1 Clustering with K-means

Consider the two datasets Fig.1(a,b) and Fig.1(c,d) below. We want to cluster these datasets using K-means (L-2 norm, $K = 2$), with different initial conditions. Starting from the given initial means (black and white circles), approximate by hand a few iterations of K-means and draw the resulting separation boundary between the two clusters. Is the K-means algorithm able to separate the two clusters well on both datasets? What is the effect of the initialization of the means on each dataset? Explain.

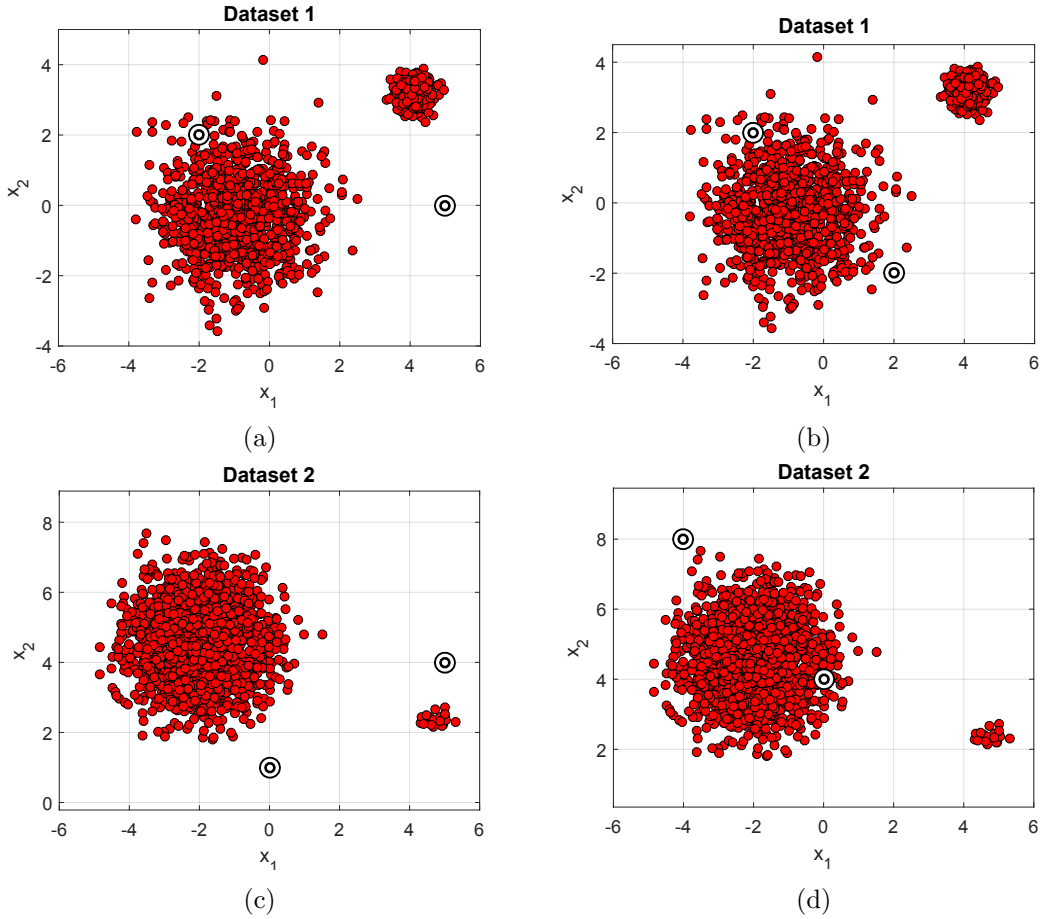


Figure 1: Datasets with different initial conditions. The black and white circles correspond to the initial means for K-means.

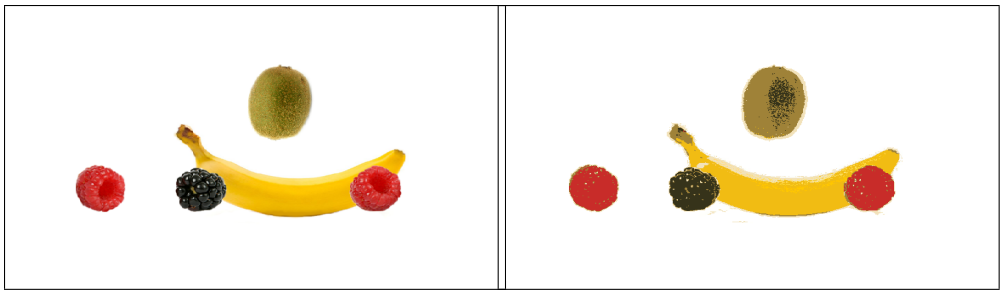
2 K-means

A) An artist wants to turn a photo of a flower (see Figure 2) into a painting. He can only fit 6 colors in his palette so it would be much easier to only use 6 colors for the painting. Could you use K-means to tell him which colors to use and where to use them ? How would you do this ? Note: you can represent the image in a $M \times 3$ matrix where M is the number of pixels and 3 are the necessary components to represent the color (RGB).

