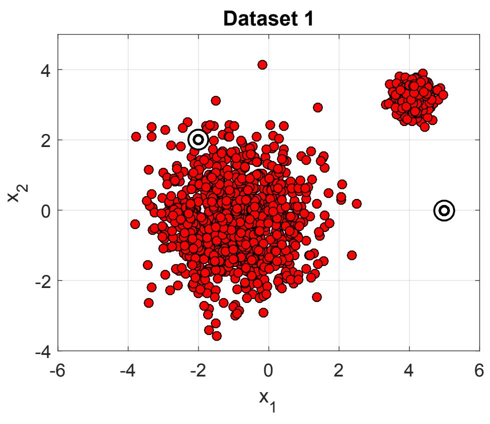
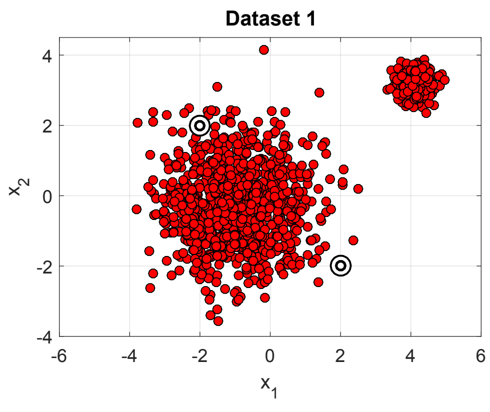
1 Clustering with K-means

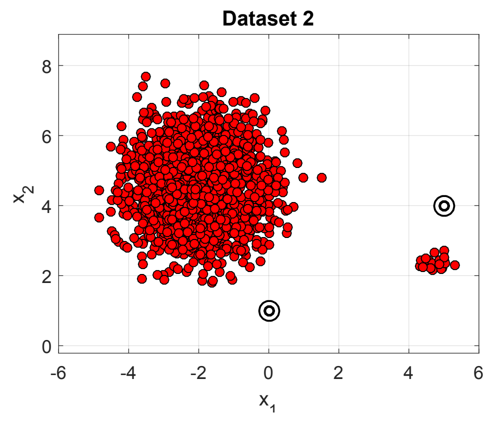
Consider the two datasets Fig.1(a,b) and Fig.1(c,d) below. We want to cluster these datasets using K-means (L-2 norm, $K = 2$), with different initial conditions. Starting from the given initial means (black and white circles), approximate by hand a few iterations of K-means and draw the resulting separation 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.



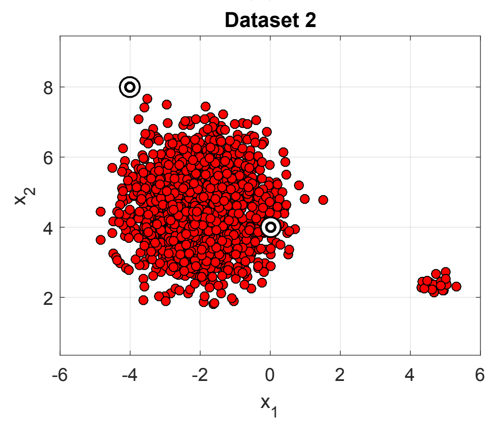
(a)



(b)



(c)



(d)

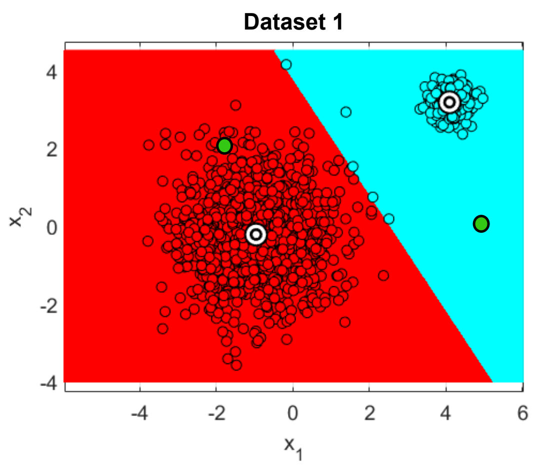
Figure 1: Datasets with different initial conditions. The black and white circles correspond to the initial means for K-means.

