Algorithme des k moyennes

Maxime CHAMBREUIL
maxime.chambreuil@insa-rouen.fr

1er avril 2003

Table des matières

1 Explication

1.1 Les fonctions

1.1.1 Distance

function d = distance (echantillon , centre , type)

En entrée :
echantillon : matrice de l'échantillon
centre : matrice des centres des clusters

type : 1 → distance euclidienne, 2 → distance de Mahalanobis

En sortie :
d : Matrice des distances

Cette fonction calcule la distance entre les points de l'échantillon et les centres des clusters suivant la définition de la distance demandée. Elle renvoie donc une matrice de la taille de l'échantillon (en ligne) et du nombre de centre (en colonne).

1.1.2 Affectation

function classe = affectation (echantillon , centre , type)

En entrée :
echantillon : matrice de l'échantillon
centre : matrice des centres des clusters
type : 1 → distance euclidienne, 2 → distance de Mahalanobis

En sortie :
classe : vecteur des classes d'appartenance des échantillons

Cette fonction calcule la distance entre les points de l'échantillon et les centres des clusters, grâce à la fonction *distance*. Pour chaque point de l'échantillon, elle renvoie ensuite le numéro du centre le plus proche.

1.1.3 Nouveaux centres

function centre = nouveaux_centres (echantillon , classe , nbCentre)

En entrée :
echantillon : matrice de l'échantillon
classe : vecteur des classes pour chaque individu
nbCentre : Nombre de clusters

En sortie :
centre : Matrice centre des clusters

Pour chaque cluster, on recherche les individus qui lui appartiennent. On fait ensuite la moyenne pour renvoyer le centre du cluster.

1.1.4 K moyennes

function [centre , classe] = k_moyennes (echantillon , nbCentre , type)

En entrée :
echantillon : matrice de l'échantillon
nbCentre : Nombre de clusters
type : 1 → distance euclidienne, 2 → distance de Mahalanobis

En sortie :
centre : Matrice centre des clusters
classe : Vecteur des classes d'appartenance pour chaque individu

On commence par initialiser aleéatoirement les centres des clusters avec des individus de l'échantillon. On affecte ensuite les individus aux clusters avec la fonction *affectation*, puis on calcule les nouveaux centres des clusters avec la fonction *nouveaux_centres*. On calcule la norme de la différence entre les nouveaux centres et les anciens. Si cette norme est inférieure à un seuil fixé et petit alors les centres ne bougent plus et on les retourne avec le vecteur des classes d'appartenance. Sinon, on continue : affectation, calcul des nouveaux centres, etc...

1.1.5 Formes fortes

[centreFF , classeFF] = formes_fortes (echantillon , N , nbCentre , type)

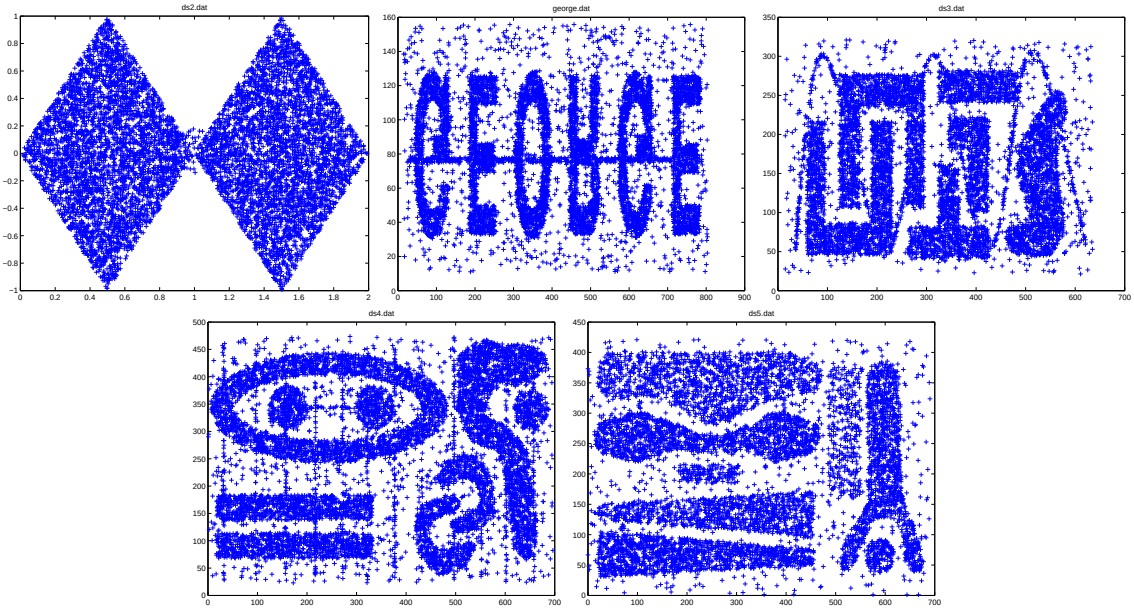
En entrée :
echantillon : matrice de l'échantillon
N : Nombre d'appels à la fonction *k_moyennes*
nbCentre : Nombre de clusters
type : 1 → distance euclidienne, 2 → distance de Mahalanobis

