MASTER : Traitement de l'Information et Exploitation des Données
Fouad Badran, Aymeric Chazottes, Cécile Mallet, Awa Niang, Charles Sorror, Sylvie Thiria

# TPC01 : Cartes topologiques (CT) : Données simulées et reconnaissance de chiffres manuscrits

## I - Les Objectifs

Les 2 parties de ce TP ont pour but de commencer à familiariser l'auditeur, avec le maniement des cartes topologiques et leur exploitation.

Partie 1 : Dans cette 1ère partie, nous travaillerons avec des données simulées qui représentent soit la lettre **Z** soit la lettre **F**. Nous chercherons à trouver une taille de carte topologique optimum d'abord en 1D puis en 2D. Dans le cas 2D nous complèterons les résultats avec classification par labellisation.

Partie 2 : Mise en œuvre des cartes topologiques pour la classification de chiffres manuscrits pour lesquels différents codages seront utilisés. On s'intéressera à trouver des paramètres d'apprentissage optimum et à montrer une représentation interne de la carte.

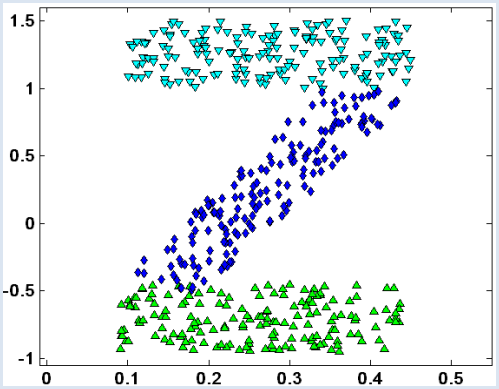
==========

*Le rapport de TP devra être synthétique. Il doit montrer la démarche suivie, et ne faire apparaître que les résultats nécessaires. Il s'agit de quantifier les résultats tout en rédigeant un rapport qui les analyse et les commente. Les paramètres utilisés devront être indiqués, Les graphiques des expériences doivent être insérés dans le rapport. Les résultats présentés devront être analysés et commentés.*

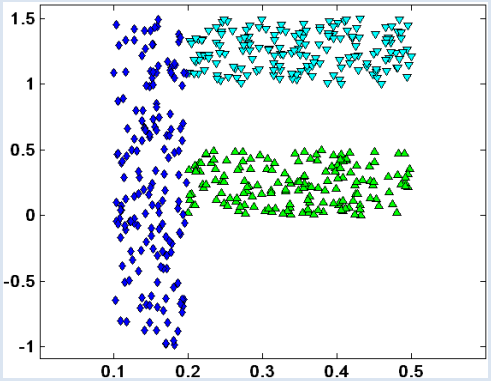
## II - Les Données

ŏ) Pour la 1ère partie du TP, les fonctions **Zcreadata** et **Fcreatdata**, mises à votre disposition servent à la génération des données simulées en 2D qui représentent respectivement la lettre Z et la lettre F, dont les points sont répartis en 3 classes. Deux paramètres sont utilisés par ces fonctions : N, pour indiquer le nombre total de données et classnames (optionnel), pour renseigner le nom des classes qui serviront à la labellisation des données (valeur par défaut : {'A', 'B', 'C'}).
En retour on obtient : X, l'ensemble des données, labs, les labels de classe associés aux données et cnames, les noms des composantes.

La fonction **lettreplot.m** nous permet de visualiser les données tirées, en représentant, par des formes différentes, les points des différentes classes :



pour le Z



pour le F

Les 3 classes sont représentées par des points de formes différentes qui sont placées sur chacune des barres qui forment la lettre :

| pour le Z | pour le F |
|---|---|
| - classe « top » : triangles dirigés vers le bas, | - classe « top » : triangles dirigés vers le bas, |
| - classe « bottom » : triangles dirigés vers le haut | - classe « middle » : triangles dirigés vers le haut |
| - classe « diag » : losanges | - classe « left » : losanges |

ŏ) Pour la 2ème partie du TP nous disposons du fichier **x.txt** qui est la base de données de chiffres manuscrits. Elle est composée de 480 chiffres codés en binaire (±1), dans une matrice 256x480. Cela signifie que chaque image binaire 16x16 a été transformée en un vecteur de dimension 256 qui, à son tour, correspond à une colonne de la matrice du fichier **x.txt**. Dans ce fichier, les pixels sont codés par les valeurs -1 et +1.

Ces données ont par ailleurs été codées selon les différentes conventions suivantes :

**HX** : Histogramme des projections du chiffre sur l'axe des x : dans chaque colonne on calcule le nombre de pixels noir – le nombre de pixels blancs. HX conduit à un vecteur de 16 composantes.