Solutions

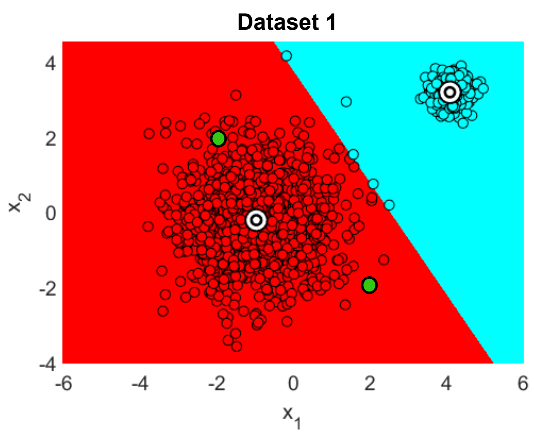
Dataset 1

K-Means with L2-Norm results in a clustering where each data point is assigned to its nearest cluster centroid using the Euclidean L2-Norm as the distance metric for proximity. Hence, all the points that are closer to a cluster centroid are assigned to that cluster. Once all points have been assigned to a cluster, new cluster centroids are computed. This procedure is repeated until the clusters are stable.

In order to obtain the separating boundary, one must compute the perpendicular bisector of the line joining the two cluster centers. The result of K-means applied on the first dataset is shown in Figure 2.



(a)

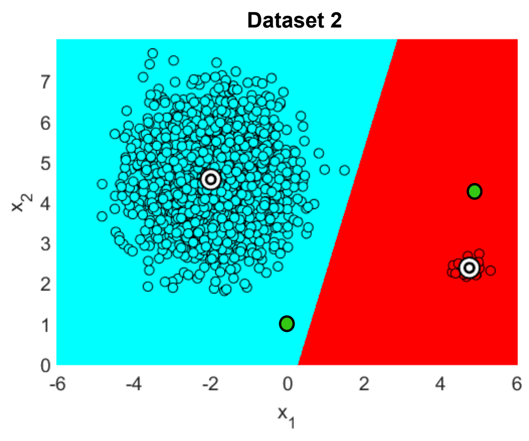


(b)

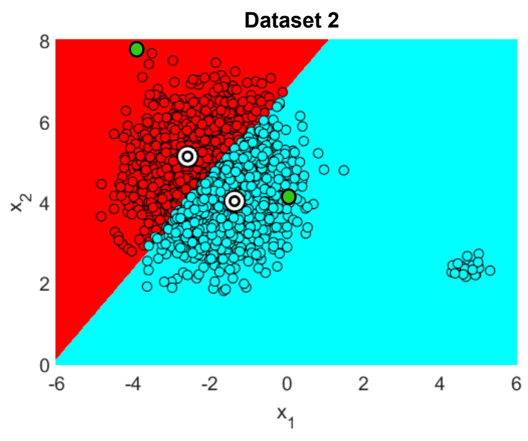
Figure 2: Results of K-means (L2-Norm, K=2) for the first dataset with two different initial conditions. The initial means are shown as green circles and the final ones as black and white circles.

We can see that even though we start with two very different initial centroid locations, the first ones placed very close to each cluster (Fig.2a) and the second ones close to only one cluster (Fig.2b), the final clustering result is exactly the same. This is due to the fact that the data is sufficiently balanced between each cluster. Hence, in the second case (Fig.2b), one of them stays with the left cluster, while the upper right cluster pulls the other centroid towards itself.

Dataset 2



(a)



(b)

Figure 3: Results of K-means (L2-Norm, K=2) for the second dataset with two different initial conditions. The initial means are shown as green circles and the final ones as black and white circles.

The results of K-means with K=2 and the L2-Norm applied on the second dataset are shown on Fig.3. For the first initial condition (Fig.3a), the algorithm clustered the data as expected. However, in the second case (Fig.3b), rather than separating the clusters, K-means splits the left cluster in two. In this case, the data is not well balanced between the clusters. The upper right cluster is not able to pull the other centroid towards itself due to its low weight compared to the big cluster. In the case of an unbalanced dataset, a good choice of initial conditions is thus particularly important.