En sortie :
centreFF : Matrice centre des clusters
classeFF : Vecteur des classes d'appartenance pour chaque individu

On exécute N fois la fonction *k_moyennes* en sauvegardant les affectations successives. Ensuite, on récupère les indices des lignes identiques, chaque groupe d'individus dont les lignes sont identiques constitue une classe. On appelle ensuite la fonction *nouveaux_centres* pour calculer les centres des clusters.

1.2 Echantillons

Voici les différents échantillons que nous avions à disposition :



A partir de ses images, j'en ai déduit que je devais chercher :

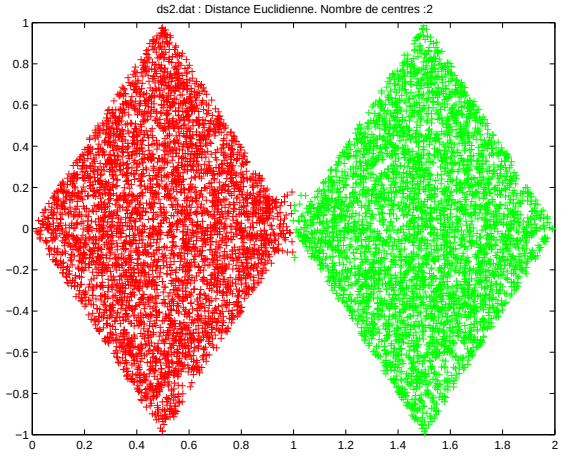
$$ds2 \rightarrow K = 2classes$$
$$george \rightarrow K = 6$$
$$ds3 \rightarrow K = 7$$
$$ds4 \rightarrow K = 9$$
$$ds5 \rightarrow K = 8$$

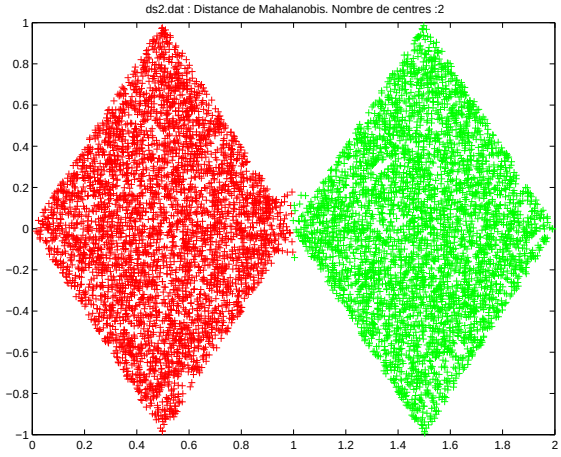
1.3 Initialisation des centres

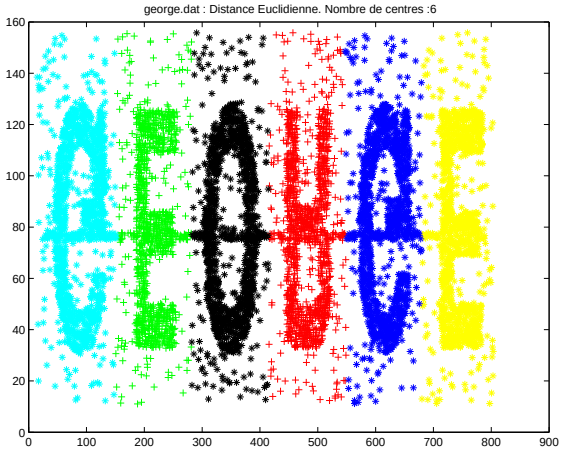
Lors de nos premiers tests, nous utilisions des centres générés aléatoirement. Cela nous a posé des problèmes de calcul de nouveaux centres : On obtenait des classes vides. Nous avons donc décidé de prendre comme centre d'un cluster un individu au hasard.