(a) (b)

Figure 2: a) The photo of the flower. b) Just in case you didn't know what a palette is.

B) You have to write a piece of software that will assign a different label to each fruit present in Figure 3a. You know the image has five different fruits and that the background is white. You run the K-Means on all the M pixels, that don't have a white color, and you represent each datapoint as a vector with three dimensions (the RGB values), $\mathbf{x}^i \in \mathbb{R}^3$. You ran K-Means with $K = 5$ and obtained the results in Figure 3b. The K-Means assigned the two raspberries to the same cluster because they have similar color. What information should you add to the K-Means algorithm so that he could separate all the fruits ? Should you use the L-2 norm in this case ?



(a) (b)

Figure 3: a) Photo with five different fruits in a white background. b) Result of K-Means.

3 Semisupervised clustering F1-measure

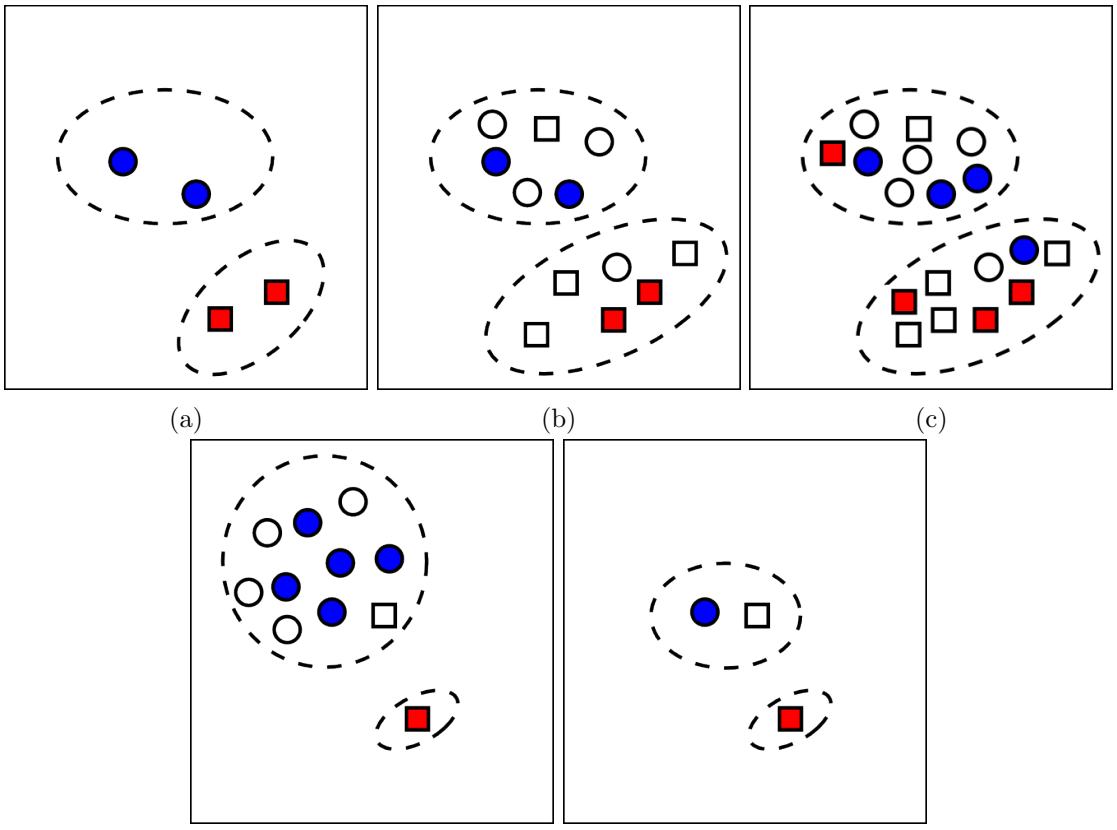


Figure 4: The figures above show a two-class dataset. The classes are shown by the shapes. This dataset was subject to a clustering algorithm whose result is shown using the dotted ellipses. In each case, the algorithm returned two clusters as shown

A) Compute the semi-supervised clustering F1-Measure in each case. The labeled data points for computing the F1-Measure are shown in color. To recall, the semi-supervised clustering F1-Measure is given by :

$$
\begin{aligned}
F_1(C,K) &= \sum_{c_i \in C} \left(\frac{|c_i|}{M} \max_k \{F_1(c_i,k)\} \right) \\
F_1(c_i,k) &= \frac{2R(c_i,k)P(c_i,k)}{R(c_i,k)+P(c_i,k)} \\
R(c_i,k) &= \frac{n_{ik}}{|c_i|} \\
P(c_i,k) &= \frac{n_{ik}}{|k|}
\end{aligned}
\tag{1}
$$

Where M is the total number of **labeled** datapoints, k indexes the cluster number, $C = \{c_i\}$ is the set of classes, n_{ik} is the number of datapoints of class c_i in cluster k, $|c_i|$ is the number of datapoints in class c_i and $|k|$ is the number of datapoints in cluster k.

