HW6 - SI221:
k-means - 24/03/2020

Instructions

- Please submit by group of 2 the source code and a pdf report of your work by March 31st noon. Each file must have as title: *TP_kMeans_Student1_Student2*

- Send your work in a zipped file to: `giovanna.varni@telecom-paris.fr` and `emile.chapuis@telecom-paris.fr`

Objective

This assignement is aimed at coding some parts of the k-means algorithm to perform a color image quantization that reduces the number of distinct colors used in an image, usually with the intention that the new image should be visually similar as possible to the original one.

An RGB image can be viewed as three different images (a red image, a green image and a blue scale image) stacked on top of each other. In Python, an RGB image of width w and height h consists of an array of $w * h$ pixels, and each pixel is represented by 3 colour pixels. Each color pixel is coded on one byte (256 possible values) and therefore a pixel can take about 16 million values (256 x 256 x 256 colors).

For many applications having such a resolution is not required and we may want to reduce this number to $k \ll 16$ million to save storage space.

About k-means

History

K-means has a rich and diverse history as it was independently discovered in different scientific fields. The term *k-means* was first used by James MacQueen in 1967, though the idea goes back to Hugo Steinhaus in 1956. The standard algorithm was first proposed by Stuart Lloyd of Bell Labs in 1957 as a technique for pulse-code modulation, though it wasn't published as a journal article until 1982. In 1965, Edward W. Forgy published essentially the same method, which is why it is sometimes referred to as Lloyd-Forgy. Even though k-means was first proposed over 50 years ago, it is still one of the most widely used algorithms for clustering. Simplicity and empirical efficiency are the main reasons for its popularity.

K-means

Given a set of samples $\mathbf{x_1} \dots \mathbf{x_m}$ in $\mathbb{R}^n$, the algorithm starts by designating K class centers $\mu_1, \dots \mu_K$ in $\mathbb{R}^n$ (which could be taken among the samples). The next two steps are then carried out iteratively:

- Partition the samples into K classes $\mathcal{C}_1 \dots \mathcal{C}_K$ such that the i^{th} sample belongs to class $\mathcal{C}_j$ if and only if $j = \text{argmin}_{k \in [K]} \|\mathbf{x_i} - \mu_\mathbf{k}\|_\mathbf{2}^\mathbf{2}$
- Update class center μ_k of $\mathcal{C}_k$ to $\frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \mathbf{x_i}$, for all k.

The algorithm stops according to a *stopping criterion* set by the user: a limit on the number of iterations, or the algorithm has converged, i.e.,

$$\text{cost} = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\mathbf{x_i} - \mu_\mathbf{k}\|_\mathbf{2}^\mathbf{2}$$

remains the same between two iterations, or the difference of $\sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\mathbf{x_i} - \mu_\mathbf{k}\|_\mathbf{2}^\mathbf{2}$ between two successive iterations is below a given threshold.

Exercise 1

Generate a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^{300}$ as follows. The label variable y_i takes value uniformly over $\{0, 1, 2\}$. Given y_i, the domain variable $\mathbf{x}_i \in \mathbb{R}^2$ is generated according to an independent bivariate Gaussian distribution $\mathbf{x}_i \sim \mathcal{N}(\mu, \sigma^2 \mathbf{I})$ with mean vector $\mu \in \mathbb{R}^2$ given by:

$$\begin{cases} \mu = (-1, 1) \text{ if } y_i = 0 \\ \mu = (1, 1) \text{ if } y_i = 1 \\ \mu = (0, 1) \text{ if } y_i = 2. \end{cases}$$

And $\sigma^2 = 0.05$.

1. Initiate the centroids by picking up one point at random per Gaussian. Run k-means over the data. Provide a different color for each cluster and plot the means of each clusters.
2. Plot the cost(t) as a function of the number of iterations t. Repeat 50 times and consider for each of these experiments the number of iterations T (till convergence). Provide a histogram on the empirical distribution of the T's.
3. Repeat 2. with centroids initiated randomly (centroids could all belong to the same Gaussian).
4. Now we will study the case where clusters are no longer clearly defined. For each value of $\sigma \in \{0.1, 0.2, 0.3\}$ repeat 3.

5. Show that when minimizing the cost over centroids, the centroids that minimize the cost are indeed the means of their cluster. Hint: consider the cost function restricted to one cluster and find the optimal centroid (i.e., minimizing the cost of that cluster).

6. What happens if we decided to change the L2 norm to the L1 norm. Namely the cost is now

$$\sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} |\mathbf{x}_j - \boldsymbol{\mu}_i|^2.$$

   (a) Repeat 3.

   (b) Consider now a modified version of K-Means, the K-Median algorithm, where centroids are now the median vectors instead of the average. Repeat 3. and compare with (a). What do you observe?

   (c) Through an analogous argument to 5. show that when the L2 distance is replaced by the L1 distance centroids that minimize the cost are the medians of their clusters.

Exercise 2

Download from

`https://www.dropbox.com/sh/r1qu378v4fylh0b/AAA9AVXwKymcqyDcuIlfyAGDa?dl=0` in the folder TP_Kmeans the Python notebook `Kmeans` and open it. Dowload also the image in the data folder. In the first part of the notebook, you will find several functions written to perform color quantization using k-means. Then, you will find some lines to read the image (`landscape.jpg`) you will work on. The goal is to decrease the number of colors to 64. To initialize consider two cases: choose 64 colors randomly from the first 500 pixels of the image, and choose them randomly from the entire image. Consider $\mathbf{x_i} = [\mathbf{R_i}, \mathbf{G_i}, \mathbf{B_i}]$.

Implementing k-means for color quantization

- What is the size (width and height) of the image `landscape.jpg` ?

- What plays the role of m and n ?

- What does the instruction `image_array = np.reshape(image, (w * h, d))` do?

- Complete the implementation of the following functions: `calculate_centroids`, `convergence` and `KMeans`.

- Choose 5 properly spaced out values of the parameters threshold (`threshold`) and of the maximum number of iterations (`max_iterations`). Run the script and provide your comments on how these changes on the parameters' values affect the resulting images.