

Université Lille 1 – Master 1

RDF – TP n°08

Classification non-supervisée – Algorithme des k-means

BARCHID Sami – SLIMANI Anthony
10/03/2019

Sommaire

<i>Introduction</i>	<i>2</i>
<i>Segmentation d'une image en niveaux de gris</i>	<i>3</i>
Présentation de l'image étudiée.....	3
Classification par k-means	4
Segmentation.....	5
<i>Segmentation d'une image couleur</i>	<i>6</i>
<i>Annexe 1 – Macro pour segmenter l'image avec les 4 cercles.....</i>	<i>8</i>
<i>Annexe 2 – Macro pour segmenter l'image du test d'Ishihara</i>	<i>10</i>

Introduction

L'objectif de ce TP est d'apprendre à classifier des données multi-variées par clustering dans un contexte d'apprentissage non-supervisé. L'outil de clustering qui sera vu ici est **kmeans**.

Ce rapport est divisé en 2 parties :

- **Segmentation d'une image en niveaux de gris** : application de la segmentation non-supervisée par clustering k-means sur une image en niveaux de gris.
- **Segmentation d'une image couleur** : application de la segmentation non-supervisée par clustering k-means sur une image en couleurs.

Segmentation d'une image en niveaux de gris

Le but de cette partie est de binariser une image en niveaux de gris grâce à la méthode de clustering k-means.

Présentation de l'image étudiée

L'image étudiée dans cette partie est composée de deux attributs : les niveaux de gris et les niveaux de texture sur un voisinage de 5 pixels. Ce sont les deux attributs que nous avons déjà utilisés auparavant dans le TP de segmentation et binarisation.

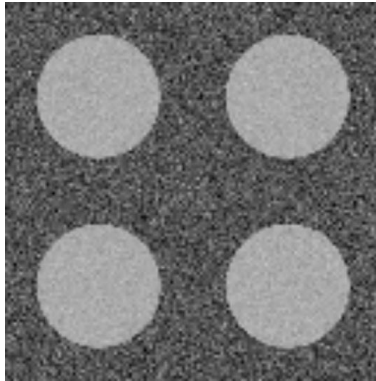


FIGURE 1 – NIVEAUX DE GRIS DE L'IMAGE ETUDIEE

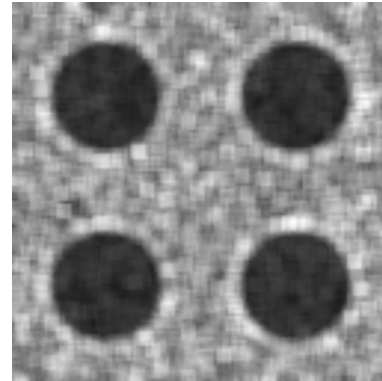


FIGURE 2 – NIVEAUX DE TEXTURE DE L'IMAGE ETUDIEE

Nous sommes alors capables d'afficher les observations représentant les pixels dans le plan de projection des deux attributs choisis. La figure 3 représente les nuages obtenus. Nous remarquons facilement que les points de l'image se séparent en deux nuages en figure 4.

