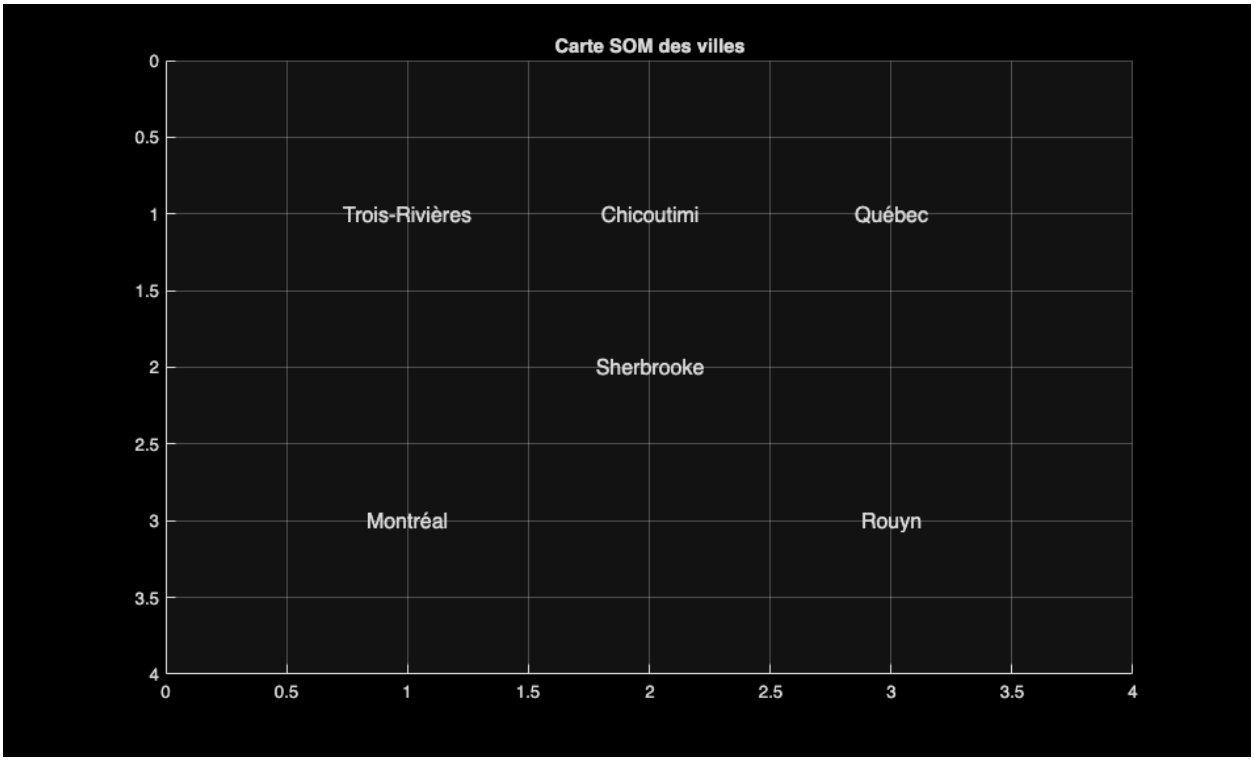
LA PRÉSERVATION TOPOLOGIQUE

```
% Charger les données
S = load('qdist.mat');
qdist = S.qdist;

villes = ["Montréal", "Québec", "Trois-Rivières","Chicoutimi", "Sherbrooke",
"Rouyn"];


mapsize = [3 3];
alpha = geometric(1, .99, 50); % taux d'apprentissage
[c, p, pMap] = som(qdist, mapsize, 'alpha', alpha);

fprintf("\nPositions des villes sur la carte 3x3 :\n");

for i = 1:6
    classe = c(i);
    pos = pMap(classe, :);   % coordonnée (x,y)
    fprintf("%s -> classe %d -> position (%d, %d)\n", villes(i), classe,
pos(1), pos(2));
end

figure; hold on;
for i = 1:6
    classe = c(i);
    pos = pMap(classe,:);
    text(pos(2), pos(1), villes(i), ...
        'HorizontalAlignment','center',...
        'VerticalAlignment','middle',...
        'FontSize',12)
end

xlim([0 4]); ylim([0 4]);
set(gca,'YDir','reverse')
grid on;
title("Carte SOM des villes")

% Le SOM 3x3 organise les villes selon leurs proximités dans la matrice
qdist.
% On observe que la carte de Kohonen reproduit bien les relations
géographiques :
%
% - Québec, Chicoutimi et Sherbrooke apparaissent alignées sur la première
rangée,
%   ce qui reflète leur proximité relative dans l'est et le sud du Québec.
%
% - Trois-Rivières et Montréal se retrouvent dans la même colonne, ce qui est
%   cohérent avec leur position réelle le long du fleuve Saint-Laurent.
%
% - Rouyn est isolée dans un coin de la carte, ce qui correspond à son
```

```
éloignement
%   géographique des autres villes.
%
% Bien que la carte SOM inverse certaines directions (haut/bas, gauche/
droite),
% elle réussit à préserver la topologie générale et les relations de
voisinage.
% Le réseau fournit donc une représentation fidèle et utile de la géographie
% du Québec à partir uniquement des distances inter-villes.
```

Nombre d'epoques : 50

Positions des villes sur la carte 3x3 :
Montréal -> classe 1 -> position (3, 1)
Québec -> classe 2 -> position (1, 3)
Trois-Rivières -> classe 3 -> position (1, 1)
Chicoutimi -> classe 4 -> position (1, 2)
Sherbrooke -> classe 5 -> position (2, 2)
Rouyn -> classe 6 -> position (3, 3)