2 K-means

A) An artist wants to turn a photo of a flower (see Figure 4) 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 ? <u>Note</u>: 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).

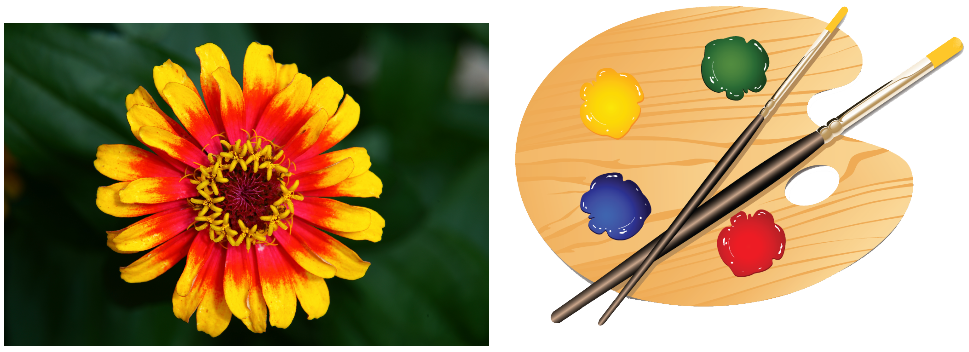

(a) (b)

Figure 4: 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 5a. 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 5b. 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 ?

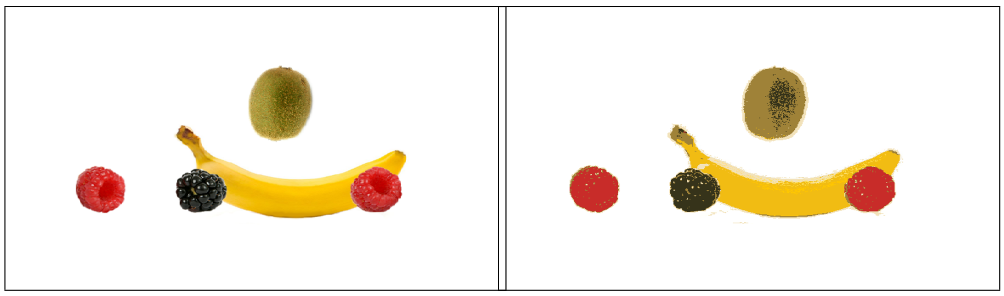

(a) (b)

Figure 5: a) Photo with five different fruits in a white background. b) Result of K-Means.

Solutions

A) We need to assign each pixel to one of the six different colors. For the image to be recognizable, pixels with similar color should be represented by the same color. This can be achieved by running K-Means (with L-2 norm) on the photo's pixels represented in the RGB space. This means that we will have M datapoints and each datapoins is a vector with dimension three $\mathbf{x}^i \in \mathbb{R}^3$. We set K=6 and we find 6 clusters where each datapoint is assigned to the nearest cluster. The colors to be used by the painter are the means of each cluster. In order to tell the painter where to use those colors, we assign each pixel to the nearest cluster center, $k_i = \text{argmin}_k\, d(\mathbf{x}^i, \mathbf{u}^k)$ where $d(\mathbf{x}^i, \mathbf{u}^k)$ corresponds to the L-2 norm. We can then paint each pixel with its cluster color obtaining the result shown in figure 6.