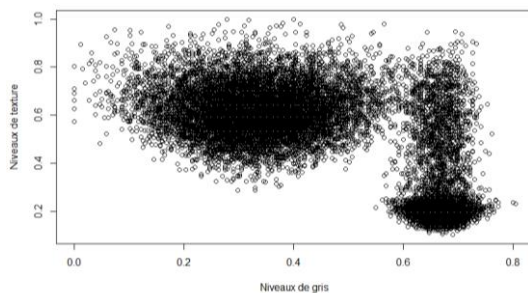


FIGURE 3 – REPRESENTATION DES PIXELS DANS LE PLAN DE PROJECTION DES DEUX ATTRIBUTS CHOISIS

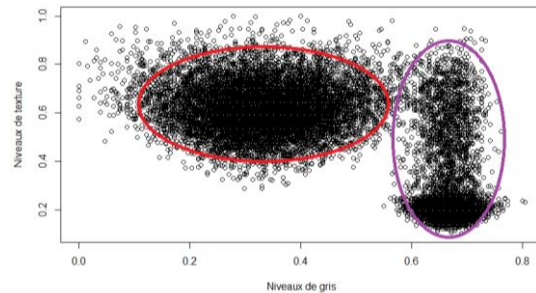


FIGURE 4 – IDENTIFICATION EMPIRIQUE DES DEUX NUAGES DE POINTS IDENTIFIES

- Un nuage dont le niveau de gris est plus élevé et le niveau de texture en majorité plus bas (entouré en mauve). Les pixels de ce nuage représentent les pixels qui composent le fond de l'image puisque ceux-ci ont un niveau de gris bas et varient beaucoup en niveau de texture (c'est pour ça que le fond de l'image est plus foncé que les objets et est composé de pixels de différentes valeurs de niveaux de gris).
- Un nuage dont le niveau de gris est moins élevé et le niveau de texture plus haut (entouré en rouge). Les pixels de ce nuage représentent les pixels qui composent les 4 disques puisque ceux-ci ont un niveau de gris élevé et varient peu en niveau de texture (ils sont quasiment tous gris clairs de façon homogène).

Classification par k-means

Nous allons donc appliquer la classification non-supervisée par k-means sur l'image présentée, qui va classer chaque pixel de l'image suivant sa distance avec un des centres de gravité des classes que nous avons définies.

Afin d'appliquer le clustering par k-means, l'utilisateur doit donc donner deux choses :

- Le nombre de classes à retrouver. Il faut donc choisir un nombre de classe pertinent pour obtenir une bonne classification.
- La condition d'arrêt du clustering. Ici, ce sera un nombre d'itération maximal, c'est-à-dire le nombre de fois qu'il va recalculer l'assignation des points aux classes et mettre à jour les centres de gravité des classes. Il est donc nécessaire de choisir un nombre d'itération au bout duquel le clustering par k-means a convergé, c'est-à-dire que la position des centres de gravité de va plus changer énormément entre deux itérations.

Comme nous avons identifié empiriquement deux nuages de points (le fond et les formes de l'image), nous avons décidé de compter deux classes. Nous avons aussi décidé d'un nombre d'itérations maximale de 30. La classification par k-means nous donne alors la figure suivante :

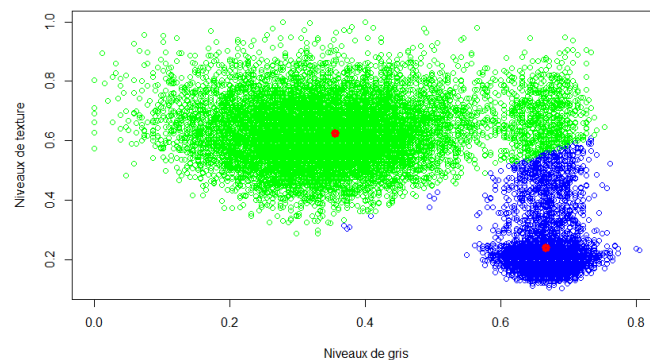


FIGURE 5 – CLASSIFICATION CLUSTERING PAR K-MEANS

- La première classe est en **vert**.
- La deuxième classe est en **bleu**
- Les centres de gravité des classes sont affichés en **rouge**.

Nous constatons que le clustering par k-means a classé les pixels de telle manière à respecter en grande partie les nuages que nous avons définis empiriquement. Cependant, nous remarquons que certains pixels ont été mal classés. Nous pourrions donc nous attendre à avoir une légère erreur de classification.

Segmentation

Après avoir obtenu les classes et leurs centres de gravité grâce au clustering par k-means, nous pouvons alors effectuer la binarisation de l'image. Pour ce faire, nous devons tout d'abord calculer, pour chaque pixel, la distance séparant l'observation associée au centre de gravité de chaque classe. Après avoir récupéré ces distances, nous pouvons obtenir l'image segmentée qui résulte du minimum des deux distances. Nous obtenons ainsi l'image segmentée suivante :



FIGURE 6 – IMAGE SEGMENTÉE

Nous disposons, pour ce TP, d'une image de référence permettant de mesurer le taux de bonne classification de l'image. La figure suivante représente l'image de référence :

