C M S C 1 7 0

MACHINE PROBLEM NO.2

"IMAGE COMPRESSION USING K-MEANS"

OVERVIEW

As stated in the beginning of the course, you are free to use any high-level programming language you are comfortable with. This includes (but is not limited to) Java, C/C++, and Python. But I'll give *bonus points* if you implement it using Python. The focus of this machine problem is to implement the K-Means algorithm and apply it to image compression. To understand the requirements, be sure to carefully read all the instructions below.

PART ONE: THE K-MEANS ALGORITHM

1. Read **kmdata1.txt**. The file contains a dataset with *300 rows* (representing the examples) and *2 columns* (representing the features). Assume that **K=3** and that you have the three (3) centroids initialized at **(3, 3)**, **(6, 2) and (8, 5).**

2. Run **10 iterations** of K-Means. For each iteration n:
    a. output the centroid assignments to **iter<n>_ca.txt**
    b. output the new centroid locations to **iter<n>_cm.txt**
    c. output J, the value of the distortion function, appending it to **iter<n>_cm.txt**
    d. output dJ, the difference of the current J to the previous J, appending it to **iter<n>_cm.txt**

    Thus, after the 1st iteration, you should have iter1_ca.txt and iter1_cm.txt; for the 2nd iteration, you'll have iter2_ca.txt and iter2_cm.txt, and so on.

Hint: After the 1st iteration, your output files should contain the following correct values (except for J and dJ, which you should calculate on your own):

iter1_ca.txt

```
1
3
2
...
```

iter1_cm.txt

```
2.428301 3.157924
5.813503 2.633656
7.119387 3.616684
J= ___
dJ=___
```

3. After each of the 10 iterations, print
   a. a scatter graph of all data points, colored according to the centroid they belong to
   b. the location of the 3 centroids distinctly marked

   (You don't have to code the graphing part in your code. You are free to use any plotting tool for this part: Octave, Excel, etc.)

4. Write down briefly your observations from the scatter plots and the values of J.

PART TWO: IMAGE COMPRESSION

The exercise above was meant to illustrate how K-Means algorithm worked. In your implementation for the image compression below, the K centroids have to be assigned randomly, and they should initially coincide with K random examples.

1. In this exercise, you will use K-Means for image compression. **kmimg1.png** is a **128x128 image** containing **256** color intensities for **red**, **green** and **blue** (RGB). Extract the RGB values of the image to obtain a 128x128x3 array. This array contains the RGB values at a given x & y location. For example, if the image was loaded into an 1-based array A, A[30,25,2] will contain the color intensity at x=30, y=25 of green (specified by the index 2 in the array).

Resize this array into a 16384 x 3 array since you will not need the x & y information (16384 =128 x 128); you will only work on the RGB values.

2. Apply image compression using K-means.
   a. Randomly choose 16 centroids from the 16384 pixels.
   b. Run 10 iterations of K-means to cluster the pixels into the 16 colors nearest to each one. Generate the compressed image by assigning the intensity value of a pixel to the intensity value of the centroid to which it has been assigned. (You may have to round off to integer values the intensity values of the 16 centroids.) For example, if the 129th pixel (which is at x = 2, y =1, if the array is 1-based) is assigned to centroid 4, which has RGB values of (25, 35, 64), then, the RGB values of the 129th pixel is set to (25, 35, 64).
   c. Print the compressed image alongside the compressed image.

Note: The original image required 24 bits to encode each pixel (8 bits for each of the RGB values). The size of the original file is thus 128x128x24= 393,216 bits. With the image compression done with K-Means, we only need to store the RGB values of the 16 centroids, then assign the centroid value for each of the pixels (instead of assigning the RGB values). Thus, the compressed image can be represented in 16*24+128*128*4= 65,920 for a compression factor of around 6.

GENERAL GUIDELINES

1. This machine problem will be done in groups of three (3). Make sure that your groupmates are in the same section as you. Section C has 33 students, so there should be 11 groups in total. Section D has 32 students, so there must be 10 groups and one brave pair in total. Groups must only be a maximum of three students each. *No exceptions*.
2. Submit a written report explaining your observations of the scatter plots and the values of J in PDF format.
3. In your report, include a page with a description of the contribution of each team member.
4. Submit a file called README.txt that explains clearly how to compile and run your program.
5. Include the original and compressed image in your submission.
6. Email your output (codes + documentation) via a .zip folder to avgerman@up.edu.ph ON OR BEFORE **DECEMBER 6, 2017 WEDNESDAY (11:59 PM)**.
7. Email subject: **[SUBMISSION] CMSC170_MP2**

DISCLAIMER

This machine problem was based on a programming assignment by Andrew Ng and was modified accordingly to Sir Tony Briza's specifications for his CMSC 170 class 1st semester, SY 2014 – 2015.