RDF : TP 8 : Compression d'image par méthode de clustering

binôme : Benjamin Ruytoor et Aurore Allart
date : 15 mars 2013

Introduction

Dans ce TP, nous allons utiliser l'algorithme K_means pour compresser des images. Cette algorithme est en apprentissage non supervisé. C'est algorithme de partitionnement de données, c'est à dire une méthode dont le but est de diviser des observations en clusters.

1. Implémentation de K_Means

○ Question 1

Un point de donnée est de taille 3, car chaque pixel contient 3 valeurs de couleur : pour la valeur de rouge, pour la valeur de vert et pour la valeur de bleu.

○ Question 2

La taille de l'ensemble des données est de n* la taille de l'image. Donc 3*16384.

○ Commentaire sur l'algorithme

Dans l'ensemble c'était assez simple à comprendre. Par contre je n'ai pas eu besoin d'itérer 10 fois k_means, j'obtiens tout de suite une image compressée. Si j'essaye d'itérer 10 fois, j'obtiens une image en 2 couleurs environ.

○ Résultat

Avec le code ci-dessous,nous obtenons cette image :

Figure 1 : l'image compressée.

2. Code

```
//Charger l'image
A = imread('bird_small.png');
//Diviser par 255 pour avoir les valeurs entre 0 et 1
B = im2double(A);
img_size=size(A);

//Transformer la matrice A en une matrice de Nx3 où
//N = nombre des pixels et chaque ligne contient les valeurs R G et B du pixel. Ceci donne les
donnee à clustere:
X = matrix(B, img_size(1) * img_size(2), 3);
```

```
//Clusterer les pixels en K=16 clusters (i.e. 16 couleurs)
K=16;
//Fixer le maximum nombre d'iterations de k-means à 10
max_iter=10;
//N'oubliez pas d'initialiser les premiers centres aléatoirement
[X_ligne,X_col] = size(X);
I = grand(1,'prm',(1:X_ligne)');
Y = zeros(X_ligne,1);
//création du tableau de clusters avec des valeurs aléatoires
clusters = X(I(1:16),:);

   //parcours des pixels de l'image
   for i = 1:X_ligne

      //T tableau de la taille des clusters avec comme valeurs, ceux du pixel i
      T=repmat([X(i,1),X(i,2),X(i,3)],K,1);
      dist = zeros(16,1);
      //calcul des distances du pixel i, avec les clusters
                  dist(1:16)=  sqrt((clusters(1:16,1)-T(1:16,1))^2  +  (clusters(1:16,2)-T(1:16,2))^2  +
(clusters(1:16,3)-T(1:16,3))^2);
      //retourne l'indice de la + petite distance
      [m,indice] = min(dist);
      //remplis le tableau à l'indice i, par la valeur de l'indice du clusters le + proche
      Y(i)=indice;
   end

   //parcours des clusters
   for j =1:16
      //calcul des moyennes pour les images en rapport avec le cluster définit dans le tableau Y
      clusters(j,1) = mean(X(find(Y==j),1),'r');
      clusters(j,2) = mean(X(find(Y==j),2),'r');
      clusters(j,3) = mean(X(find(Y==j),3),'r');
   end

//compression pour récuperer la nouvelle image
X_comp = zeros(X_ligne,3);
X_comp(1:X_ligne,:) = clusters(Y(1:X_ligne),:);
X_compressed = matrix(X_comp, img_size(1),img_size(2),3);
//affichage de l'image
imshow(X_compressed);
```