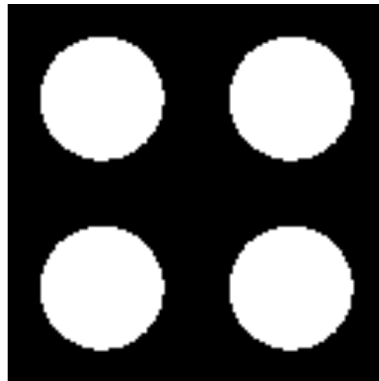


FIGURE 7 – IMAGE DE RÉFÉRENCE

Le calcul du taux de bonne classification nous donne un résultat de $0.95 = 95\%$ de **bonne classification**. Nous comprenons donc que le clustering par k-means est une technique en majorité efficace dans le cas de notre image pour segmenter une image dans un contexte d'apprentissage non-supervisé. Nous observons aussi que, comme nous l'avons dit précédemment, nous remarquons de légères erreurs de segmentation dans l'image. Cela correspond aux points ayant été classés dans la mauvaise classe par rapport aux deux nuages que nous avons définis empiriquement. Ce problème s'explique du fait que le clustering par k-means est plus adapté sur des nuages sphériques. Or, dans notre cas, les nuages sont plus ovales que sphériques. De ce fait, certains points du nuage (les plus éloignés du centre de gravité) sont plus proches du centre de gravité de l'autre nuage de points. De ce fait, ils sont mis dans la mauvaise classe.

Segmentation d'une image couleur

Pour finir notre TP, nous avons appliqué la classification de kmeans sur une image du test d'Ishihara représentée en figure 8. On utilise la planche n°8 représentant un 15 en vert sur un cercle de fond orange/rouge. Elle est utilisée pour détecter une déficience entre le rouge et le vert.

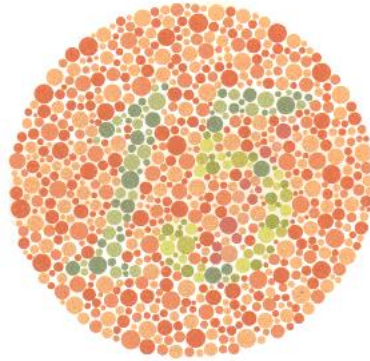


FIGURE 1 - IMAGE DU TEST DE D'ISHIHARA

On peut distinguer à première vue 3 classes. La première est celle qui représente les pixels de fond de couleur blanc. La seconde correspond aux pixels du cercle orange/rouge. Et la dernière correspond aux pixels du chiffre 15 de couleur vert. Notre but est de segmenter cette image pour retrouver ces trois classes notamment la troisième qui permettra de détecter le chiffre 15. On a appliqué la méthode de kmeans avec $k=3$, on a pu constater que la classification obtenue n'était pas satisfaisante. Aucune des clusters obtenus ne permet de distinguer les pixels du chiffre 15 du reste. On a alors ajouté 2 classes supplémentaires afin d'améliorer nos résultats.

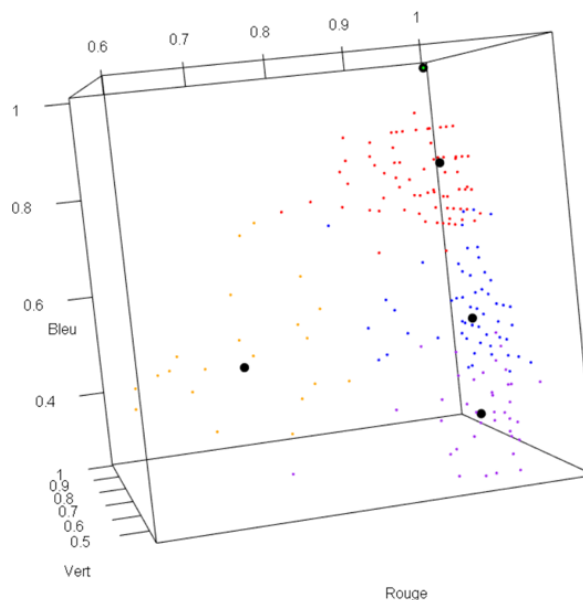


FIGURE 8 - REPRESENTATION GRAPHIQUE DE LA CLASSIFICATION DE L'IMAGE AVEC LES CENTRES DE GRAVITE DES CLASSES EN NOIR

Le tableau suivant présente les résultats obtenus pour un nombre de classes $k = 5$:




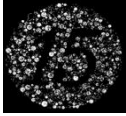

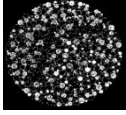
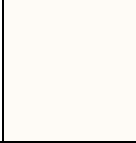
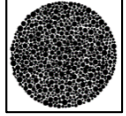


	Rouge	Vert	Bleu	Hexa	Couleur	Résultat
Cluster 1	0,6959008	0,700892	0,466274	#B1B377		
Cluster 2	0,9438535	0,6222458	0,4514542	#F19F73		
Cluster 3	0,9723902	0,7705071	0,5525878	#F8C48D		
Cluster 4	0,9980209	0,983082	0,9660367	#FEFBF6		
Cluster 5	0,9079452	0,493674	0,344287	#E87E58		

TABLEAU 1 - KMEANS SUR UNE IMAGE DU TEST D'ISHIHARA

Le cluster 1 nous permet de détecter le chiffre 15 car son centre de gravité a pour couleur un vert kaki. Les clusters 2, 3 et 5 nous permettent de détecter le cercle de couleur orange tandis que le cluster 4 nous permet de déterminer les pixels de fond de couleur blanche.

Le fait que nous ayons choisis 5 classes est justifié par le fait que la répartition des points de l'image n'est pas sphérique. En effet, on remarque que les cercles rouges sur l'image d'Ishihara varient du orange au rouge plus foncé, ce qui a pour effet d'étendre la répartition des points du nuage rouge. Le problème, c'est que le clustering par k-means utilise la distance à un centre de gravité pour définir la classe d'un pixel. Par conséquent, si nous avions conservé un nombre de classes $k=3$, nous nous serions retrouvés avec des pixels du nuage rouge qui se serait retrouvés plus proches du nuage de pixels blancs ou du nuage de pixels vert.

Le fait d'avoir divisé le nuage de pixels rouge en trois classes de pixels rouge a permis d'utiliser trois centres de gravité dans le nuage de pixels rouge, ce qui a pu augmenter la précision de la classification (puisque nous avons moins de chance qu'un pixel rouge soit plus proche du centre de gravité vert ou blanc que d'un des centres de gravité rouge).