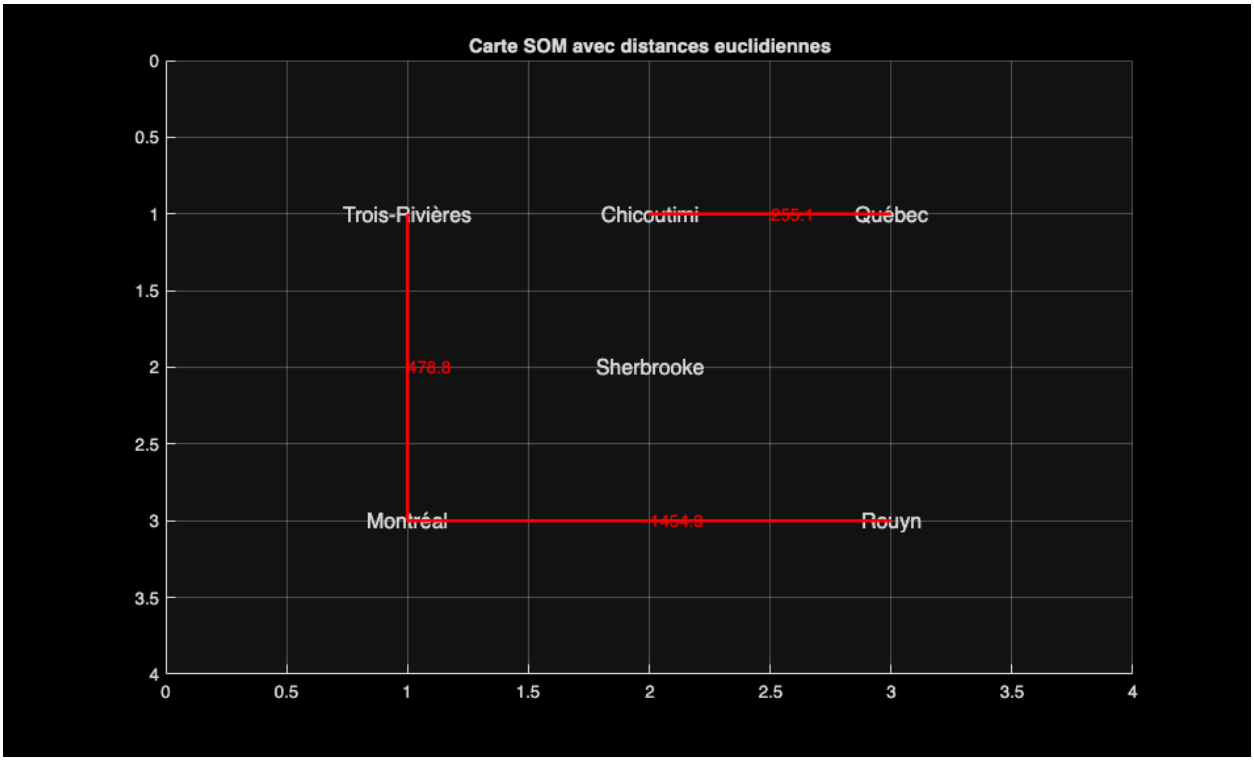


Calcul des distances entre prototypes

```
D = squareform(pdist(qdist, "euclidean"));

disp("Distances euclidiennes entre les villes :");
disp(D);
```

```
Distances euclidiennes entre les villes :
   1.0e+03 *

         0    0.7835    0.4788    0.7445    0.6868    1.4543
    0.7835         0    0.5239    0.2551    0.2817    1.4483
    0.4788    0.5239         0    0.4452    0.3177    1.6568
    0.7445    0.2551    0.4452         0    0.2634    1.5387
    0.6868    0.2817    0.3177    0.2634         0    1.5899
    1.4543    1.4483    1.6568    1.5387    1.5899         0
```

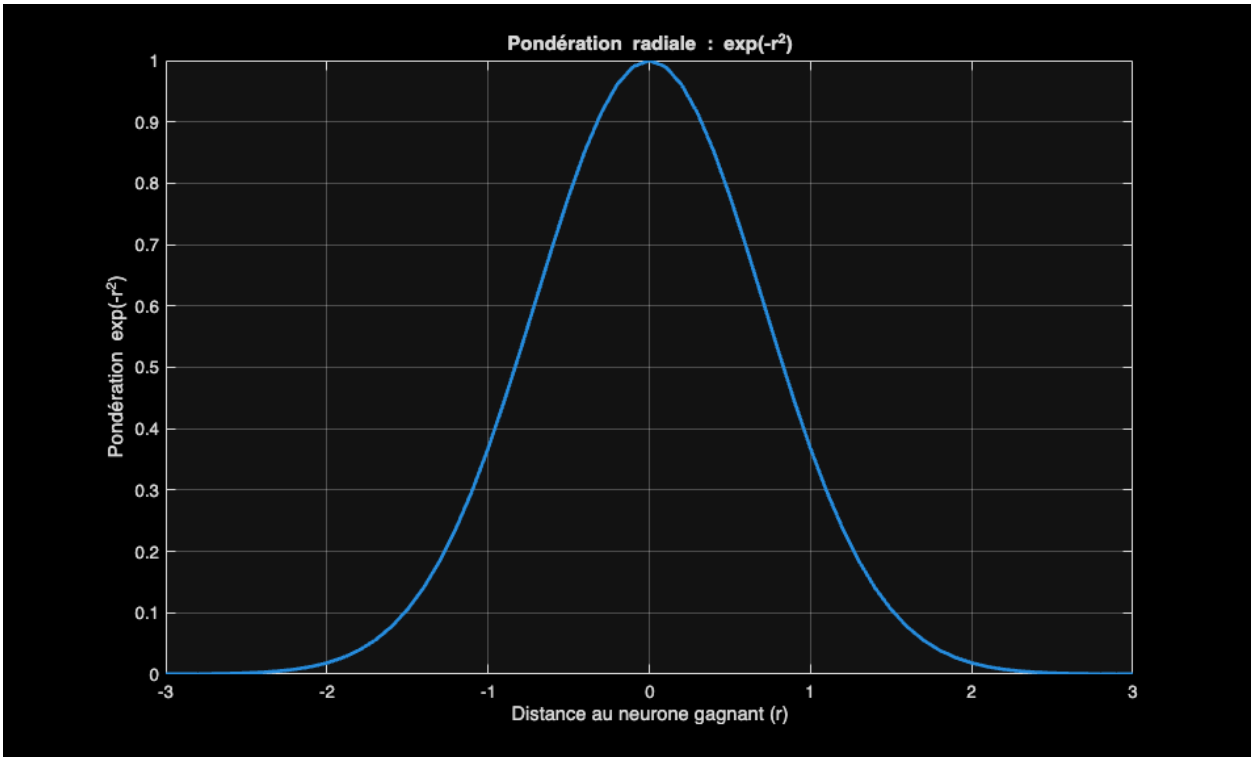
```
villes = ["Montréal","Québec","Trois-
Rivières","Chicoutimi","Sherbrooke","Rouyn"];

% Paires à afficher
pairs = [1 3;   % Mtl — TR
         2 4;   % Qc — Chicoutimi
         1 6];  % Mtl — Rouyn

figure; hold on;
grid on;
axis([0 4 0 4]);
set(gca,'YDir','reverse');
title("Carte SOM avec distances euclidiennes");

for i = 1:6
    pos = pMap(c(i),:);
    text(pos(2), pos(1), villes(i), ...
        "HorizontalAlignment","center","FontSize",12);
end

for k = 1:size(pairs,1)
    i = pairs(k,1);
    j = pairs(k,2);
    pos1 = pMap(c(i),:);
    pos2 = pMap(c(j),:);

    mid = (pos1 + pos2)/2;

    plot([pos1(2) pos2(2)], [pos1(1) pos2(1)], "r-", "LineWidth",2);
    text(mid(2), mid(1), sprintf("%.1f", D(i,j)), "Color","red");
end

% Conclusion :
% La carte SOM organise les villes en fonction de la similarité entre leurs
% profils de distances, et non selon la géographie réelle. Les petites
% distances euclidiennes (ex. Québec—Chicoutimi, Québec—Sherbrooke) se
% retrouvent sur des neurones proches du SOM. Les grandes distances
% euclidiennes (ex. Rouyn avec toutes les autres villes) placent Rouyn
% dans un coin éloigné du SOM.
%
% La carte SOM parvient donc à reproduire des relations de voisinage
% basées sur les similarités dans la matrice qdist.
```



