Pattern recognition systems – Lab 6

K-means clustering

1. Objectives

This laboratory session deals with the problem of clustering a set of points. This is a machine learning task that is unsupervised, i.e. the class labels of the points are not known and not required. Successful methods will identify the underlying structure in the data and group similar points together.

2. Theoretical Background

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics [1].

The input for the method is the set of data points: $X = \{x_i, i = 1{:}n\}$. Each point is d-dimensional $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$. The goal of the k-means clustering method is to partition the points into K sets denoted by $S = \{S_k | k = 1{:}K\}$. The mean value of the points in each set is named m_k. The partitioning must be done in such a way as to minimize the following objective function:

$$J(X,S) = \sum_{k=1}^{K} \sum_{x \in S_k} dist(x, m_k)$$

where $dist(.,.)$ is the Euclidean distance function in d-dimensional space:

$$dist(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

This is an NP-hard problem but there are several approximations that provide good results. Lloyd's method proposes to divide the problem into two parts. If we have the partitioning we can calculate the means, but we cannot know the partitions if the cluster centers are unknown. The idea is to start with a random set of cluster centers and iteratively refine these. The algorithm is guaranteed to converge to a local minimum but it is may not be the global minimum [2].

Let L denote the membership function for each point, so $L_i \in 1{:}K, i = 1{:}n$. The membership function returns the cluster of the ith point. Start by assigning the cluster centers to random points from the dataset: $m_k = x_{r_k}$, where r_k is uniformly distributed random integer from *1:n*. In order to ensure better convergence more advanced initialization techniques can be applied. In [3] the authors define the k-means++ method. This relies on drawing the point with a given distribution that disfavors points that are close together.

Afterwards perform several iterations of assignment steps and update steps. When the membership does not change or the maximum number of iterations is reached, the algorithm is halted. The steps of the method are given in the following algorithm.

K-means algorithm

Initialization – Randomly select the *K* centers from the set of input points. Let each r_k be a uniformly distributed random integer from *1:n*, then the initial means are chosen as:

$$m_k = x_{r_k}$$

Assignment - Assign each point from the input dataset to the closest cluster center found so far. The membership function will take the value of the index of the closest center:

$$L_i = argmin_k dist(x_i, m_k)$$

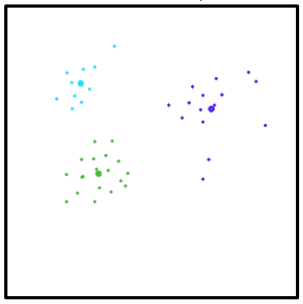
Update – Recalculate the cluster centers based on the membership function. The new cluster centers are the means of the points from the cluster. In the following, summation is performed on all elements that are in cluster k, i.e. they have membership of $L_i = k$.

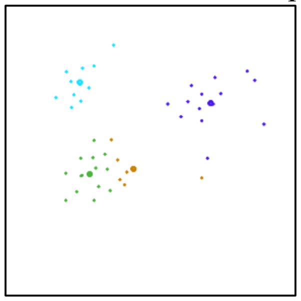
$$m_k = \frac{\sum_{L_i=k} x_i}{\sum_{L_i=k} 1} = \frac{\sum_{x \in S_k} x}{|S_k|}$$

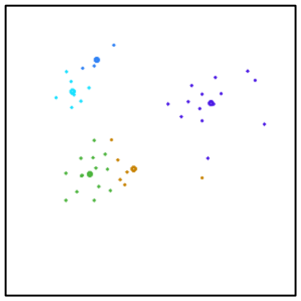
Halting condition - If there is no change in the membership function then the algorithm can be halted because further calculation will lead to no changes in the mean values. A maximum number of iterations can also be enforced. If none of the above conditions are met the algorithm continues with the assignment step.