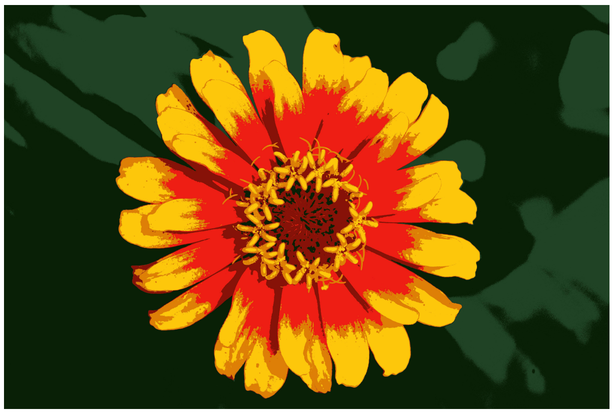
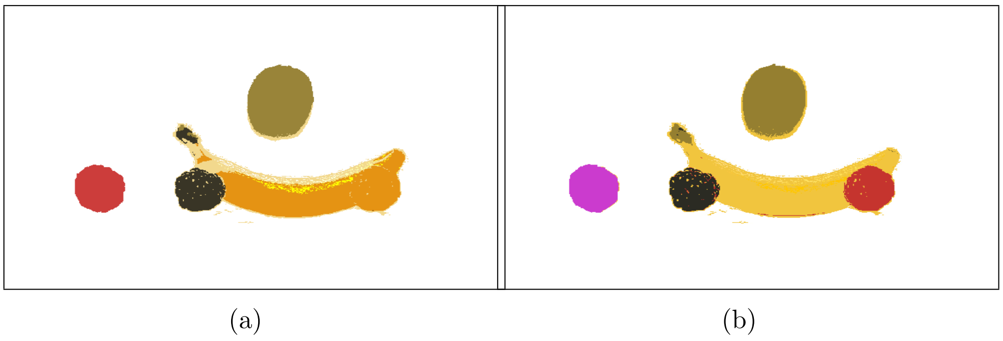


Figure 6: Photo of the flower after replacing each pixel color by its cluster's color. This image was created using only 6 different colors.

B) The two raspberries have similar colors but different locations, therefore we could add the pixels positions to our data to help the clustering assignment process. We now have M datapoints with five dimensions each, $\mathbf{x}^i \in \mathbb{R}^5$, where M is the number of pixels, and five correspond to the number of dimensions, three for color and two for position. We need to be careful here when using the K-Means. The dimensions which correspond to the color information can have values from 0 to 255 but the position values can go from 0 to the maximum image size (which might be much higher than 255). If we run the K-Means L-2 norm, in these conditions, pixels' positions will have an higher responsibility for the assignment of clusters than the their color (see Figure 7a). If we want the position to contribute as much as the color for the result of the K-Means, some variables should have more weight for the cluster assignment than others. Therefore, instead of using K-Means L-2 norm we could use a weighted norm $d(\mathbf{x}^i, \mathbf{u}^k) = w_1 \sum_{j=1}^{3} d(x_j^i, \mu_j^k) + w_2 \sum_{j=4}^{5} d(x_j^i, \mu_j^k)$, where w_1 is the weight associated with the color variables and w_2 is the weight associated with the position variables (see Figure 7b for the results). Each pixel is assigned to the closest cluster center, $k_i = \text{argmin}_k \, d(\mathbf{x}^i, \mathbf{u}^k)$. The weights can for example be the inverse of the standard deviations of the color variables and the position ones and so be seen as a normalization factor.
Note that the same kind of solution can be obtained by normalizing directly the data according to each dimension. This can be done by removing the mean of each dimension and dividing by the standard deviation for example. The concept of normalizing data when the scales of some dimensions are different has already been discussed in the exercises about PCA and is still of importance here.



(a) (b)

Figure 7: a) Result of K-Means without variable normalization and using the L-2 norm. We can see that the right part of the banana is assigned to the same cluster as the right raspberry. b) Result of K-Means with a weighted norm, the misclassifications in this case less significant. We changed the color of one cluster to see that it is a different one.