Le principe de voisinage

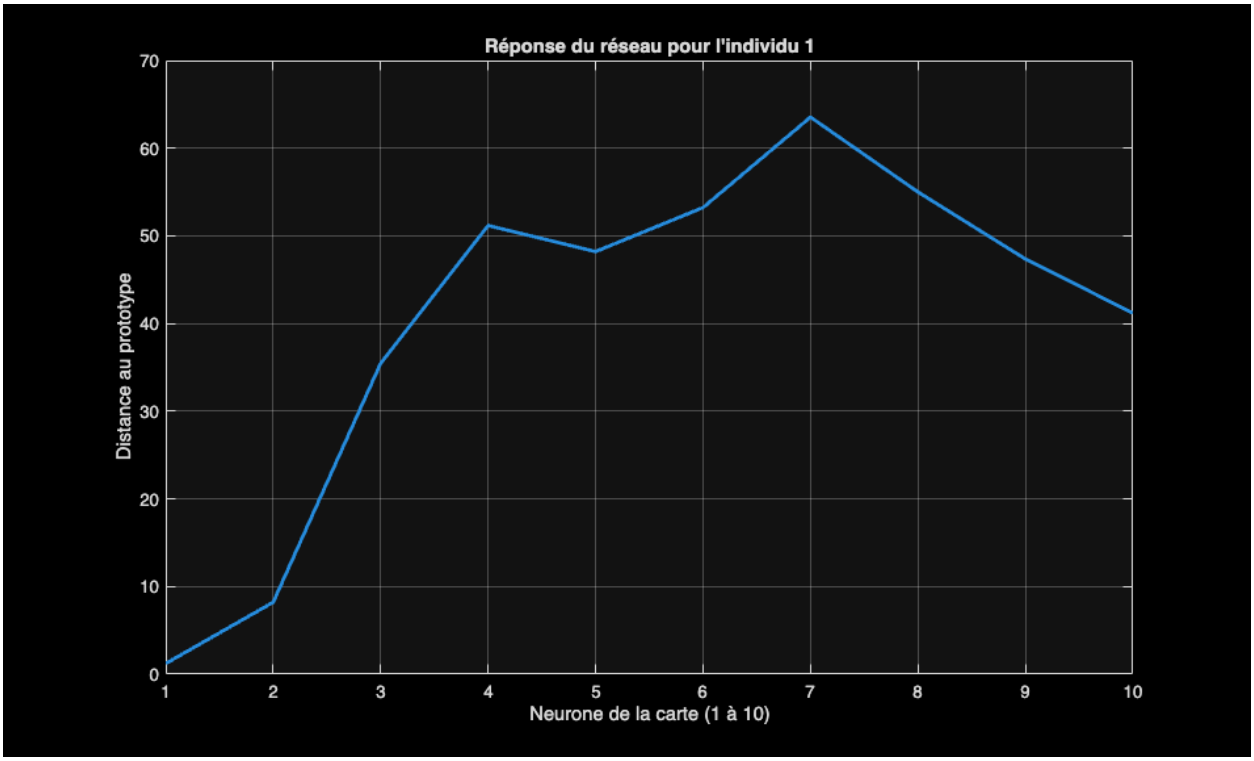
Manip 1

```
figure;
r = -3:0.1:3;
plot(r, exp(-r.^2), 'LineWidth', 2)
title('Pondération radiale : exp(-r^2)')
xlabel('Distance au neurone gagnant (r)')
ylabel('Pondération exp(-r^2)')
grid on
```




```
data = load('samples1.mat') ;
data = data.data ;

alpha = geometric (1, .9, 500) ;
mapsize = [1 10] ;
[c, p, pMap, rMap] = som(data, mapsize, 'alpha' , alpha) ;

figure;
plot(1:10, rMap{1}, 'LineWidth', 2)
title('Réponse du réseau pour l''individu 1')
xlabel('Neurone de la carte (1 à 10)')
ylabel('Distance au prototype')
grid on
```

Etat stable atteint
Nombre d'epoques : 363



Affichage du neurone gagnant

```
[~, gagnant] = min(rMap{1})
fprintf('Le neurone gagnant pour l''individu 1 est le neurone %d\n',
gagnant);

classe = c(1)
pMap(classe,:)
fprintf('Le prototype associé au neurone gagnant est [%f, %f]\n',
pMap(classe,1), pMap(classe,2));

% Le neurone gagnant est celui où la réponse (distance au prototype) est
% la plus petite. Ici, le minimum est au neurone 1.

% La courbe semble inversée parce que rMap contient des DISTANCES :
% - un neurone proche du vecteur d'entrée □ petite distance □ valeur basse
% - donc le neurone gagnant apparaît comme un creux, pas un pic.
```

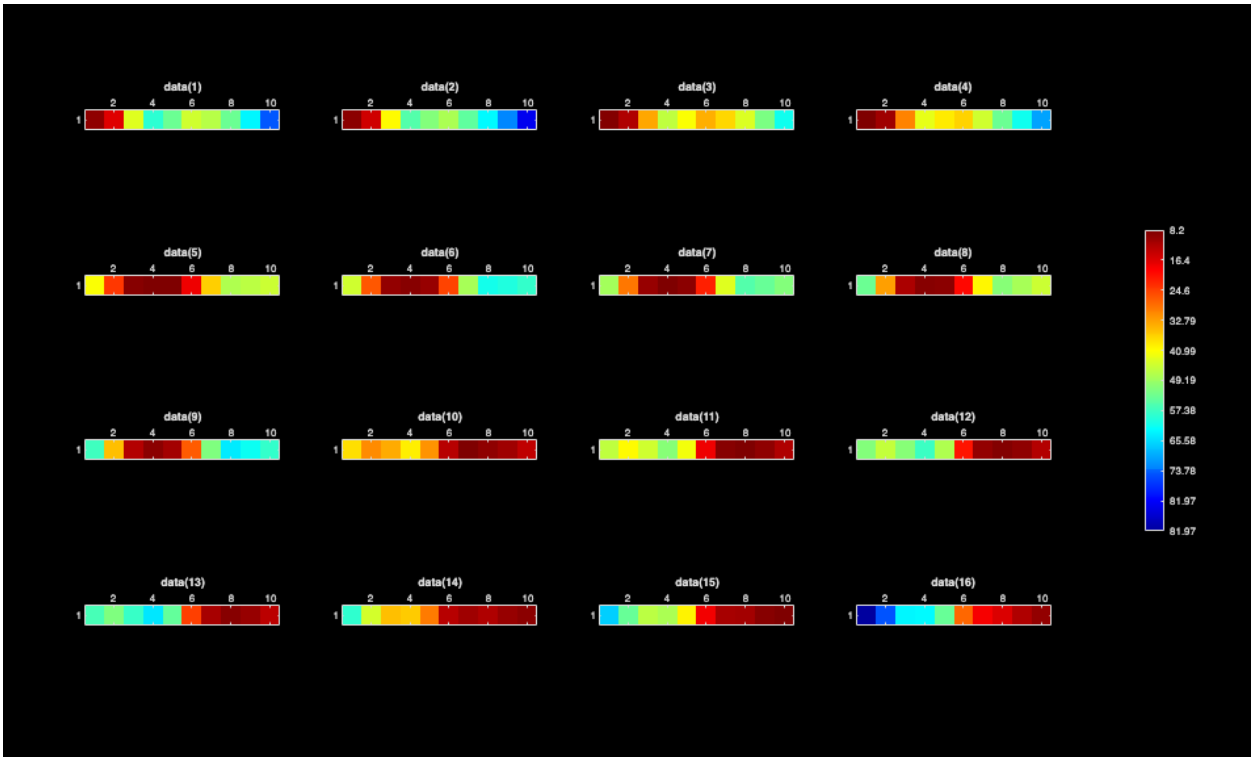
```
gagnant =

     1

Le neurone gagnant pour l'individu 1 est le neurone 1

classe =

     1
```

```
ans =

     1     1

Le prototype associé au neurone gagnant est [1.000000, 1.000000]
```

Q4

```
[c, p, pMap, rMap] = som(data, mapsize, 'alpha' , alpha, 'datamap', 1) ;

% La fenêtre "datamap" montre la réponse du SOM pour chaque individu.
% Chaque petite figure correspond à un individu, et les couleurs indiquent
% la distance entre l'individu et chaque neurone de la carte.
% - Bleu = petite distance donc c'est le neurone gagnant
% - Rouge/jaune = grande distance donc c'est un neurone moins adapté
% Cela permet de voir directement quel neurone gagne pour chaque vecteur,
% ce qui confirme les classes dans c et les positions dans pMap.
```