Annexe 1 – Macro pour segmenter l'image avec les 4 cercles

```
library ("EBImage")

# Fonction de chargement d'une image en niveaux de gris
rdfReadGreylImage <- function (nom) {
  image <- readImage (nom)
  if (length (dim (image)) == 2) {
    image
  } else {
    channel (image, 'red')
  }
}

#####
# Lecture des deux attributs de l'image étudiée.
#####
reference <- rdfReadGreylImage('rdf-masque-ronds.png') # lecture de l'image de reference pour comparer les
résultats
image1 = rdfReadGreylImage("rdf-2-classes-texture-1.png") # attribut 1 => niveaux de gris
image2 = rdfReadGreylImage("rdf-2-classes-texture-1-text.png") # attribut 2 => niveaux de texture
# Afficher les deux attributs
display (image1, "rdf-2-classes-texture-1.png", method="raster", all=TRUE)
display (image2, "rdf-2-classes-texture-1-text.png", method="raster", all=TRUE)
# Conversion en une matrice des données de taille 16384x2
imagev<-cbind(as.vector(image1), as.vector(image2))
#####

#####
# Affichage des observations de l'image
#####
# Tout afficher en noir car aucune classification
plot(imagev, col="black", xlab = "Niveaux de gris", ylab = "Niveaux de texture", title = "Observations par pixel")
#####

#####
# Application des kmeans sur les données
# (classification non-supervisée en deux classes)
# (on recalcule l'assignation des points et la mise à jour des centres de gravité 30 fois)
#####
ikm<-kmeans(imagev, 2, iter.max = 30)
#####
```

```
#####
# Affichage des observations de l'image après classification par kmeans
#####
couleur<- rep("green", length(imagev)) # classe 1 en vert
couleur[ikm$cluster==2]="blue" # Classe 2 en bleu
centers_aff<-cbind(ikm$centers[,1], ikm$centers[,2]) # définition des centres de gravité
plot(imagev, col=couleur, xlab = "Niveaux de gris", ylab = "Niveaux de texture") # Affichage des points colorés
selon la classe
points(centers_aff, col="red", lwd=5) # affichage des centres de gravités en rouge
#####

#####
# Calcul des distances entre les attributs des pixels et les centres de gravités
#####
# distance entre les attributs des pixels et le centre de gravité 1
distance_cdg1 <- (image1-ikm$centers[1,1])**2 + (image2-ikm$centers[2,1])**2
# distance entre les attributs des pixels et le centre de gravité 2
display (distance_cdg1, "distance_cdg1", method="raster", all=TRUE)

#Distance entre les attributs des pixels et le cdg de la classe 2
distance_cdg2 <- (image1-ikm$centers[1,2])**2 + (image2-ikm$centers[2,2])**2
display (distance_cdg2, "distance_cdg2", method="raster", all=TRUE)
#####

#####
# Segmenter l'image en calculant l'image qui résulte du minimum des deux distances
#####

# On affiche en blanc les pixels qui sont plus proches du centre de gravité 2 que du 1
binarisation1 = distance_cdg1 - distance_cdg2 >= 0
display (binarisation1, "Segmentation", method="raster", all=TRUE)
# Calcul du taux de bonne classification via l'image de reference
taux_bonne_classification1 <- 1 - sum(abs(binarisation1 - reference)) / length(binarisation1)

# On affiche en blanc les pixels qui sont plus proches du centre de gravité 1 que du 2
binarisation2 = distance_cdg1 - distance_cdg2 < 0
display (binarisation2, "Segmentation", method="raster", all=TRUE)
# Calcul du taux de bonne classification via l'image de reference
taux_bonne_classification2 <- 1 - sum(abs(binarisation2 - reference)) / length(binarisation2)
#####
```

Annexe 2 – Macro pour segmenter l'image du test d'Ishihara

```
library("EBImage")
library(rgl) # utilisé pour les plot3d

#####
# Lecture des deux attributs de l'image étudiée.
#####
# Lecture de l'image
dalton15 = readImage("cas_3_dalton15.png")

#Conversion en une matrice des données 82360x3 (pour chaque pixel, on a un attribut Rouge vert et bleu)
imagev<-cbind(as.vector(dalton15[,1]), as.vector(dalton15[,2]), as.vector(dalton15[,3]))

display (dalton15, "cas_3_dalton15.png", method="raster", all=TRUE)

# définition des valeurs des attributs
attribut1 = dalton15[,1] # attribut de rouge
attribut2 = dalton15[,2] # attribut de vert
attribut3 = dalton15[,3] # attribut de bleu
#####

#####
# Affichage des observations de l'image
#####
# Tout afficher en noir car aucune classification
# Affichage 3D de imagev (nécessite RGL)
plot3d(attribut1, attribut2, attribut3,
      type="p",
      xlab="Rouge",
      ylab="Vert",
      zlab="Bleu")
#####

#####
# Application des kmeans sur les données
# (classification non-supervisée en cinq classes)
# (on recalcule l'assignation des points et la mise à jour des centres de gravité 150 fois)
#####
ikm<-kmeans(imagev, 5, iter.max = 150)

# Récupération des centres de gravité
cdg1 = ikm$centers[1,]
cdg2 = ikm$centers[2,]
```

```

cdg3 = ikm$centers[3,]
cdg4 = ikm$centers[4,]
cdg5 = ikm$centers[5,]
#####

#####

# Affichage des observations de l'image après classification par kmeans
#####

# Affichage du graphique des pixels suivant leurs attributs avec les couleurs des classes
couleur<- rep("red", length(imagev)) # Classe 1 en rouge
couleur[ikm$cluster==2]="green" # classe 2 en vert
couleur[ikm$cluster==3]="blue" # classe 3 en bleu
couleur[ikm$cluster==4]="orange" # classe 4 en orange
couleur[ikm$cluster==5]="purple" # classe 4 en orange
plot3d(attribut1, attribut2, attribut3, col=couleur, xlab = "Rouge", ylab = "Vert", zlab="Bleu")

# Afficher les centres de gravité en noir (et en plus gros pour les remarquer)
points3d(ikm$centers, col="black", size= 10)
#####

#####

# Calcul des distances entre les attributs des pixels et les centres de gravités
#####

#Distance entre les attributs des pixels et le centre de gravité de la classe 1
distance_cdg1 <- (attribut1 - cdg1[1])**2 + (attribut2-cdg1[2])**2 + (attribut3-cdg1[3])**2

#Distance entre les attributs des pixels et le centre de gravité de la classe 2
distance_cdg2 <- (attribut1 - cdg2[1])**2 + (attribut2-cdg2[2])**2 + (attribut3-cdg2[3])**2

#Distance entre les attributs des pixels et le centre de gravité de la classe 3
distance_cdg3 <- (attribut1 - cdg3[1])**2 + (attribut2-cdg3[2])**2 + (attribut3-cdg3[3])**2

#Distance entre les attributs des pixels et le centre de gravité de la classe 4
distance_cdg4 <- (attribut1 - cdg4[1])**2 + (attribut2-cdg4[2])**2 + (attribut3-cdg4[3])**2

#Distance entre les attributs des pixels et le centre de gravité de la classe 5
distance_cdg5 <- (attribut1 - cdg5[1])**2 + (attribut2-cdg5[2])**2 + (attribut3-cdg5[3])**2

# mettre le mode de couleur en niveaux de gris pour la binarisation
colorMode(distance_cdg1) <- 0
colorMode(distance_cdg2) <- 0
colorMode(distance_cdg3) <- 0
colorMode(distance_cdg4) <- 0
colorMode(distance_cdg5) <- 0

display (distance_cdg1, "distance_cdg1", method="raster", all=TRUE)

