L'objectif de ce TD est de mettre en place un réseau de Kohonen. Tout d'abord vous allez développer un réseau 2-D afin de vous familiariser avec le principe des réseaux de Kohonen.

1 Mise en place d'un réseau de Kohonen 2D

– *Créer une fonction*

```
[w]=kohonen2d(x,K,mu,sigma,nbiter);
```

Ce programme doit effectuer l'apprentissage d'un réseau de Kohonen pour des entrées élément de R^2.
Les paramètres :
- La variable `x` contient l'entrée du réseau de neurones. C'est un tableau à 2 lignes et N colonnes, N étant le nombre d'individus.
- La variable `K` indique le nombre de cellules du réseau (KxK).
- La variable `nbiter` indique le nombre d'itérations de l'apprentissage.
- La variable `mu` de taille `nbiter` contient toutes les valeurs de μ pour les différentes itérations (entre initiale et finale).
- La variable `sigma` de taille `nbiter` contient toutes les valeurs de σ pour les différentes itérations (entre initiale et finale).

Le résultat :
- La variable `w` contient les poids synaptiques du réseau de Kohonen 2D. C'est un tableau de taille KxKx2.

Paramétrages
Pour ce qui concerne les coefficients liés à l'apprentissage, nous proposons d'utiliser la définition classique :

$$w_j^{(t+1)} = w_j^{(t)} + h^{(t)}(q,j)[x^{(t)} - w_j^{(t)}]$$

avec (t) l'indice d'itération et

$$h^{(t)}(q,j) = \mu^{(t)} \exp\left(-\frac{\|position(q) - position(j)\|^2}{2(\sigma^{(t)})^2}\right)$$

Le paramètre q correspond au numéro de noeud gagnant, μ correspond au coefficient d'apprentissage, σ correspond au voisinage et ces deux paramètres doivent évoluer au cours du temps (en général suivant une décroissance en $1/t$).

Pour paramétrer cette évolution au cours du temps, nous proposons là-aussi d'utiliser les valeurs classiques :

```
MUinitial = 0.5; MUfinal = 0.1; SIGMAinitial= 3; SIGMAfinal= 0.1;
iter=0:1:(nbiter-1)
mut = MUinitial + iter/(nbiter-1) * (MUfinal - MUinitial)
sigmat = SIGMAinitial + iter/(nbiter-1) * (SIGMAfinal - SIGMAinitial)
```

En ce qui concerne l'apprentissage, à chaque boucle nous tirons au hasard un élément de l'ensemble de l'apprentissage et nous calculons le neurone gagnant afin de faire la modifications des différents poids. Ceci est répété N fois, avec N nombre d'itérations (par exemple 500).

– *Tester votre réseau de Kohonen avec des données issues d'une distribution uniforme fournies dans le fichier data.mat*
Ce fichier contient deux ensembles de données : xdisc et xunif.

– *Afficher le réseau obtenu après apprentissage en utilisant la fonction affiche_grille. Commenter le résultat.*

2 Utilisation d'un réseau de Kohonen 3D pour la réduction de couleur

Nous voulons utiliser un réseau de Kohonen afin de réduire le nombre de couleurs d'une image codée à l'origine sur 16 millions de couleurs. Pour cela nous devons utiliser un réseau de Kohonen 3D de taille KxKxK correspondant aux nombres de couleurs que l'on conserve après réduction.

2.1 Programmation kohonen3D

– *Créer une fonction*

```
[w]=kohonen3d(x,K,mu,sigma,nbiter);
```

Ce programme doit effectuer l'apprentissage d'un réseau de Kohonen pour des entrées élément de R^3.
Les paramètres :
- La variable `x` contient l'entrée du réseau de neurones. C'est un vecteur à 3 lignes

- La variable `K` indique le nombre de cellules du réseau (KxKxK).
- La variable `mu` contient les valeurs de mu.
- La variable `sigma` contient les valeurs de sigma.
- La variable `nbiter` indique le nombre d'itérations de l'apprentissage.

Le résultat :

- La variable `w` contient les poids synaptiques du réseau de Kohonen 3D. C'est un tableau de taille KxKxKx3.

2.2 Réduction du nombre de couleurs

Nous proposons maintenant d'utiliser votre réseau afin de réduire le nombre de couleur de l'image femme

– *Charger l'image femme.ppm*

– *Organiser les données de l'image couleur sous la forme d'un tableau de 3 lignes et 256*256 colones pour pouvoir les traiter à l'aide de votre programme*
Ceci peut se faire à travers le code suivant :
```
x=zeros(3,256*256); ind=1; for b=1:256, for bb=1:256, x(:,ind)=img(b,bb,:); ind=ind+1; end; end;
```

– *Appliquer l'apprentissage sur un réseau de 8*8*8 noeuds. Analyser les poids issus de l'apprentissage*
Nous allons maintenant pouvoir appliquer l'étape de codage qui consiste à remplacer chaque couleur de l'image d'origine par son code (le numéro du noeud et donc codé sur 3 valeurs comprises entre 1 et K). L'étape de décodage consiste à remplacer le code dans l'image compressée par la couleur associée.

– *Après apprentissage, remplacer les pixels couleurs de l'image d'origine par leurs valeurs approchées (étape de codage-décodage). Visualiser l'image résultant de la compression. Commenter et faire plusieurs tests en faisant varier K et nbiter.*
Nous allons tester la robustesse de ce codage au bruit.

– *Afin de visualiser l'intérêt d'utiliser un réseau de Kohonen, appliquer l'étape de codage. Ajouter ensuite un bruit léger sur ces trois entiers par pixel (modifier les coordonnées de quelques valeurs). Reconstruire alors la nouvelle image à partir du dictionnaire. Commenter la dégradation et conclure.*