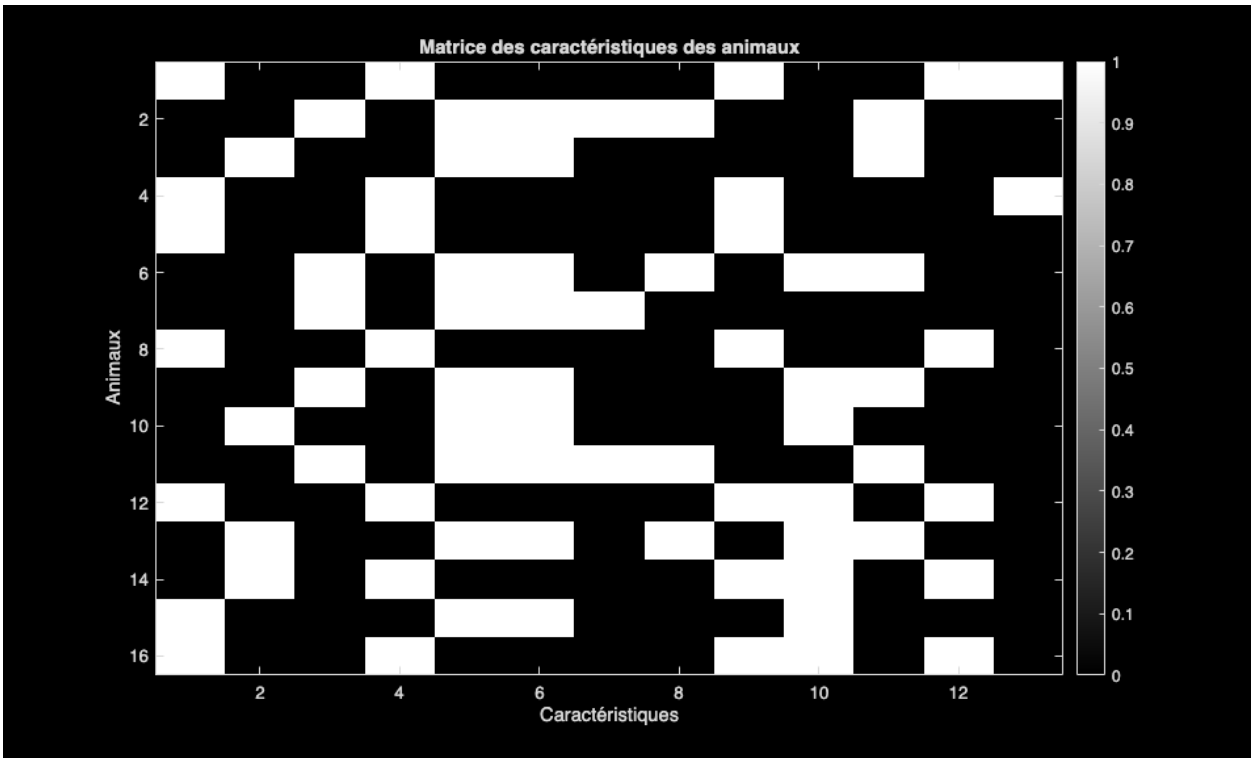
```
Etat stable atteint
Nombre d'epoques : 380
```



Le rayon d'apprentissage

```
% Chargement des données animals.mat
S = load('animals.mat');
animals = S.animals;
```

```
% Visualisation de la matrice animals
figure;
imagesc(animals);
colormap(gray);
colorbar;
title('Matrice des caractéristiques des animaux');
xlabel('Caractéristiques');
ylabel('Animaux');
```

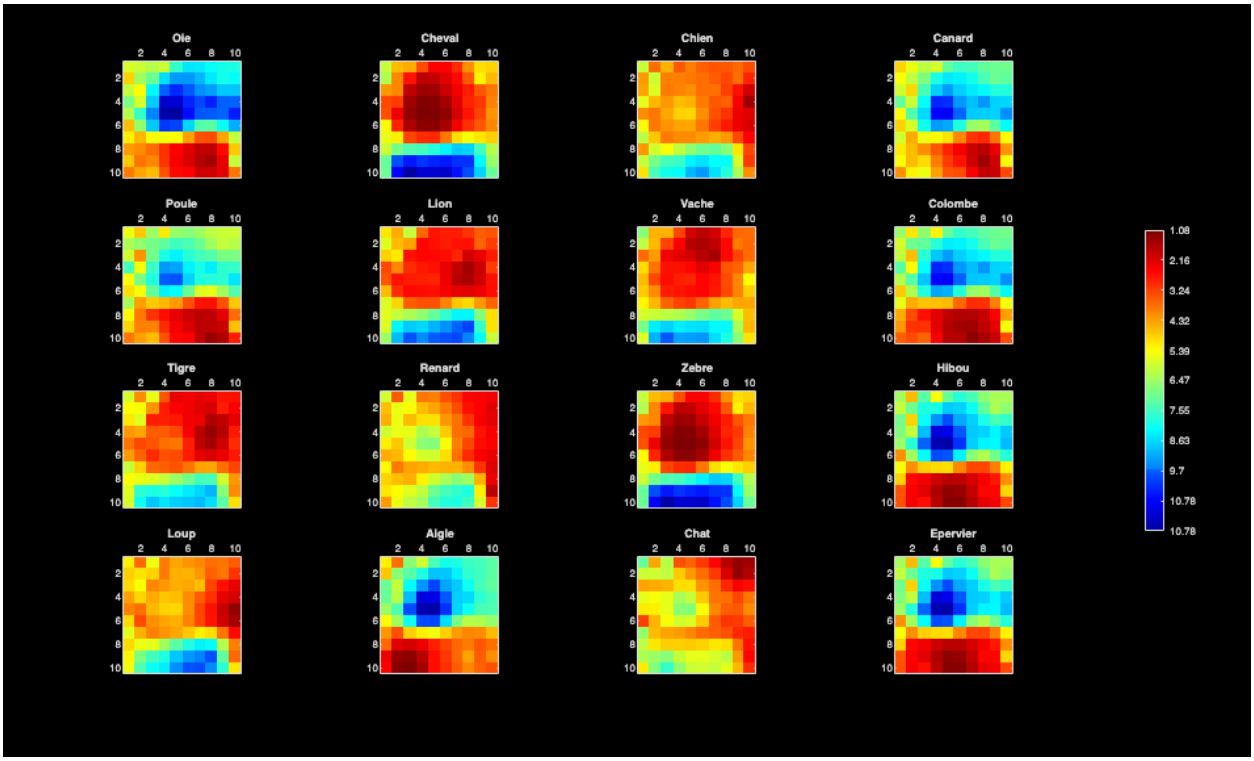


Noms des animaux pour les étiquettes oie cheval chien canard poule lion vache colombe tigre renard zebre hibou loup aigle chat epervier

```
names = {'Oie', 'Cheval', 'Chien', 'Canard', 'Poule', 'Lion', 'Vache',
'Colombe', 'Tigre', 'Renard', 'Zebre', 'Hibou', 'Loup', 'Aigle', 'Chat',
'Epervier'};
mapsize = [10 10];
alpha = geometric(1, .9, 500);
som(animals, mapsize, 'alpha', alpha, 'datamap', 1, 'datatags', names);

% 1) Où sont projetés les oiseaux ?
% Les oiseaux (Oie, Canard, Poule, Colombe, Hibou, Aigle, Épervier)
apparaissent tous regroupés dans la zone en haut à droite et en bas à gauche
de la carte SOM.
% Cette région contient surtout des couleurs froides pour les oiseaux, ce
qui indique qu'ils activent les mêmes neurones du SOM.

% 2) Y a-t-il d'autres animaux projetés dans ces zones ?
% Non, les autres animaux sont projetés dans des zones distinctes de la
carte, loin des oiseaux.
```

```
% 3) Y a-t-il des sous-groupes parmi les oiseaux ?
% Oui, on peut distinguer deux sous-groupes :
% - Les oiseaux de basse-cour (Oie, Canard, Poule, Colombe) sont projetés en
haut à droite,
% - Les oiseaux de proie (Aigle, Hibou, Épervier) apparaissent en bas à
gauche.
% Cela reflète leurs différences biologiques et comportementales.

% 4) Division de la carte en régions principales
% En divisant la carte en 4 grandes zones, on observe une organisation
% logique des animaux :

% - Région 1 (haut droite) : Oiseaux de basse-cour : Oie, Canard, Poule,
Colombe
%
% - Région 2 (bas gauche) : Oiseaux de proie : Aigle, Hibou, Épervier
%
% - Région 3 (haut gauche) : Herbivores à quatre pattes : Cheval, Vache,
Zèbre
%
% - Région 4 (centre et bas centre) : Carnivores : Lion, Tigre, Loup,
Renard, Chien, Chat

% Cette répartition n'est pas aléatoire : le SOM regroupe les animaux
% selon leurs caractéristiques communes (morphologie, régime, comportement).
% On observe clairement des familles cohérentes : oiseaux ensemble,
% carnivores ensemble, herbivores ensemble, etc.
```