```

```

display (distance_cdg2, "distance_cdg2", method="raster", all=TRUE)
display (distance_cdg3, "distance_cdg3", method="raster", all=TRUE)
display (distance_cdg4, "distance_cdg4", method="raster", all=TRUE)
display (distance_cdg5, "distance_cdg5", method="raster", all=TRUE)
#####

#####
# Segmenter l'image en calculant l'image qui résulte du minimum des distances à chaque centre de gravité
#####
# Afficher en blanc les points plus proches du centre de gravité 1 que des autres
display ((distance_cdg1 - distance_cdg2) <= 0 & (distance_cdg1 - distance_cdg3) <= 0 & (distance_cdg1 -
distance_cdg4) <= 0 & (distance_cdg1 - distance_cdg5) <= 0, "classe1", method="raster", all=TRUE)

# Afficher en blanc les points plus proches du centre de gravité 2 que des autres
display ((distance_cdg2 - distance_cdg1) <= 0 & (distance_cdg2 - distance_cdg3) <= 0 & (distance_cdg2 -
distance_cdg4) <= 0 & (distance_cdg2 - distance_cdg5) <= 0, method="raster", all=TRUE)

# Afficher en blanc les points plus proches du centre de gravité 3 que des autres
display ((distance_cdg3 - distance_cdg1) <= 0 & (distance_cdg3 - distance_cdg2) <= 0 & (distance_cdg3 -
distance_cdg4) <= 0 & (distance_cdg3 - distance_cdg5) <= 0, method="raster", all=TRUE)

# Afficher en blanc les points plus proches du centre de gravité 4 que des autres
display ((distance_cdg4 - distance_cdg1) <= 0 & (distance_cdg4 - distance_cdg2) <= 0 & (distance_cdg4 -
distance_cdg3) <= 0 & (distance_cdg4 - distance_cdg5) <= 0, "classe4", method="raster", all=TRUE)

# Afficher en blanc les points plus proches du centre de gravité 5 que des autres
display ((distance_cdg5 - distance_cdg1) <= 0 & (distance_cdg5 - distance_cdg2) <= 0 & (distance_cdg5 -
distance_cdg3) <= 0 & (distance_cdg5 - distance_cdg4) <= 0, "classe4", method="raster", all=TRUE)
#####

```