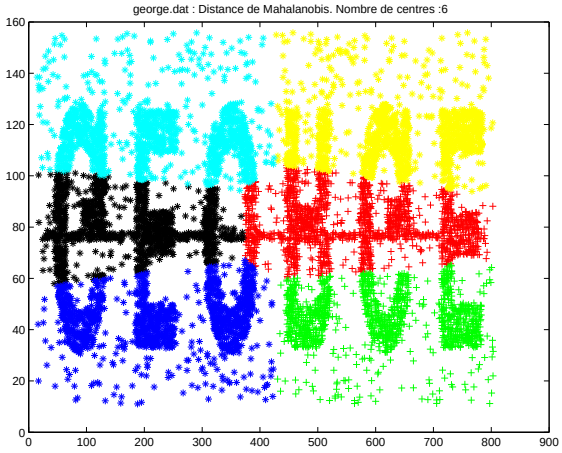
1.4 Distance d'Euclide vs Distance de Mahalanobis

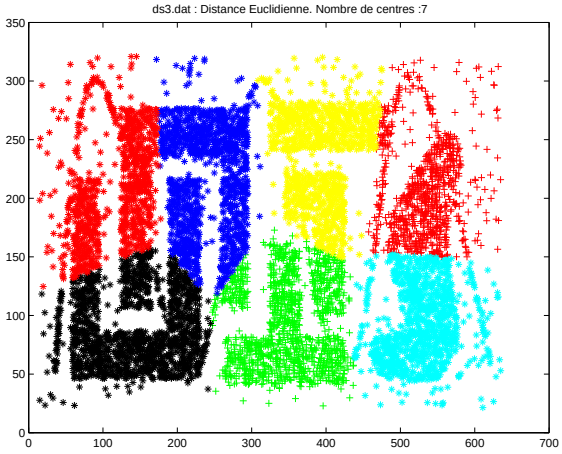
Voici pour chaque échantillon les figures obtenus, en calculant d'une part la distance euclidienne et d'autre part la distance de Mahalanobis, sans utiliser les formes fortes :