Etat stable atteint
Nombre d'epoques : 359



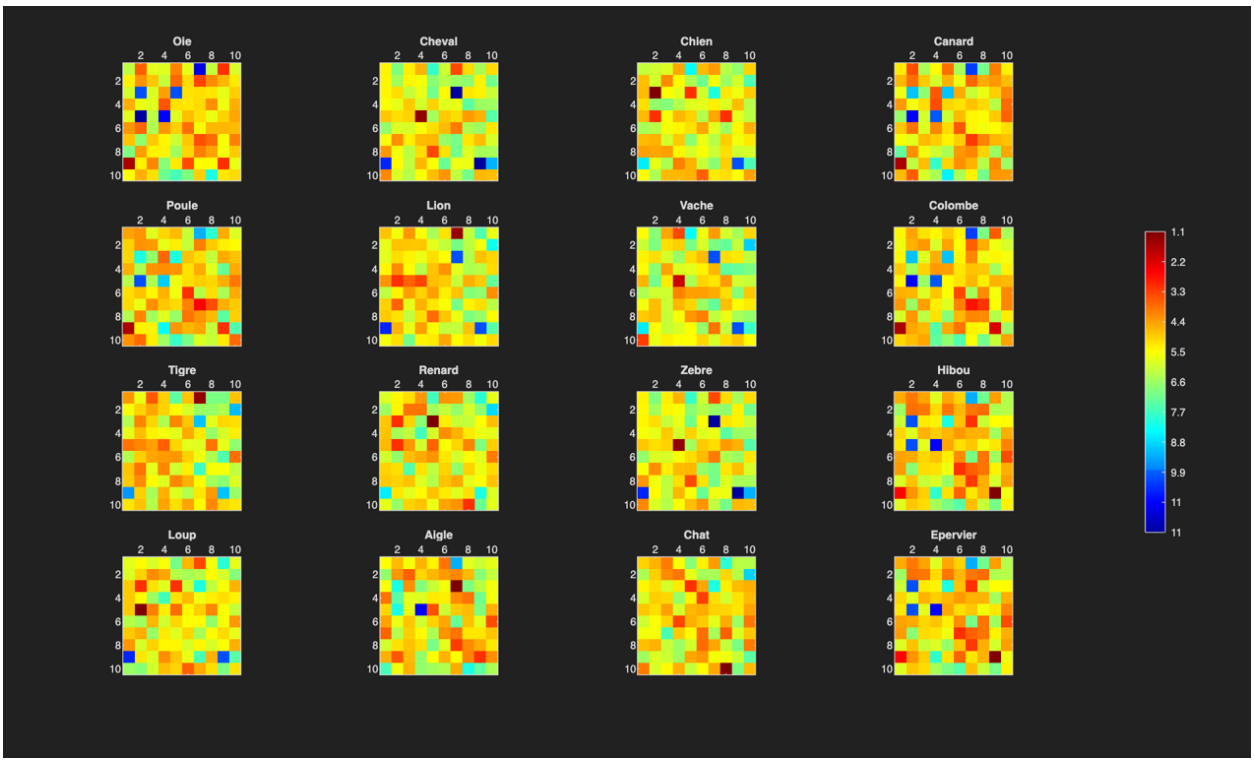
Autre manip : radius apprentissage nul

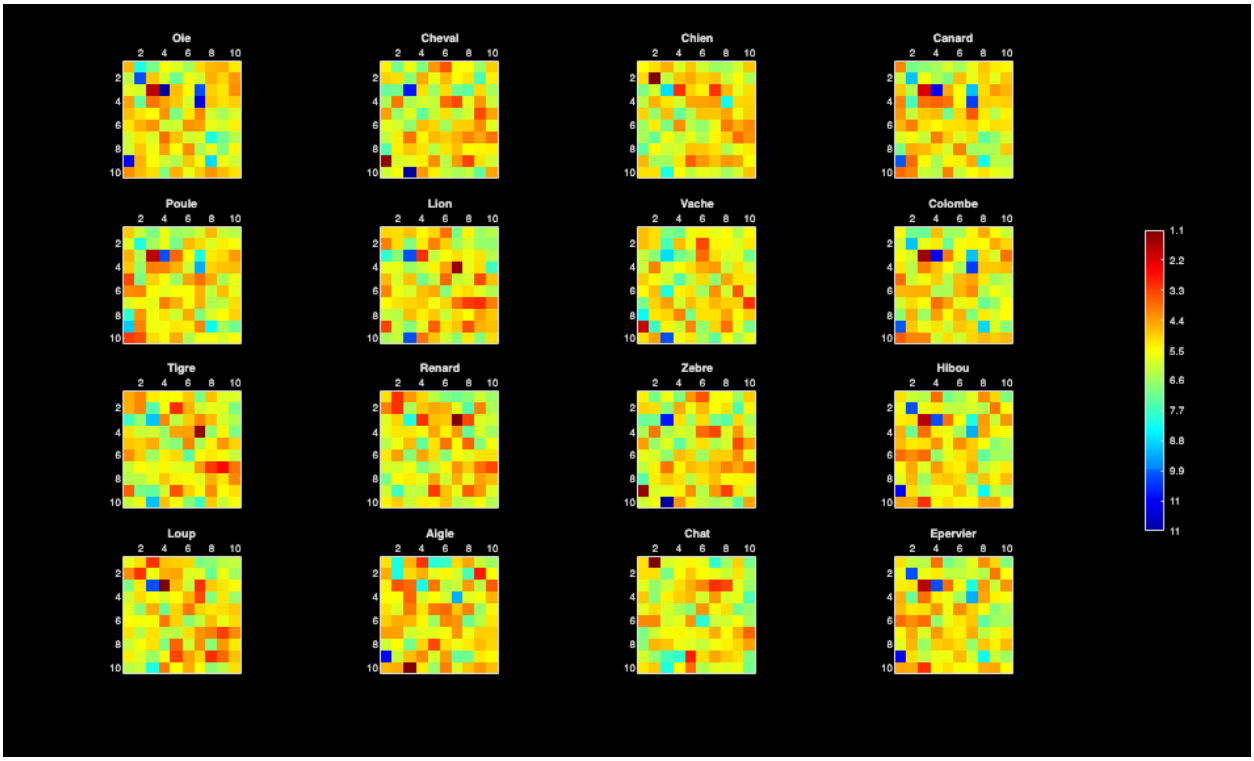
```
clear; clc;

% Charger données animals
load animals.mat   % contient 'animals' et 'names'

% Paramètres
alpha = geometric(1, .9, 500);   % taux d'apprentissage
radius = zeros(1,500);           % rayon nul à chaque époque
mapsize = [10 10];

% SOM avec rayon nul

[c, p, pMap, rMap] = som(animals, mapsize, 'alpha', alpha, 'radius',
radius,'datamap', 1,'datatags', names);

% Analyse des résultats
% Avec un rayon d'apprentissage nul, chaque neurone n'influence que lui-
même, la carte n’apprend plus de structure.
% Les neurones voisins ne sont jamais ajustés, donc les familles d’animaux
ne se regroupent plus clairement et la carte devient désorganisée.

% Nouvelle figure pour comparaison
set(gcf, 'HandleVisibility', 'off');

som(animals, mapsize, 'alpha', alpha, 'radius', radius, 'datamap', 1,
'datatags', names);
```

```
% Avec un rayon nul, la carte SOM perd sa capacité à organiser les données.
% Les animaux ne sont plus projetés aux mêmes endroits, les regroupements
naturels disparaissent et la carte n'est plus informative.
%  Cela montre que le voisinage est essentiel pour obtenir une carte SOM
stable et cohérente.
```