**HY** : Histogramme des projections du chiffre sur l'axe des y : dans chaque ligne on calcule le nombre de pixels noir – le nombre de pixels blancs. HY conduit aussi à un vecteur de 16 composantes.

**PH** : Profil Haut – pour chaque colonne, on code la coordonnée de la première transition blanc/noir en partant du haut. PH est un vecteur de 16 composantes.

**PB** : Profil Bas – pour chaque colonne, on code la coordonnée de la première transition blanc/noir en partant du bas. PB est un vecteur de 16 composantes.

**PG** : Profil Gauche – pour chaque ligne, on code la coordonnée de la première transition blanc/noir en partant de la gauche. PG est un vecteur de 16 composantes.

**PD** : Profil Droit – pour chaque ligne, on code la coordonnée de la première transition blanc/noir en partant de la droite. PD est un vecteur de 16 composantes.

(Il est à noter que les coordonnées sont indicées de la gauche vers la droite en ligne et de haut en bas pour les colonnes)

Ces conventions de codage peuvent être combinées pour former différents fichiers d'apprentissage. Nous disposons des cas suivants :

1 : codage HX seul ; vecteur de 16 composantes. Fichier d'entrée : **hx.txt**
2 : codage HX,HY ; vecteur de 32 composantes. Fichier d'entrée : **hx_hy.txt**
3 : codage PG,PD ; vecteur de 32 composantes. Fichier d'entrée : **pg_pd.txt**
4 : codage HX,HY,PG,PD ; vecteur de 64 composantes. Fichier d'entrée : **hx_hy_pg_pd.txt**
5 : codage PB,PH ; vecteur de 32 composantes. Fichier d'entrée : **pb_ph.txt**
6 : codage HX,HY,PB,PH ; vecteur de 64 composantes. Fichier d'entrée : **hx_hy_pb_ph.txt**

On précise que toutes les données de ces fichiers ont été de surcroît « normalisées » dans l'intervalle [-1, 1], SAUF **hx.txt**

**Préambule**

La réalisation d'une carte topologique fait intervenir un grand nombre de méta paramètres qui en complique la mise en œuvre. Ces métas paramètres portent aussi bien sur l'architecture de la carte que sur l'algorithme d'apprentissage.

Pour simplifier l'approche des CT, nous avons fait le choix d'une certaine configuration pour l'ensemble de nos TP. Rien n'empêchera bien sur le lecteur curieux d'en essayer d'autres. En particulier, nous avons toujours retenu des CT à connexions hexagonales, de forme rectangulaire à voisinage gaussien et initialisées de façon linéaire. Par ailleurs nous utiliserons deux étapes d'entrainement de la CT pour lesquelles nous avons retenu la version batch de l'algorithme d'apprentissage. La première étape utilise (en principe) une température initiale plus élevée permettant ainsi la prise en compte d'un voisinage élargi. La seconde étape, avec des températures plus petites permet un raffinement plus local de la quantification vectorielle.

La configuration que nous venons de décrire correspond d'ailleurs à celle proposée par défaut par la fonction d'apprentissage automatique (som_make) de la SOM_Toolbox.
Les principaux paramètres sur lesquels nous jouerons, selon les TP, sont la taille de la carte, les nombres d'itérations d'apprentissage et les températures initiales et finales.

Nous venons d'évoquer un premier niveau de difficulté que sont les aspects algorithmiques et techniques nécessaires à notre domaine d'expertise. Un second niveau concerne le choix, la pertinence des représentations utilisées pour l'étude d'un problème mais également et surtout notre capacité à les expliquer et à les interpréter. C'est souvent sur ce point de compétence que doit se concentrer l'attention et que l'effort doit être porté.

## V - 1ère Partie : Données simulées en forme de lettre Z et F

L'un des choix important dans la définition d'une CT est celui du nombre de neurones. C'est plus particulièrement ce paramètre que nous voulons mettre à l'épreuve dans l'exercice qui suit. Pour cela nous nous appuierons sur 2 jeux de données simulées qui représentent les lettres **Z** et **F**. On utilisera pour chaque lettre un ensemble de 500 points et on se limitera dans tous les cas à des cartes ne dépassant pas 50 neurones maximum (soit 10 formes en moyenne par neurone).

Pour ce travail, vous devez donc choisir le paramétrage et la dimension de la carte topologique
Remarque : Vous pourrez par exemple retenir les paramètres d'apprentissage suivants :

| | nombre d'itérations | Température initiale | Température finale |
|---|---|---|---|
| 1ère étape | 20 | 5 | 1.25 |
| 2ème étape | 50 | 1.25 | 0.10 |

Il est indiqué d'éviter de démultiplier les nombres d'expériences, nous vous proposons de garder toujours ces mêmes paramètres (ce qui n'est pas une obligation).