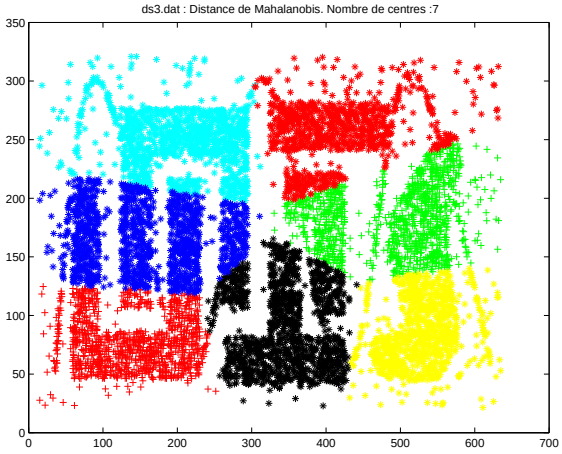

ds2.dat : Distance Euclidienne. Nombre de centres :2

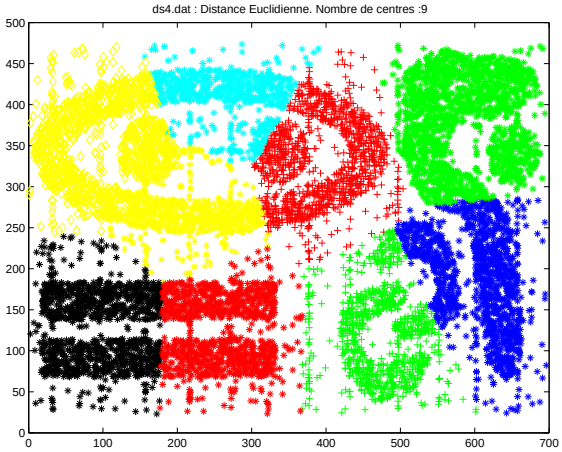

ds2.dat : Distance de Mahalanobis. Nombre de centres :2

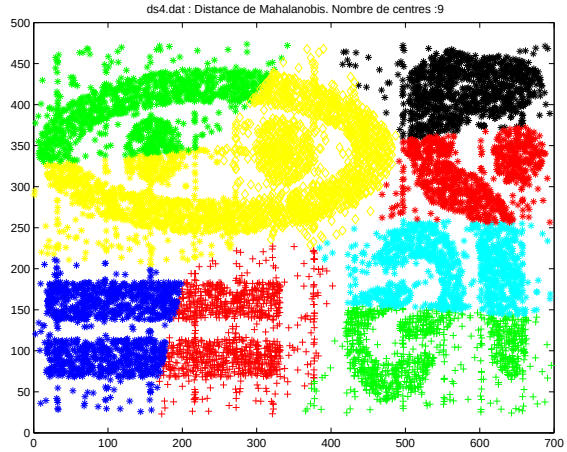

george.dat : Distance Euclidienne. Nombre de centres :6

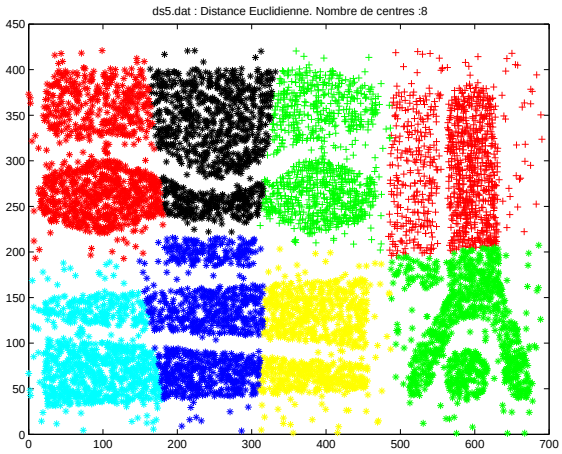

george.dat : Distance de Mahalanobis. Nombre de centres :6