Etat stable atteint
Nombre d'epoques : 357

Etat stable atteint
Nombre d'epoques : 360

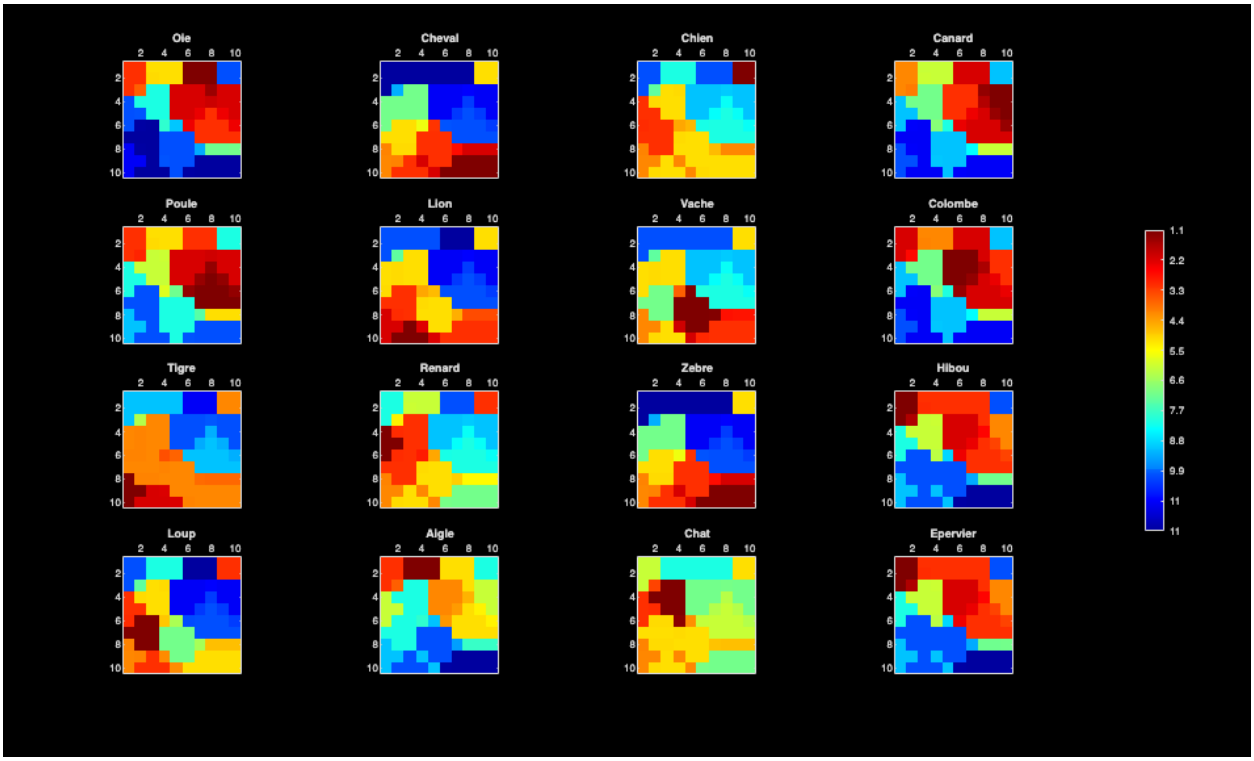




La fonction d'apprentissage

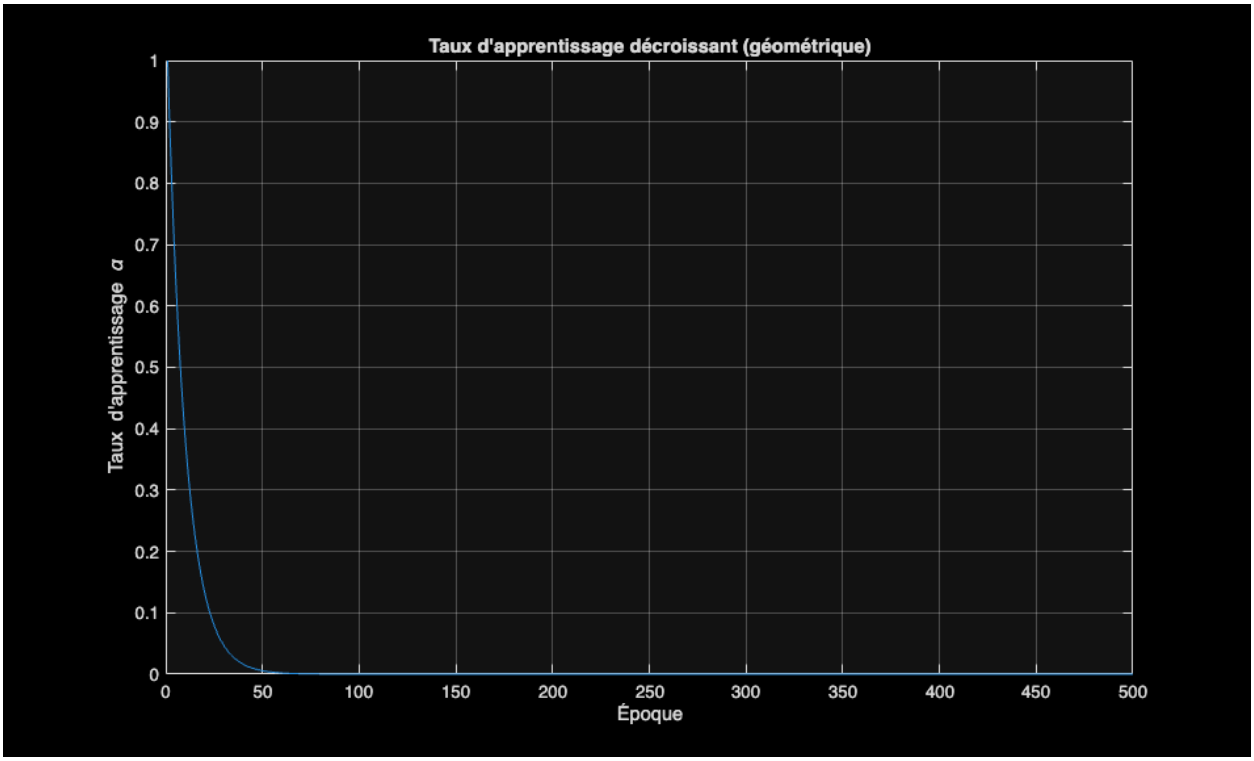
```
% Manipulations
clear; clc;

% Charger données animals
S = load('animals.mat');
animals = S.animals;
names = {'Oie', 'Cheval', 'Chien', 'Canard', 'Poule', 'Lion', 'Vache',
'Colombe', 'Tigre', 'Renard', 'Zebre', 'Hibou', 'Loup', 'Aigle', 'Chat',
'Epervier'};


mapsize = [10 10];
som(animals, mapsize, 'epochs', 500, 'datamap', 1, 'datatags', names);

% Analyse
% La carte SOM obtenue sans alpha est moins lissée et moins organisée que
celle obtenue avec un alpha géométrique.
% Les regroupements d'animaux sont plus flous et moins stables, donc
l'apprentissage linéaire est moins efficace pour structurer la carte.
% Avec l'apprentissage géométrique, la topologie est mieux respectée et les
familles d'animaux sont mieux séparées.
```