3 Semisupervised clustering F1-measure

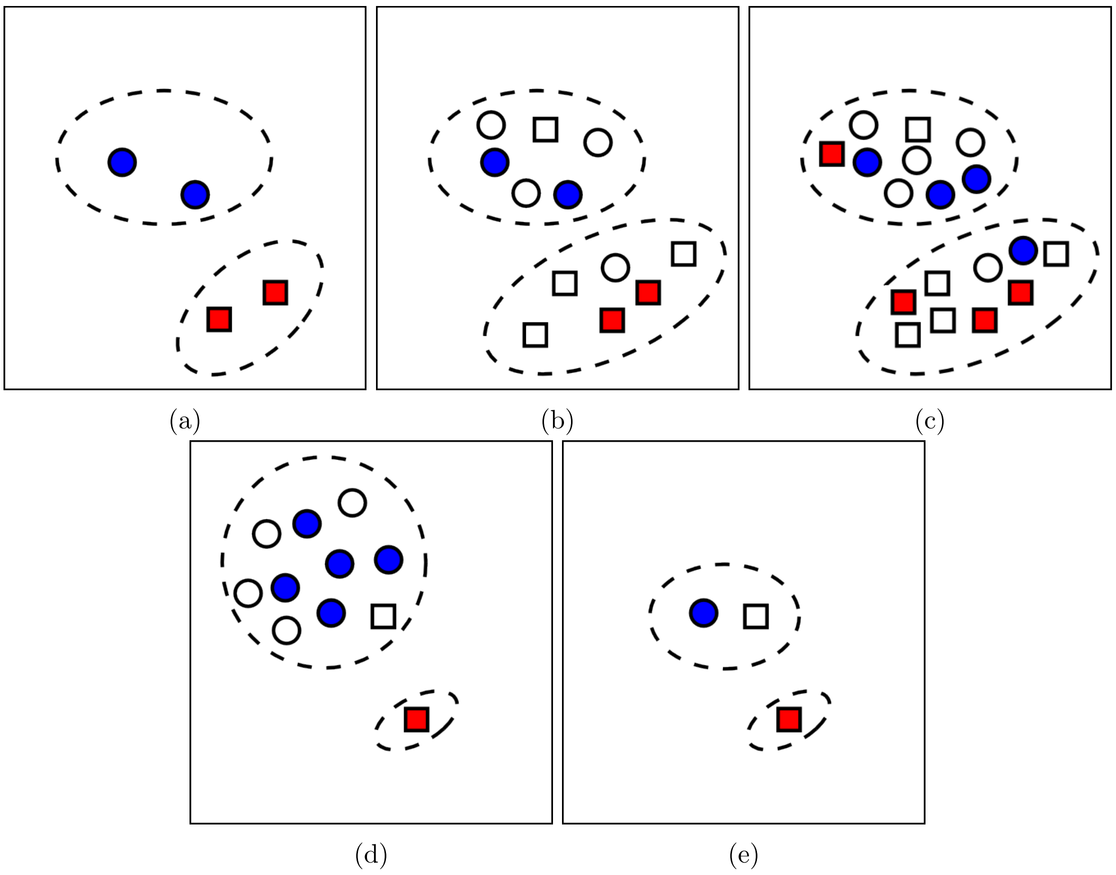


Figure 8: The figures above show a two-class dataset. The classes are shown by the shapes. This dataset was subject to a clustering algorithm whose result is shown using the dotted ellipses. In each case, the algorithm returned two clusters as shown

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.8a with Fig.8b and discuss the effect of the **percentage of labeled data**. Next, compare the results of Fig.8b and Fig.8c and explain the effect of **mis-clustered labels** on F1-measure. Compare the results of Fig.8d with Fig.8e and discuss the effect of **unbalanced classes**.

Solutions

A) The first equation is essentially a weighted sum of each class' individual F1-Measure. However, to compute each of the terms, we must first assign each class to a particular cluster. This is done in the $\max_k \{F_1(c_i, k)\}$ expression which is assigning that cluster to a class for which the F1-Measure is maximum. Hence we have two nested loops, one over all the classes and the other over all the clusters for picking the best cluster.
The numbering of clusters is arbitrary. Here we choose the top cluster as Cluster 1 and the bottom cluster as Cluster 2.

Note that if precision (P) and recall (R) are both zero, the F1-Measure is also zero. We can show this as follows

$$F = \frac{2 * P * R}{P + R} = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

When $P \to 0$ and $R \to 0$, $\frac{1}{P} \to \infty$ and $\frac{1}{R} \to \infty$; i.e. the denominator of F goes to infinity. So $F \to 0$

Class c_i	$\frac{n_{c_i}}{M}$	Cluster k	P	R	F1(c_i, k)	$\max_k$ F1(c_i, k)	F1
Circle	$\frac{2}{4}$	1	$\frac{2}{2}$	$\frac{2}{2}$	1	1	$\frac{1}{2} \times 1$
		2	$\frac{0}{2}$	$\frac{0}{2}$	0		$+$
Square	$\frac{2}{4}$	1	$\frac{0}{2}$	$\frac{0}{2}$	0	1	$\frac{1}{2} \times 1$
		2	$\frac{2}{2}$	$\frac{2}{2}$	1		$=$ 1.0

