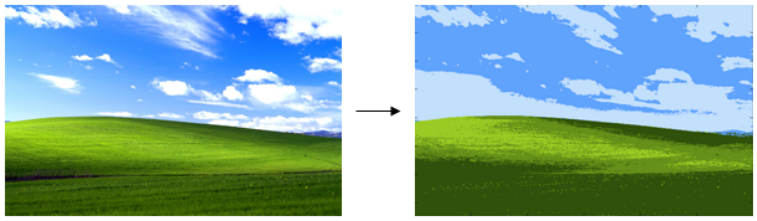
COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS
DEPARTMENT OF APPLIED INFORMATICS

Introduction to Artificial Intelligence
Exercise 10 - k-means

November 27, 2025

10. k-MEANS - IMAGE POSTERIZATION

Using k-means method we will posterize images - replace the whole scale of image colours to only a few, keeping the most of the image appearance.

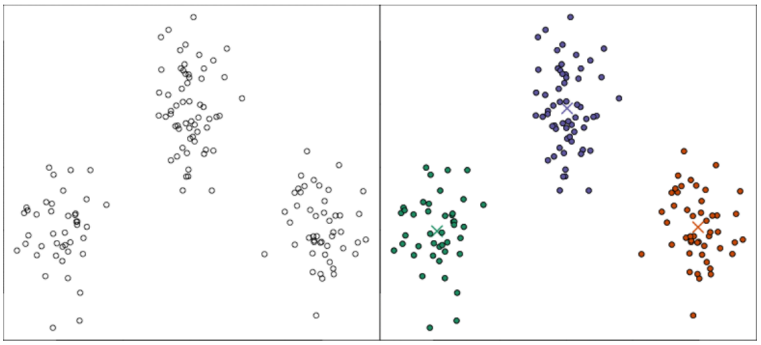


Data:

Input for our program will be an image, i.e. set of pixels, each of which has its own color (RGB). Using these colors, we will compute few (e.g. 5) "main" colors using k-means algorithm. We'll then alter the image to consist only of these 5 "main" colors (posterization). Your task is to implement the middle part of this process, i.e. finding the main colors.

k-means clustering:

The main idea of k-means lies in finding k centers that represent clusters within input data. Example: first we only have raw 2-dimensional data with no additional information (left). The k-means algorithm finds three centers (crosses in right image) that represent clusters from input data.



We will have 3-dimensional data - R, G, B components of color. Therefore, resulting centers will also be 3-dimensional colors, and they will represent those "main" colors mentioned above. We will ignore the arragement of the pixels (their [x,y] coordinates), and we will process the pixels in random order, same way as we treated input datapoints in neural network.

Algorithm:

There are different flavors for k-means algorithm, all of which yield almost identical results.

1) Stochastic method: we will always look only at one point (pixel color), and repeat the following modification of centers for each point during a few iterations:

- take one point x (pixel color)
- find the closes center point to x c^*
  $c^* = \text{argmin}_{i=1..k}(||c_i - x||)$
- "pull" the center c^* a little towards the point x
  $c^* := c^* + \alpha(x - c^*)$

Similarly as when training neural networks, here we also iterate through all points x in random order, and then repeat the whole process several times.

2) Batch method: we work always with all the points and move the centers right into the middles of the clusters:

- split all points $x_1, \ldots, x_n$ into k clusters $C_1, \ldots, C_k$ according to which of the centers c_i is the closest
  $C_i = \{x | i = \text{argmin}_{j=1..k}(||c_j - x||)\}$
- set the center positions $c_1, \ldots, c_k$ to cluster centers $C_1, \ldots, C_k$
  $c_i = mean(x \in C_i)$

Pseudocode:

```
1:  Initialize centers
2:  while number_of_iterations < max_iterations do
3:      ── Stochastic version ──
4:      for each input x in random order do
5:          c* ← find center closest to x
6:          c* ← c* + α.(x − c*)
7:      ── Batch version ──
8:      C ← list of sets (clusters)
9:      for i in 1,...,k do
10:         C[i] ← set of inputs that center c_i is closest to
11:     for i in 1,...,k do
12:         c_i ← mean(C[i])
13:     if no center c_i was moved then
14:         found stable solution, quit
```

Task 1: Into the prepared skeleton of the code, finish the function *k_means(...)* the first way - stochastic method.
Task 2: Use also the batch method to implement the k-means.

To keep the code clear and modular, you can use (and implement) the suggested functions *find_best_center(pixel_color)* and *split_pixels_to_clusters(num_clusters)*.
Prepared in the program outline:

- self.pixels: list of all picture pixels, each pixel is a list [R, G, B] (for example self.pixels[42] == [255, 0, 0])
- self.centers : list of k centers, each center is a list [R, G, B] (for example self.centers[3] == [250, 10, 10])
- distance(a, b): computes the euclidian distance between two points (or colors)