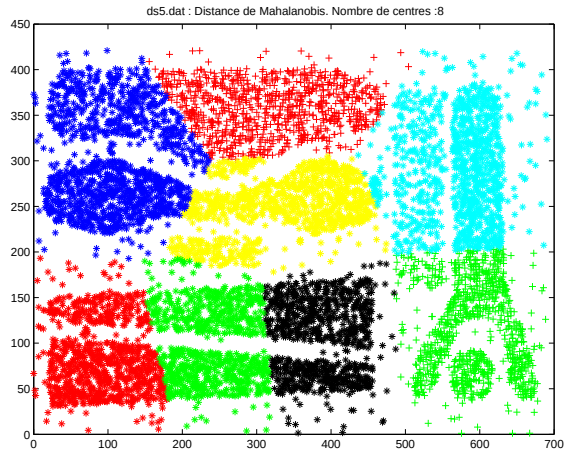

ds3.dat : Distance Euclidienne. Nombre de centres :7


ds3.dat : Distance de Mahalanobis. Nombre de centres :7



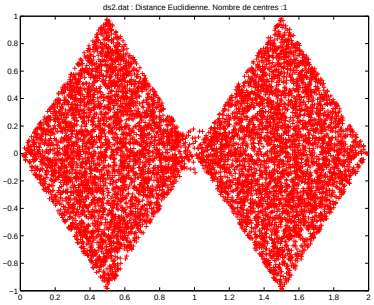


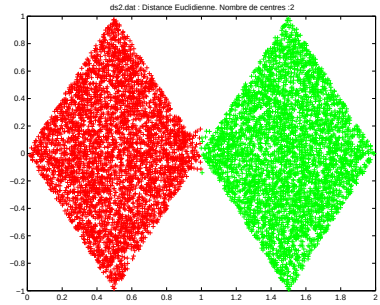


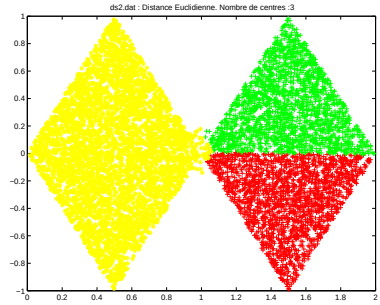


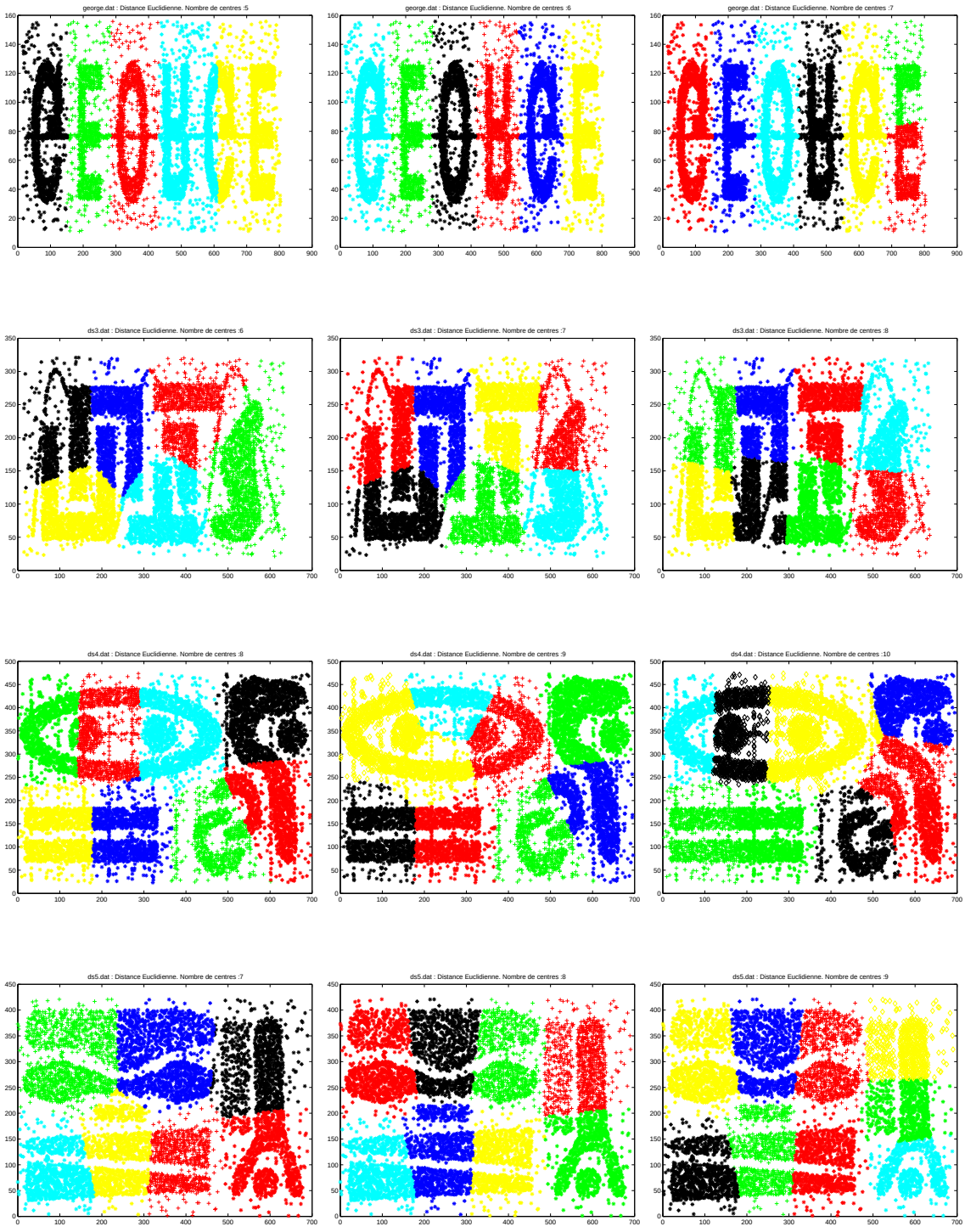
1.5 Variation suivant K - Distance Euclidienne