Nombre d'epoques : 500



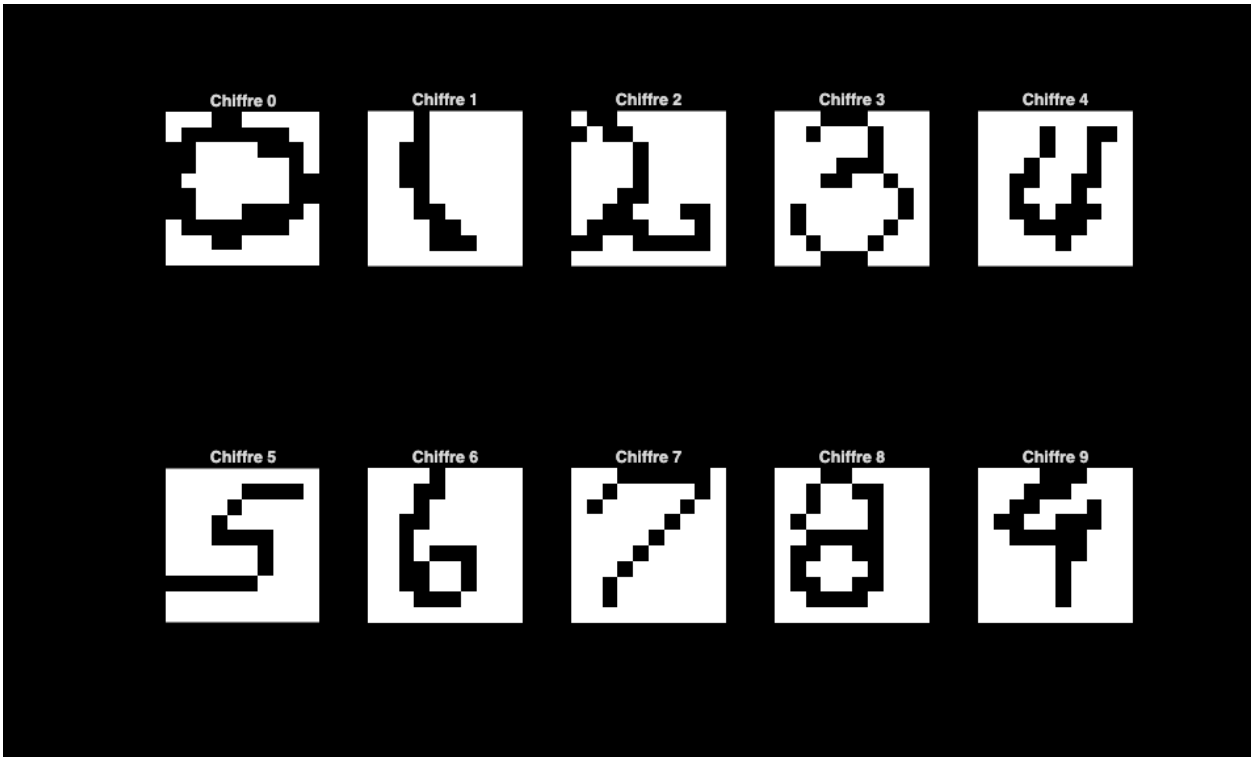
Visualisation de l'alpha

```
figure;
alpha = geometric(1, .9, 500);
plot (1:length(alpha), alpha)
title('Taux d''apprentissage décroissant (géométrique)')
xlabel('Époque')
ylabel('Taux d''apprentissage \alpha')
grid on
```



Reconnaissance de formes avec le SOM

```
clear all; clc;%% 1. Charger les données digits.mat
load digits.mat

% Visualisation des d0 à d9
figure;
for i = 0:9
    subplot(2,5,i+1);
    img = reshape(eval(['d' num2str(i) '(1,:)']), 10, 10);
    imshow(img', []);   % le transpose() pour remettre l'image dans le bon
sens
    title(['Chiffre ' num2str(i)]);
end
dig = [d0 ; d1 ; d2 ; d3 ; d4 ; d5 ; d6 ; d7 ; d8 ; d9];
```



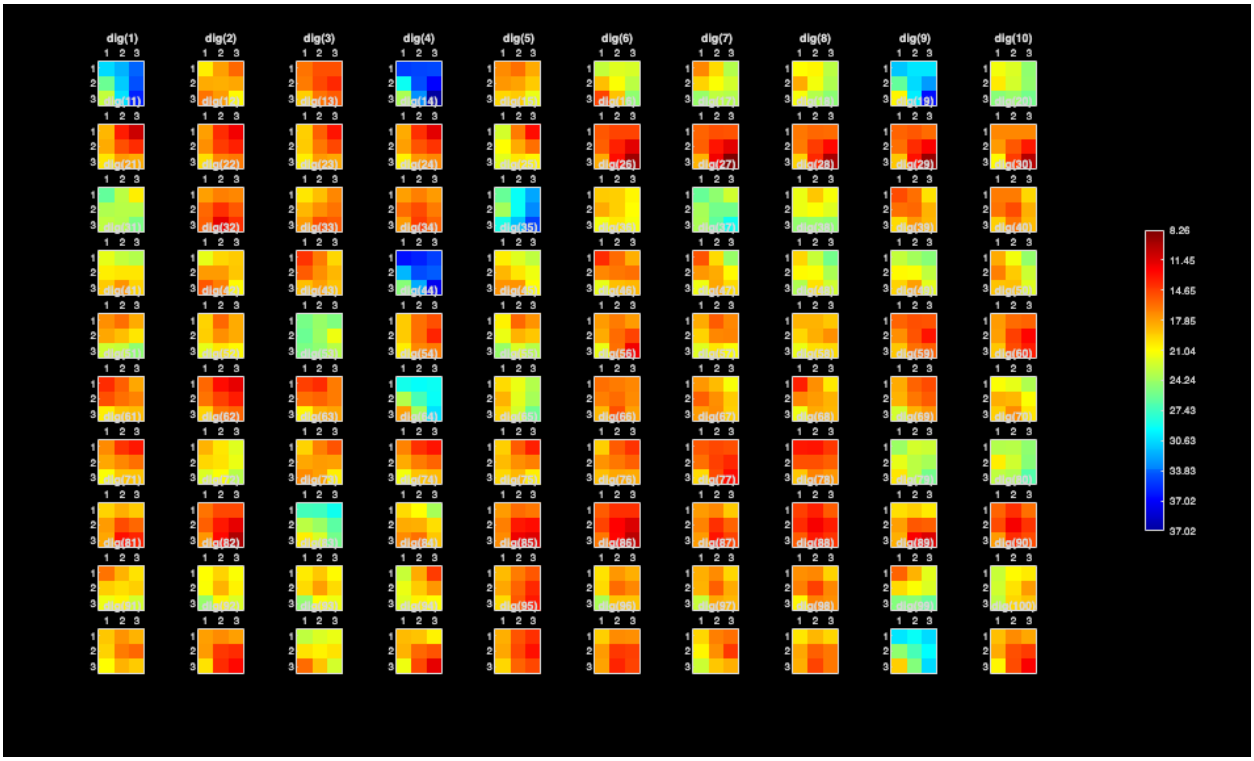
2. Paramètre d'apprentissage

```
alpha = geometric(1, .9, 1000);
```

3. SOM avec carte 3x3 avec d1 à d9

```
[c3, p3] = som(dig, [3 3], 'alpha', alpha, 'datamap', 1);
%La carte 3x3 contient seulement 9 neurones, ce qui est insuffisant pour
représenter les 100 images de chiffres.
% Plusieurs chiffres sont donc projetés sur les mêmes neurones.
% Les distances affichées sur les heatmaps sont très similaires d'un neurone
à l'autre. Aucun clustering clair n'apparaît et la topologie n'est pas
respectée.
% La carte est trop compressée car elle ne parvient pas à séparer les
classes ni à capturer les différences visuelles entre les chiffres.
%
% Ainsi, le SOM 3x3 donne une représentation inutilisable.
```

