Théo KERNEVES
Forage de données
TP n°2

Compte-rendu

Introduction

Ce sujet de TP consiste à implémenter des algorithmes d'apprentissage non supervisé dans le but d'effectuer un clustering de couleurs à partir d'images données.
Les algorithmes à implémenter sont :

- K-means
- DBSCAN

Étant plutôt à l'aise avec Java, j'ai décidé d'utiliser ce dernier dans le cadre du TP. J'ai utilisé JavaFX pour le développement de l'interface graphique et la manipulation d'images.

Application

Pour utiliser l'application, à partir d'un terminal, il suffit de se placer dans le répertoire où se situe le fichier ColorClustering.jar et exécuter la commande suivante :

- *java -jar ColorClustering.jar*

A noter que l'application a été développée en Java 1.8

La première chose à faire est de cliquer sur le bouton « Choisir Image » en haut à gauche. Une fenêtre s'ouvrira vous permettant de choisir l'image.

Une fois fait, vous devriez avoir un aperçu de l'image dans l'application (*figure 1*).

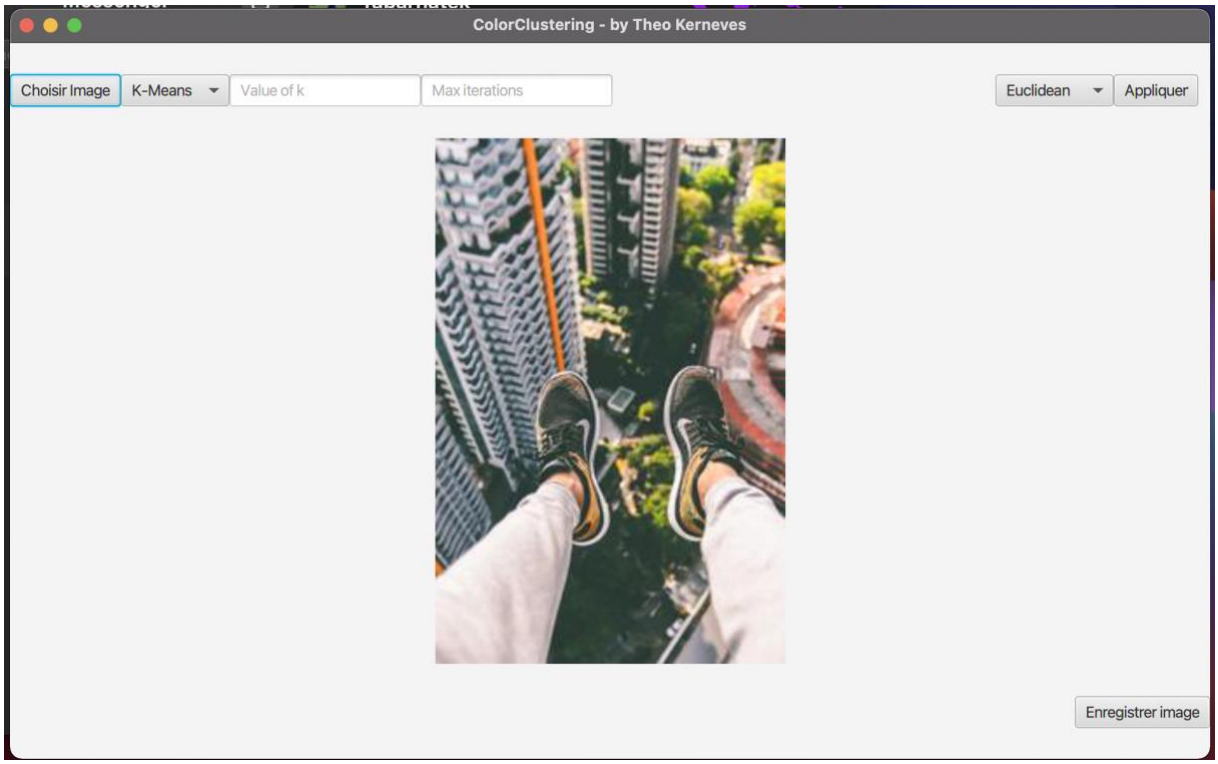

Figure 1 - ColorClustering Preview

Le premier paramètre à choisir est l'algorithme à utiliser (K-Means ou DBSCAN). Les autres paramètres à choisir seront différents en fonction de l'algorithme.
Dans le cas de K-Means :