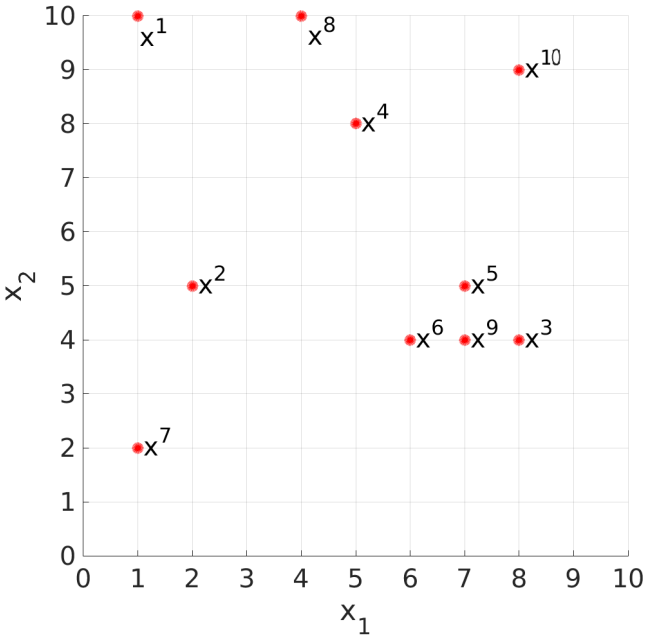
B) Compare the results of Fig.4a with Fig.4b and discuss the effect of the **percentage of labeled data**. Next, compare the results of Fig.4b and Fig.4c and explain the effect of **mis-clustered labels** on F1-measure. Compare the results of Fig.4d with Fig.4e and discuss the effect of **unbalanced classes**.

4 DBSCAN

You are given the following points :

Coordinates / Points	x_1	x_2
x^1	1	10
x^2	2	5
x^3	8	4
x^4	5	8
x^5	7	5
x^6	6	4
x^7	1	2
x^8	4	10
x^9	7	4
x^{10}	8	9

Data points



Visualisation of the points

The distance matrix based on the L-2 norm is the following :

	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}
x^1	0	5.1	9.2	4.5	7.8	7.8	8	3	8.5	7.1
x^2		0	6.1	4.2	5	4.1	3.2	5.4	5.1	7.2
x^3			0	5	1.4	2	7.3	7.2	1	5
x^4				0	3.6	4.1	7.2	2.2	4.5	3.2
x^5					0	1.4	6.7	5.8	1	4.1
x^6						0	5.4	6.3	1	5.4
x^7							0	8.5	6.3	9.9
x^8								0	6.7	4.1
x^9									0	5.1
x^{10}										0

Table 2: Distance Matrix

Recall the pseudocode for DBSCAN described in Algorithm 1.

Algorithm 1 DBSCAN

```
procedure DBSCAN(X, ϵ, MinPoint)
    K = 0
    Cores = {}
    ∀ x in X, Labels(x) = Noise
    for each point x in X do
        Neighbors N_x = GetNeighbors(X, x, ϵ)
    if |N_x| >= MinPoints − 1 then
        Cores = Cores ∪ {x}
    for each c ∈ Cores do
        if Labels(c) = Noise then
            K = K + 1;
            Labels(c) = K
            Neighbors N_c = GetNeighbors(X, c, ϵ)
            ReachableSet = N_c
            while ReachableSet not empty do
                Get and remove last element r from ReachableSet
                if Label(r) = Noise then
                    Labels(r) = K
                    if r ∈ Cores then
                        Neighbors N_r = GetNeighbors(X, r, ϵ)
                        ReachableSet = ReachableSet ∪ N_r
    return K, Cores, Labels
```

In this algorithm, the *Cores* have *(MinPoints - 1)* samples[1] in their neighborhood and are used to define the boundaries of the clusters (circles) for testing. A core's neighborhood is merged with another (i.e. we "extend" a given cluster) only if the two cores are neighbors.

A) For each pair of parameters given after, find the number of clusters given by the DBSCAN algorithm and draw them:

1. $\epsilon = 2.5$ and $MinPoints = 2$
2. $\epsilon = 3.5$ and $MinPoints = 2$
3. $\epsilon = 3.5$ and $MinPoints = 4$

[1]The *MinPoints* parameter correspond to the minimum number of samples in the ϵ-neighborhood of a point for it to be considered as a core. This includes the point itself

5 Computational Cost (To be done at home)

The performance of a machine learning technique must often be evaluated in terms of its computational costs. The more computational steps are required the more unlikely it is that the algorithm could be ported for real-time computation on small portable hardware (robots, cell phones, etc). Computational costs are also tightly linked to the curse of dimensionality. The larger the dimension of the dataset, the heavier the computational costs. Knowing whether computational costs grow linearly or exponentially with the number of datapoints, M, and the dimension of the dataset, N, is hence crucial. One will prefer a method that grows only linearly with M and N.

A) Compute the computational cost per iteration of K-means and Soft-K-means.

B) Compute the computational cost of DBSCAN.
How do you think you can reduce the complexity (Think about a way to avoid finding the distance to all the other points for each point) ?

C) Discuss the pros and cons of these clustering techniques given your answer to previous questions.