Table 1: Results (a)

Class c_i	$\frac{n_{c_i}}{M}$	Cluster k	P	R	F1(c_i, k)	$\max_k$ F1(c_i, k)	F1
Circle	$\frac{2}{4}$	1	$\frac{2}{6}$	$\frac{2}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2} \times \frac{1}{2}$
		2	$\frac{0}{6}$	$\frac{0}{2}$	0		$+$
Square	$\frac{2}{4}$	1	$\frac{0}{6}$	$\frac{0}{2}$	0	$\frac{1}{2}$	$\frac{1}{2} \times \frac{1}{2}$
		2	$\frac{2}{6}$	$\frac{2}{2}$	$\frac{1}{2}$		$=$ 0.5

Table 2: Results (b)

Class c_i	$\frac{n_{c_i}}{M}$	Cluster k	P	R	F1(c_i,k)	$\max_k$ F1(c_i,k)	F1
Circle	$\frac{4}{8}$	1	$\frac{3}{9}$	$\frac{3}{4}$	$\frac{6}{13}$	$\frac{6}{13}$	$\frac{1}{2}\times\frac{6}{13} + \frac{1}{2}\times\frac{6}{13} = 0.46$
		2	$\frac{1}{9}$	$\frac{1}{4}$	$\frac{2}{13}$		
Square	$\frac{4}{8}$	1	$\frac{1}{9}$	$\frac{1}{4}$	$\frac{2}{13}$	$\frac{6}{13}$	
		2	$\frac{3}{9}$	$\frac{3}{4}$	$\frac{6}{13}$		

Table 3: Results (c)

Class c_i	$\frac{n_{c_i}}{M}$	Cluster k	P	R	F1(c_i,k)	$\max_k$ F1(c_i,k)	F1
Circle	$\frac{5}{6}$	1	$\frac{5}{10}$	$\frac{5}{5}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{5}{6}\times\frac{2}{3} + \frac{1}{6}\times 1 = 0.72$
		2	$\frac{0}{1}$	$\frac{0}{5}$	0		
Square	$\frac{1}{6}$	1	$\frac{0}{10}$	$\frac{0}{1}$	0	1	
		2	$\frac{1}{1}$	$\frac{1}{1}$	1		

Table 4: Results (d)

Class c_i	$\frac{n_{c_i}}{M}$	Cluster k	P	R	F1(c_i,k)	$\max_k$ F1(c_i,k)	F1
Circle	$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{1}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{1}{2}\times\frac{2}{3} + \frac{1}{2}\times 1 = 0.83$
		2	$\frac{0}{1}$	$\frac{0}{1}$	0		
Square	$\frac{1}{2}$	1	$\frac{0}{2}$	$\frac{0}{1}$	0	1	
		2	$\frac{1}{1}$	$\frac{1}{1}$	1		

Table 5: Results (e)

B)

- **Comparing results of (a) and (b)** : The F1-Measure in (a) is a perfect 1.0. The F1-Measure in (b) is a lower value (0.5) than in (a) despite the same labels appearing in the clusters as in (a). ***This is because there is more data in the clusters now for which we do not have the class information. This uncertainty is penalized by the F1-Measure. The F1-measure is 1.0 only when all data-points have the same class label and are all labeled. In this case, we are no longer performing***

semi-supervised clustering but classification (fully supervised clustering).

- **Comparing results of (b) and (c)** : The F1-Measure in (c) is lower than that of (b). This is due to misclustered labels in both classes. ***Notice that there are also misclustered data in (b) but they are not labeled. Thanks to the labeling in (c), F1-measure is aware of misclustered data and it has penalized the clustering accordingly.***