### Travail à faire

0°) Rappeler l'algorithme des K-moyennes. Rappeler l'algorithme des cartes topologiques. Faire le lien entre les deux méthodes et expliquer pourquoi les cartes topologiques sont souvent présentées comme une généralisation des K-moyennes.

1°) Travail sur des cartes de dimension 1.

Ici, on devra apprendre les deux lettres avec une carte avec une unique dimension. Il faudra donc se demander ce qu'implique l'apprentissage des données par la carte.

Pour chacune des 2 lettres :

- Se demander comment les données peuvent être apprises par la carte. Une carte est apprise pour obtenir le meilleur compromis entre erreur quadratique et erreur topologique. Ici il est possible de voir la carte dépliée au milieu des données. On pourra donc clairement voir si la topologie des cartes est préservée. **Pour chacune des lettres indiquer ce qu'implique une erreur quadratique faible.** (Si cela ne semble pas claire considérer quelques cartes apprises sur chacune des lettres.) **L'expliquer.**
- Retenir une carte ayant un nombre de neurones minimal (≤50) et bien dépliée au milieu des données. **Visualiser les cartes présentant les valeurs des référents pour chacune des dimensions. Faire le lien avec la carte dépliée au milieu des données.**
- Pour cette même carte, considérer la figure présentant les cardinalités de chacun des neurones. **Que dire de la répartition des données entre les neurones. Est-ce cohérent ? Expliquer.**
- Pour la carte retenue, **indiquer les erreurs de quantification et topographique. Indiquer et expliquer l'évolution des de ces deux erreurs lors de l'apprentissage.**
- **L'algorithme des cartes topologiques est non supervisé. Il faudra donc trouver des indicateurs permettant de faire ressortir la qualité de l'apprentissage des données.** Indiquer ce qui pourrait être fait. On a décidé de calculer le coefficient de Silhouette pour les différentes données. **Présenter cet indicateur et expliquer s'il est approprié ou pas.**
- Que dire de la valeur obtenue pour ce coefficient ? Expliquer.
- On effectue ensuite une agrégation des neurones de la carte par CAH. Indiquer qu'elle pourrait être l'intérêt de cette opération.
- On se trouve dans un cas de figure où l'on connait les classes des données. En considérant la carte dépliée au milieu des données, indiquer la qualité de la classification attendue.
- On a déterminé une matrice de confusion obtenue à partir d'un vote majoritaire. Indiquer le principe ayant permis de déterminer la classe des neurones.
- Discuter les performances en classification obtenues.

2°) Travail sur des cartes de dimension 2.

On reprend ce qui a été fait précédemment. On discutera l'apport d'une carte 2D. On discutera en particulier le dépliement de la carte au milieu des données. S'il y en a on expliquera d'éventuels neurones vides. On discutera aussi le coefficient de silhouette ainsi que les performances en classification.

## V - 2ème Partie : Reconnaissance de chiffres manuscrits

A la différence de la 1ère partie, les données utilisées pour la reconnaissance de chiffres sont des données réelles et non pas simulées. Pour ne pas trop démultiplier les expériences on vous propose d'utiliser une taille de carte fixée (12x12 par exemple) neurones et de limiter les valeurs des autres paramètres. Par contre vous devrez choisir le nombre d'itérations et les températures.

## Travail à faire

Ici on va travailler sur les fichiers de représentation des chiffres manuscrits (**x.txt, hx_hy.txt, pg_pd.txt et hx_hy_pg_pd.txt**). La qualité de ces différentes représentations à conserver l'information des chiffres manuscrits ont déjà été vues. Ici on apprendra deux représentations l'une permettant de restituer l'information des chiffres manuscrits l'autre non.

On pourra dans un premier temps se limiter aux 340 premiers exemples.

Ici les données sont en dimensions multiples, on ne pourra donc plus représenter la carte au milieu des données comme cela avait été fait précédemment. On va donc considérer un jeu de données un peu plus réaliste.

Ici, on va donc considérer l'apprentissage de cartes topologiques dans des conditions réalistes :

- Reprendre ce qui a été fait dans la partie précédente pour bien comprendre le passage à un problème de dimension réaliste.
- Déterminer un apprentissage permettant d'obtenir une carte optimale. Expliquer en quoi elle est optimale.
- Evaluer l'impact de la taille des répartitions des données en terme d'apprentissage. On pourra faire varier l'ensemble d'apprentissage selon les valeurs suivantes : Napp = {50, 100, 150, 200, 250, 300, 350, 400, 450}.
- On pourra essayer de déterminer les états d' «activation» de la carte pour les 10 premiers chiffres. Il faudra pour cela calculer la répartition des données entre les différents neurones de la carte. Une fois cela fait pour chacun des chiffres on discutera les valeurs fournies par la matrice de confusion.