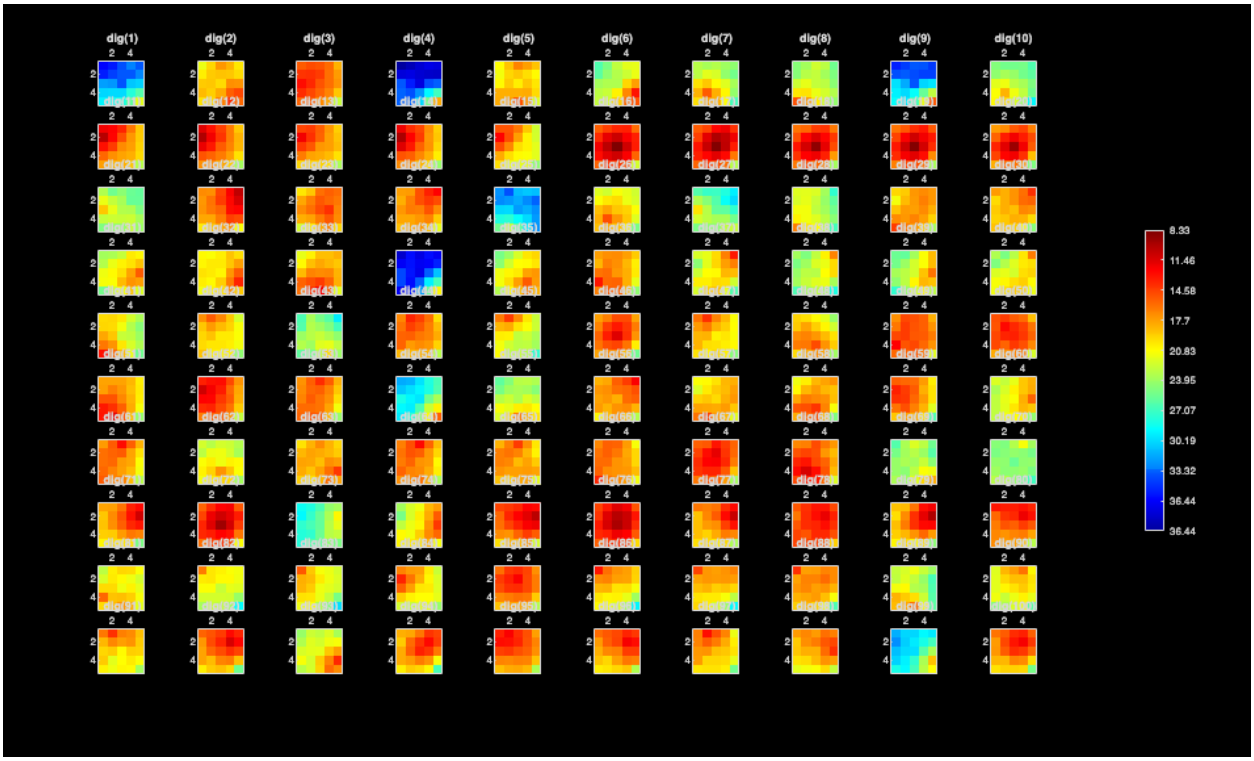
```
Etat stable atteint
Nombre d'epoques : 360
```



SOM avec carte 5x5 avec d1 à d9

```
[c5, p5] = som(dig, [5 5], 'alpha', alpha, 'datamap', 1);

% Avec une carte 5x5, certains chiffres proches visuellement (ex : 1 et 7)
se retrouvent projetés vers des zones similaires.
% Les heatmaps montrent un début de structuration et la topologie est
partiellement respectée.
% Les regroupements sont visibles mais encore imparfaits, et plusieurs
classes se superposent.
%
% Ainsi, le SOM 5x5 commence à organiser les données et affiche une structure
% globale, mais reste insuffisant pour une classification fine.
```

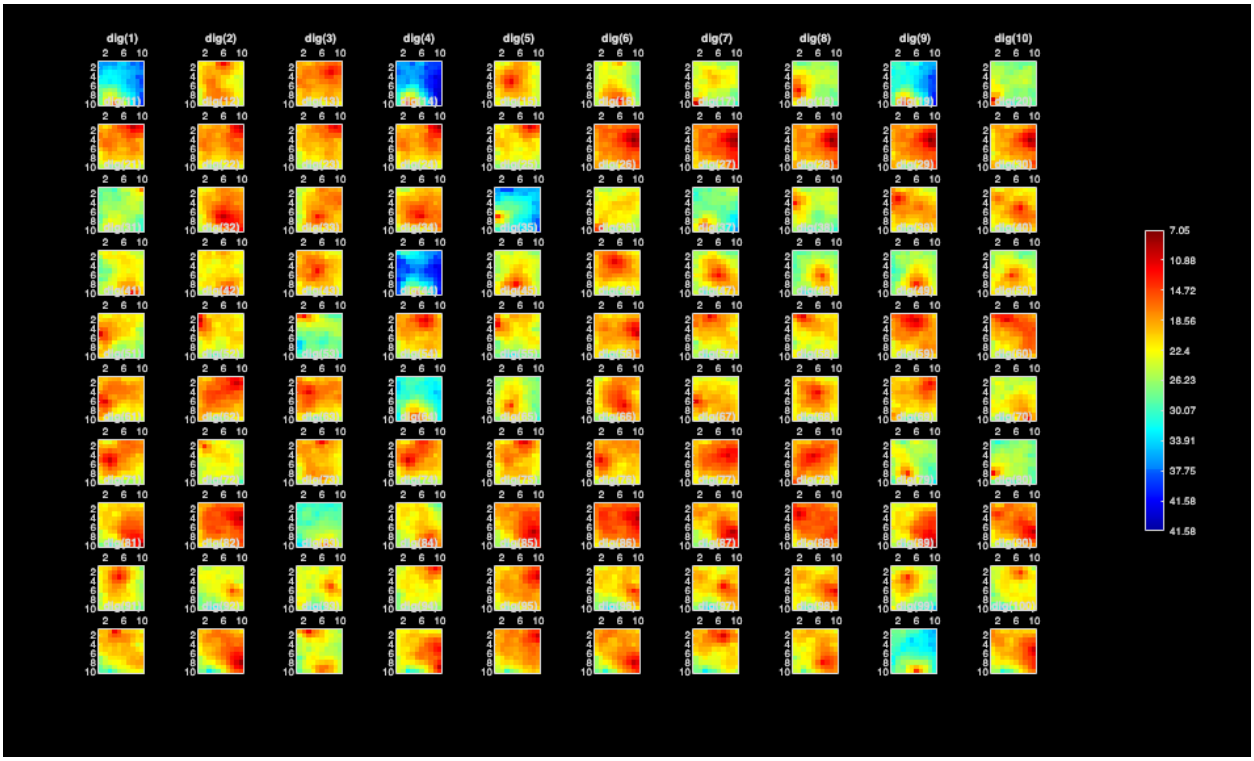
Etat stable atteint
Nombre d'epoques : 368



SOM avec carte 10x10 avec d1 à d9

```
[c10, p10] = som(dig, [10 10], 'alpha', alpha, 'datamap', 1);

% La carte 10x10 dispose d'un neurone pour chaque image, ce qui permet aux
prototypes d'être beaucoup plus précis et représentatifs.
% Les chiffres similaires (ex : 0, 6, 8) apparaissent dans des régions
voisines, tandis que les chiffres différents (ex : 1 vs 9) sont bien séparés.
% Les heatmaps montrent des neurones gagnants bien distribués. Chaque
chiffre active des zones spécifiques, ce qui indique une bonne
% organisation et une classification cohérente.
%
% Ainsi, le SOM 10x10 est le plus performant car il capture la structure des
% données et sépare efficacement les classes.
```

Etat stable atteint
Nombre d'epoques : 368



Published with MATLAB® R2025a