Pour chaque échantillon, voici les résultats obtenus en prenant comme nombre de classe K - 1, K et K + 1 :



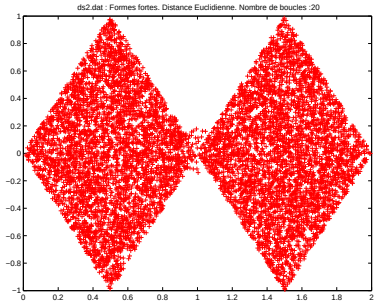


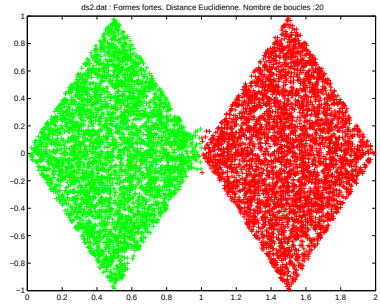


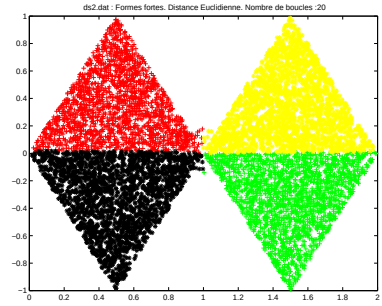


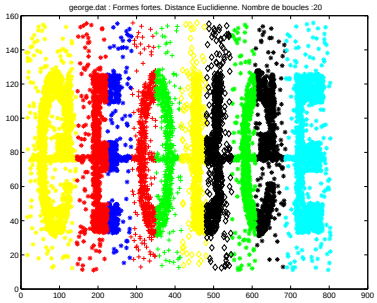
1.6 Variation suivant K - Distance Euclidienne - Formes fortes

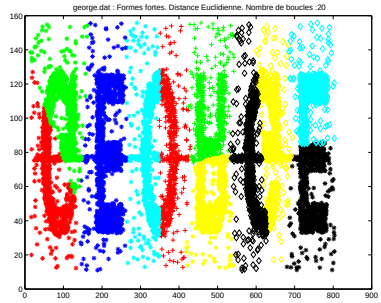
Pour chaque échantillon, on a gardé les formes fortes (N_f = 20) avec plus de 5% des individus. Voici les résultats obtenus en prenant comme nombre de classe K - 1, K et K + 1 :

