

Informatique 2^{ème} année

K moyennes et compression d'image

Dans une image, chaque pixel est habituellement représenté par trois octets, ce qui donne $256^3 = 16777216$ couleurs possibles.

Une technique pour réduire le poids de chaque pixel consiste à réduire le nombre de couleurs possibles.

Question 1 : Un fichier image est fourni dans le dossier de la classe. En utilisant son chemin d'accès, exécuter l'instruction :

```
original = plt.imread( chemin ).tolist()
```

Question 2 : La liste `original` ainsi créée est une liste de listes de pixels (et chacun de ces pixels est une liste de taille 3). Construire une liste `data0` réunissant tous les pixels de l'image.

Question 3 : Une première technique, primaire, pour réduire le nombre de couleurs possibles, est d'arrondir chaque coordonnée de chaque pixel, par exemple à l'entier divisible par 5 le plus proche. Par exemple, le pixel [144, 207, 68] sera arrondi à [145, 205, 70]. Ces arrondis étant effectués, combien reste-t-il de couleurs possibles ? Construire une liste `data`, réunissant tous les pixels de `data0`, arrondis suivant ce principe.

Dans la suite de ce TP, on applique l'algorithme des k moyennes sur l'ensemble des couleurs présentes dans une image donnée, afin de choisir une palette de k couleurs. On constituera avec ces k couleurs une image qui sera proche de l'image de départ. Afin de réduire les temps d'exécution, on travaillera sur des couleurs arrondies suivant le modèle précédent. On travaillera donc à partir de la liste `data` et non de `data0`.

Question 4 : Définir une fonction `init(data,k)` qui fait l'initialisation de l'algorithme des moyennes de la façon la plus simple : elle renvoie k points distincts au hasard parmi les données pour être les points moyens. On pourra utiliser la fonction `sample` du module `random`.

Question 5 : Définir une fonction `repartition(data,moyens)` qui fait l'étape de «répartition» (ou d'«affectation») de l'algorithme des k moyennes : elle renvoie une liste `C` telle que `C[i]` est la classe dont le point moyen est le plus proche du point `data[i]`. On pourra s'appuyer sur d'autres fonctions qu'on définira et dont on donnera une spécification précise.

Il faut maintenant passer à l'étape de mise à jour des points moyens, en calculant le barycentre de chaque classe.

Question 6 : Définir une fonction `addition(x,y)`, une fonction `multiplication(a,x)`, une fonction `arrondi(x)` qui effectuent respectivement l'addition de deux vecteurs, la multiplication d'un vecteur par un scalaire, l'arrondi des coordonnées d'un vecteur à l'entier divisible par 5 le plus proche.

Question 7 : Définir la fonction `mise_a_jour(data,C,k)` qui se sert de la liste `C` comme répartition et renvoie une la liste des barycentres des k classes.

Question 8 : Définir la fonction `kmeans(donnees,k)` qui applique l'algorithme des k moyennes jusqu'à sa convergence, et renvoie deux listes : celle des points moyens, et celle de la répartition des données dans les k classes (noté `C` précédemment).

Une fois qu'on a appliqué l'algorithme de partitionnement aux couleurs de l'image originale, on peut construire l'image «compressée» en remplaçant dans l'image d'origine chaque pixel par la moyenne de la classe dont il fait partie.

Question 9 : Définir la fonction `construction(original, moyens, C)` qui construit ainsi l'image compressée, aux mêmes dimensions que l'image originale.

Question 10 : Appliquer la procédure sur l'image fournie, avec $k = 32$, afficher l'image compressée obtenue.

Question 11 : Avec une palette de 32 couleurs au lieu de 16777216, combien de bits faut-il utiliser pour indiquer la couleur d'un pixel ?