- Valeur de k : nombre de clusters
- Max iterations : le nombre d'itérations total avant que l'algorithme ne s'arrête (critère d'arrêt)

Dans le cas de DBSCAN :

- Minimum points (m dans le cours) : nombre minimum de points regroupés pour qu'un ensemble de points soit considéré comme dense.
- Epsilon : distance maximum entre deux points pour qu'ils soient considérés être dans le même cluster. Deux points sont considérés voisins si leur distance est inférieure à epsilon.

Dans tous les cas, il faut choisir la distance entre deux couleurs à utiliser dans les algorithmes :

- Distance euclidienne
- Distance de Manhattan

Pour finir, cliquer sur « Appliquer ». Après un certain temps (cela peut être long en fonction de la taille de l'image), vous devriez voir le résultat apparaître et vous pouvez enregistrer l'image en cliquant sur « Enregistrer Image ».

Réponses aux questions

Pour les tests, je vais utiliser une image du site https://picsum.photos/images

Image utilisée : https://picsum.photos/seed/forage1/500/500 (mer)

Pour les tests avec K-Means je vais à chaque fois utiliser une valeur de 1000 pour « Max iterations ».

Question 4

Analyse K-Means :
Le seul paramètre variable dans ce cas est k : le nombre de clusters.
On remarque que plus k grandit et plus on va avoir de détails sur l'image finale. C'est normal puisqu'on offre plus de possibilités de couleurs.

Analyse DBSCAN :
C'est un peu plus délicat car il y a deux paramètres variables « min points » et « epsilon ».

On remarque que plus « min points » est faible et plus il va être « facile » d'obtenir des clusters plus nombreux à la fin (et donc une palette de couleurs disponible plus large).

Plus epsilon grandit et plus on va être amené à faire grossir les clusters « de proche en proche » (donc moins de clusters à la fin).

Si la distance est trop faible et que le nombre minimum de points pour créer un cluster est relativement petit aussi, on peut se retrouver avec des points non classifiés (couleur jaune sur les images output).

Question 5

K-Means :
J'ai l'impression que la distance de Manhattan va saturer l'image et va donner une image moins détaillée avec une valeur de k égale. Dans cet algorithme, la distance de Manhattan représente peut-être moins bien la distance entre deux couleurs.

DBSCAN :
Par contre, dans le cas de DBSCAN, j'ai l'impression que la distance de Manhattan donne des images beaucoup plus détaillées qu'avec la distance euclidienne avec moins de « bruit ».

Améliorations

Mon rendu est loin d'être parfait et nous pouvons imaginer quelques améliorations.
Premièrement, on pourrait trouver un moyen de faire arrêter k-means automatiquement sans avoir à spécifier « max iterations ».

Ensuite, même si de manière générale les algorithmes k-means et DBSCAN peuvent être optimisés, c'est surtout la génération d'images finales qui peut être améliorée. En effet, elle prend beaucoup de temps alors que c'est une étape qui devrait être négligeable par rapport à l'algorithme d'apprentissage.

L'architecture globale du code peut aussi être améliorée : utilisation d'interfaces, de classes abstraites… pour éviter la duplication de code.

Bilan

Ce TP m'a permis de me familiariser avec les algorithmes d'apprentissage non supervisé, de les mettre en pratique et d'observer les résultats de manière très visuelle (les couleurs d'une image).

J'ai aussi pu comprendre l'importance du choix de la distance dans ce cadre là