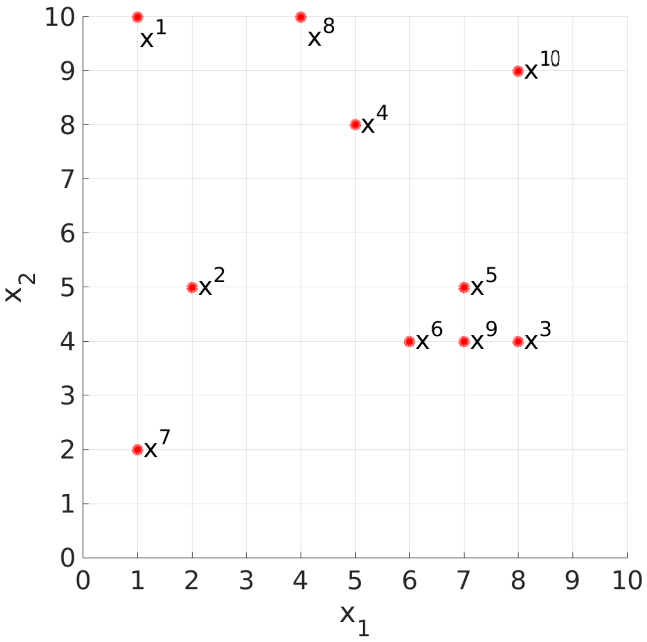
- **Comparing results of (d) and (e)** : we see that the F1-Measure in (e) is higher than that of (d) although they both have only one misclassified datapoint (1 unlabeled square wrongly clustered with circles). This is due to the fact that classes are unbalanced in the number of labeled data-points (five times more labeled data-points in Class 1 than in Class 2). The F1-Measure corresponding to the circle class (Class 1) gives more weight to its associated F1-measure than to that of Class 2. This is, however, slightly counterbalanced by the fact that Cluster 1 contains more points than Cluster 2 and hence has more associated uncertainty. ***Hence, it important to provide balanced labeling for the classes to avoid influencing the F1-Measure to favor the class with most labels***.

4 DBSCAN

You are given the following points :

Points \ Coordinates	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 7: Distance Matrix

The pseudocode for DBSCAN described in Algorithm 1.