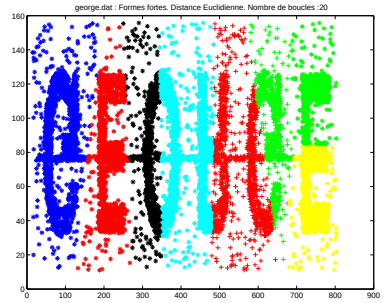

ds2.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20

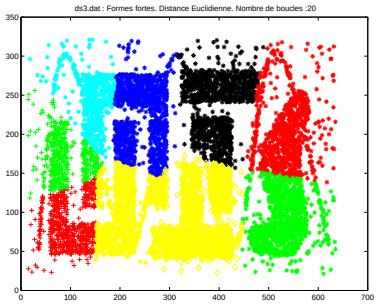

ds2.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20

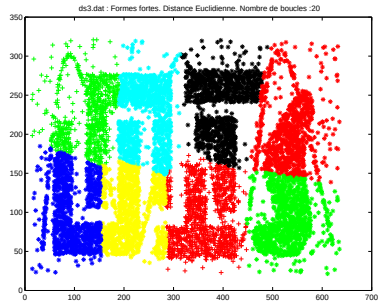

ds2.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20

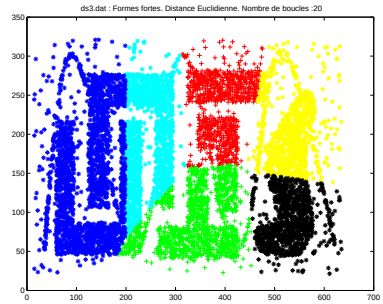

george.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20

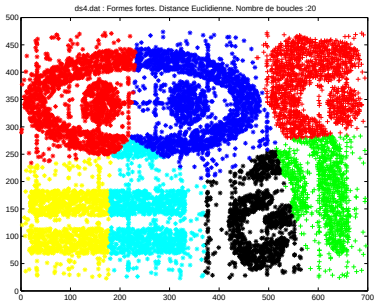

george.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20

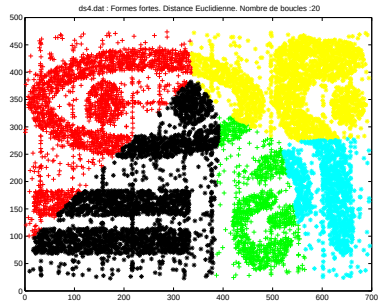

george.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20

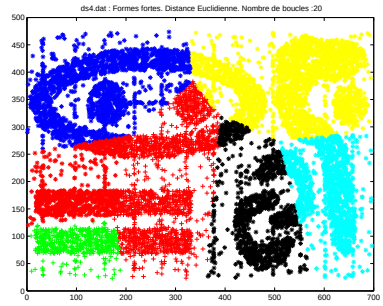

ds3.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20

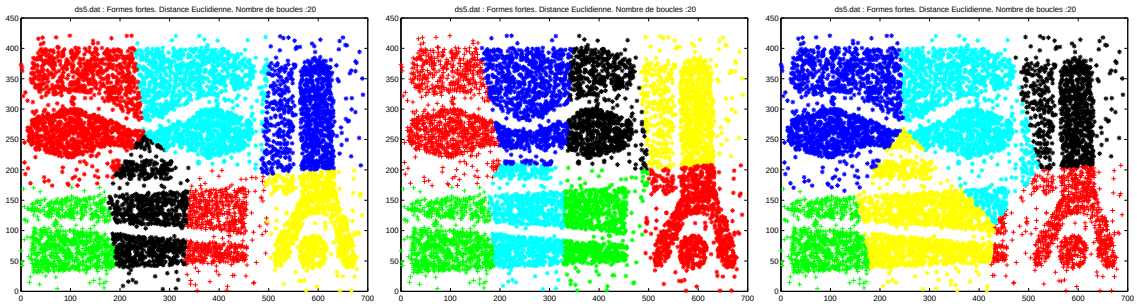

ds3.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20


ds3.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20


ds4.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20


ds4.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20


ds4.dat : Formes fortes. Distance Euclidienne. Nombre de boucles :20



2 Interprétation

2.1 Distance d'Euclide vs de Mahalanobis

On remarque que pour "ds2.dat" les résultats sont similaires, par contre pour "george.dat" la distance d'Euclide donne un bien meilleur résultat que celle de Mahalanobis. Pour les autres échantillons, il était difficile de dire qu'une méthode de calcul de la distance est meilleure qu'une autre sur un résultat. Après avoir comparé plusieurs fois les résultats, j'ai eu une préférence pour la distance d'Euclide, c'est pour cela que j'ai fait la suite du TP avec cette méthode de calcul.

2.2 Variation suivant K - Distance Euclidienne

Lors de l'augmentation de K, on peut dire qu'il y a une tendance à un redécoupage des classes : C'est flagrant pour "ds2.dat" et "george.dat", une classe est divisée en 2.

2.3 Variation suivant K - Distance Euclidienne - Formes fortes

Concernant "ds2.dat", on comprend l'intérêt des formes fortes : on demande 3 classes à l'algorithme des K-moyennes et on obtient 4 formes fortes, qui peuvent être considérées comme parfaites. Pour les autres échantillons, il est tout aussi difficile d'interpréter ses résultats : On ne peut pas dire que faire varier le nombre de classe apporte une information de qualité sur les formes fortes.

3 Conclusion

En conclusion, on peut dire que les formes fortes nous permettent de nous détacher du problème d'initialisation des centres : Elle a beau être aléatoire, on retrouve en général des formes fortes similaires.

J'en conclue aussi que l'algorithme des K-moyennes, même avec les formes fortes, n'est pas adapté à des classes convaxes : Je n'ai pas essayé mais je pense qu'il nous donnerait de très mauvais résultats sur une "cible de tir-à-l'arc" (classes concentriques).

J'en conclue aussi que si cet algorithme donne en 2 dimension ses résultats d'une piètre qualité, alors il ne doit pas être utilisé en dimension N.