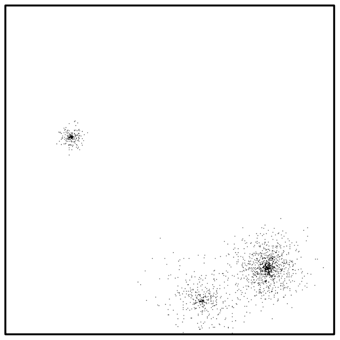
3. Example results

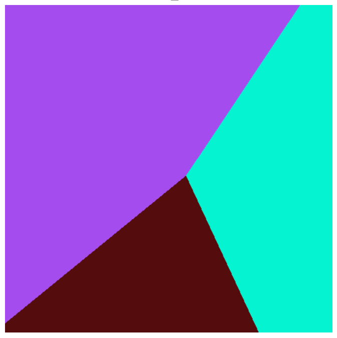
In the case of d=2, when K-means is run on a set of 2D points:


points2 - K = 3

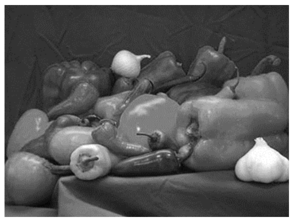

points2 - K = 4

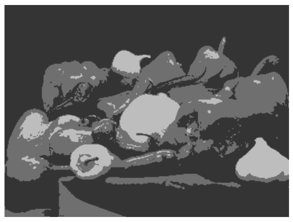

points2 - K = 5

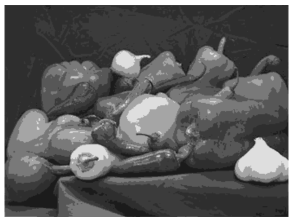

points4

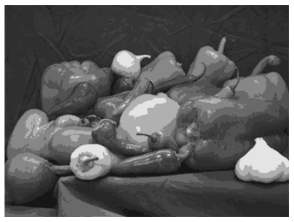

points4 - Voronoi tessalation K=3

In the case of d=1, when K-means is run on a grayscale image:


img01.jpg

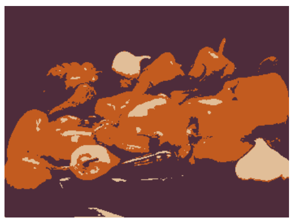

K=3

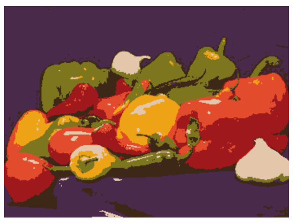

K=7

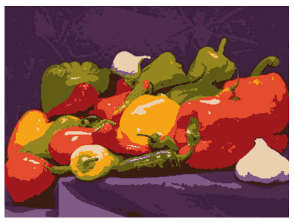

K=10

In the case of d=3, when K-means is run on a color image:

img01.jpg


K=3


K=7


K=10

4. Implementation details

Generating a random integer with uniform distribution between *a* and *b* (inclusive):

```
#include <random>

default_random_engine gen;
uniform_int_distribution<int> dist_img(a,b);
int randint = distribution(gen);
```

Creating a color image:

```
Mat img(height, width, CV_8UC3);
```

Assigning random colors to clusters:

```
const int K = 3;
Vec3b colors[K];
for(int i=0; i<K; i++)
 colors[i] = {dist_img(gen), dist_img(gen), dist_img(gen)};
```

Assigning colors[k] to position (i,j):

```
img.at<Vec3b>(i,j) = colors[k];
```

5. Practical work

1. Implement K-means on general input data (d dimensional points). Stop the algorithm once no change in the membership is observed or after a certain number of maximum iterations. The number of clusters, K, is given by the user.
2. Apply K-means on a set of 2D points (input files points*.bmp) – in this case d=2
    a. Choose random colors to visualize the clusters based on the resulting membership function.
    b. Color the neighborhood of points for better visualization
    c. Draw the Voronoi tessellation corresponding to the obtained cluster centers. For this picture you must color each image position (including the background) according to which is the closest center from it.
3. Apply K-means on a grayscale image. Use the intensity as the single feature for the input points – in this case d=1
    a. Recolor the input image based on the mean intensity of each cluster.
4. Apply K-mean on a color image. Use the color components as the features for the input points – in this case d=3
    a. Recolor the input image based on the mean color of each cluster.
5. Optionally, implement the k-means++ initialization technique from [3]. (*)

References

[1] Cluster analysis Wikipedia article - https://en.wikipedia.org/wiki/Cluster_analysis
[2] K-means Wikipedia article - https://en.wikipedia.org/wiki/K-means_clustering
[3] Arthur, David, and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding." Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007.
[4] P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. „Contour Detection and Hierarchical Image Segmentation”, IEEE TPAMI, Vol. 33, No. 5, pp. 898-916, May 2011.
[5] Image segmentation dataset:
https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html