Clustering with the k-Means algorithm

Laboratory 5, DEDP

Table of contents

1 Objective **1**

2 Theoretical aspects **1**
2.1 The k-Means algorithm 1
2.2 Video explanations online 2
2.3 Notes 2

3 k-Means algorithm in Matlab **2**

4 Exercises **3**
4.1 Exercise 1 - color simplifcation 3
4.2 Exercise 2 - replace greenscreen 3
4.3 Vector quantization 4
4.4 Other idea 4

5 Final questions **4**

1 Objective

Implement and use the k-Means algorithm for color-based segmentation of images.

2 Theoretical aspects

2.1 The k-Means algorithm

k-Means is an algorithm for data **clustering**, i.e. identifying groups of close vectors in data.

Algorithm definition

- Input:
  - unlabelled training set of vectors $\vec{x}_1 ... \vec{x}_N$
  - number of classes C
- Initialization: randomly initialize the C centroids

$$\vec{c}_i \leftarrow \text{ random values}$$

- Repeat
  1. Classification: assign each data $\vec{x}$ to the nearest centroid $\vec{c}_i$:

$$l_n = \arg\min_i d(\vec{x}, \vec{c}_i), \forall \vec{x}$$

  2. Update: update each centroids $\vec{c}_i$ = average of the $\vec{x}$ assigned to $\vec{c}_i$

$$\vec{c}_i \leftarrow \text{ average of } \vec{x}, \forall \vec{x} \text{ in class } i$$

- Output: return the centroids $\vec{c}_i$, the labels l_i of the input data $\vec{x}_i$

2.2 Video explanations online

Video explanations of the k-Means algorithm:

- Watch this, starting from time 6:28 to 7:08

  https: // www.youtube.com/watch?v=4b5d3muPQmA

- Watch this, starting from time 3:05 to end

  https: // www.youtube.com/watch?v=IuRb3y8qKX4

2.3 Notes

- It is not guaranteed that k-Means identifies good clusters
  - results depend on the random initialization of centroids
  - repeat many times, choose best result
  - smart initializations are possible (*k-Means++*)

3 k-Means algorithm in Matlab

The algorithm is implemented in Matlab with the function `kmeans()`

```
[idx, C] = kmeans(X,k);
```

The data should be in matrix X, on rows (each row = one data point).

4 Exercises

4.1 Exercise 1 - color simplifcation

1. Load the color image 'Peppers.tiff' using `imread()`. Convert the image to `double` and display it (don't convert to grayscale, leave the colors).
2. Use Matlab's k-Means algorithm to cluster all the pixel values (each pixel = a group of three values R, G, B) into 4 groups.
   - Use the `reshape()` function to resize a $M \times N \times 3$ tensor `I` into a $(M * N) \times 3$ matrix `P`, as follows:

     ```
     P = reshape(I, [], 3);
     ```

   - Use the `kmeans()` Matlab function to do the clustering. Read the documentation for more details.
3. Replace each pixel of the image with the *centroid* of its class. Display the image. How does it look?
4. Change the number of clusters from 2 to 13 and display them in single window with `subplot()`.

4.2 Exercise 2 - replace greenscreen

1. Load the image `Greenscreen.jpg`, and use k-Means to cluster colors into 4 groups.
2. Locate the largest cluster (which one hase more pixels assigned to it). This is probably the green background.
3. Load `cornfield.bmp` and resize image to the same size as `Greenscreen.jpg`.

   Use Matlab function `imresize()` for this:

   `B = imresize(A,[numrows numcols])`)
4. Construct a new image with same size as Greenscreen, but replace all the pixels belonging to the background group with the pixels from Cornfield.

   Display the resulting image.

4.3 Vector quantization

1. Repeat exercise 1, but cluster now a group of pixels:
    - Convert each 2×2 block of pixels into a single vector with 12 values.
    - Perform clustering on these 12-values data
    - Replace each group of 2×2 pixels with each centroid and plot the result.

4.4 Other idea

- make background of `flower.bmp` image lighter/darker/different color

5 Final questions

1. Suppose we do exercises 1 - 3 on a grayscale image. How will it look?