Algorithm 1 DBSCAN

```
procedure DBSCAN(X, $\epsilon$, $MinPoint$)
    K = 0
    Cores = {}
    $\forall\ x$ in $X$, Labels($x$) = Noise
    for each point $x$ in $X$ do
        Neighbors $N_x$ = GetNeighbors($X,\ x,\ \epsilon$)
    if $|N_x|$ >= MinPoints $- 1$ then
        Cores = Cores $\cup \{x\}$
    for each $c \in$ Cores do
        if Labels($c$) = Noise then
            $K = K + 1$;
            Labels($c$) = $K$
            Neighbors $N_c$ = GetNeighbors($X,\ c,\ \epsilon$)
            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 $r \in$ Cores then
                        Neighbors $N_r$ = GetNeighbors($X,\ r,\ \epsilon$)
                        ReachableSet = ReachableSet $\cup N_r$
    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 $MinPoints = 4$

Solutions

A)

1. First we need to look for the ϵ-neighborhood of each point. Let's call $N(x^i)$ the ϵ-neighborhood of the point x^i.

   We have :

   $N(x^1) = \{\}; N(x^2) = \{\}; N(x^3) = \{x^5, x^6, x^9\}; N(x^4) = \{x^8\}; N(x^5) = \{x^3, x^6, x^9\}; N(x^6) = \{x^3, x^5, x^9\}; N(x^7) = \{\}; N(x^8) = \{x^4\}; N(x^9) = \{x^3, x^5, x^6\}; N(x^{10}) = \{\}$

   Since $MinPoints = 2$, x^1, x^2, x^7 and x^{10} are considered outliers and we get two clusters $C_1 = \{x^3, x^5, x^6, x^9\}$ and $C_2 = \{x^4, x^8\}$. We just select the tightest clusters. See Figure 10.

[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

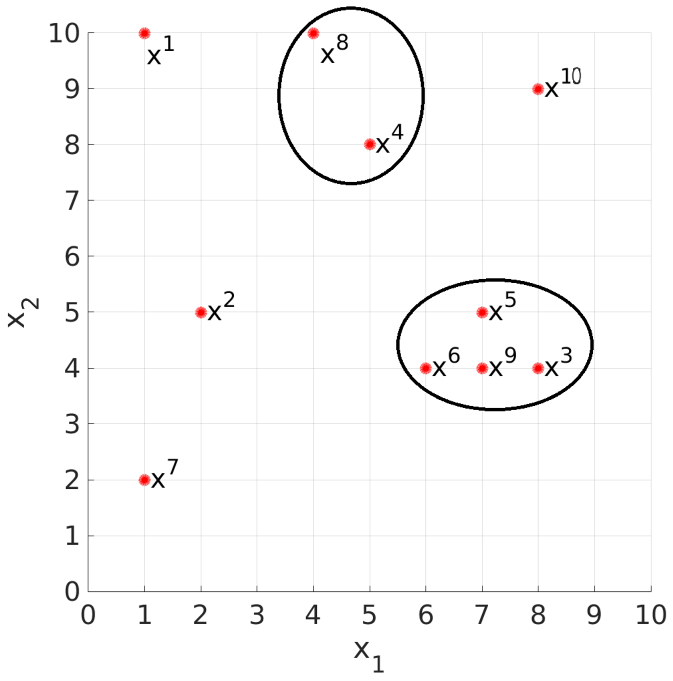


Figure 10: Clusters obtained from the application of DBSCAN with $\epsilon = 2.5$ and $MinPoints = 2$

2. If ϵ becomes 3.5, then the neighborhood of some points will increase. x^8 has now x^1 and x^4 in its neighborhood and x^{10} is also in the neighborhood of x^4. So x^1 and x^{10} joins the cluster $C_2 = \{x^1, x^4, x^8\, x^{10}\}$. Moreover, x^2 and x^7 are now in each other neighborhood. So they will form a new cluster $C_3 = \{x^2, x^7\}$. Finally, The cluster C_1 remains the same as we just increased the ϵ parameter. See Figure 11.

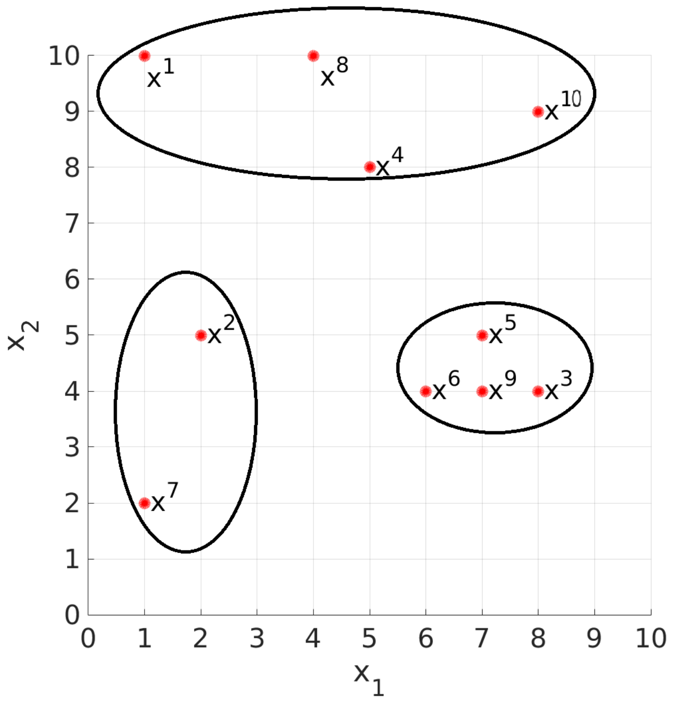


Figure 11: Clusters obtained from the application of DBSCAN with $\epsilon = 3.5$ and $MinPoints = 2$

3. We changed the parameter *MinPoints* from the previous question. Now, each point needs to have at least 3 points (and itself) in its neighborhood to not be considered as an outlier. In the cluster C_3, we only have two points so this cluster disappears. In the cluster C_2, we have 4 points but it has been built by merging two clusters $\{x^1, x^8, x^4\}$ and $\{x^8, x^4, x^{10}\}$. So none of these points has at least 3 points in its neighborhood. The cluster C_2 disappears too. In the cluster C_1, all the points are close enough to all the other points of the clusters so that C_1 still exists. The value here for *MinPoints* enables to select only the most dense clusters. See Figure 12.

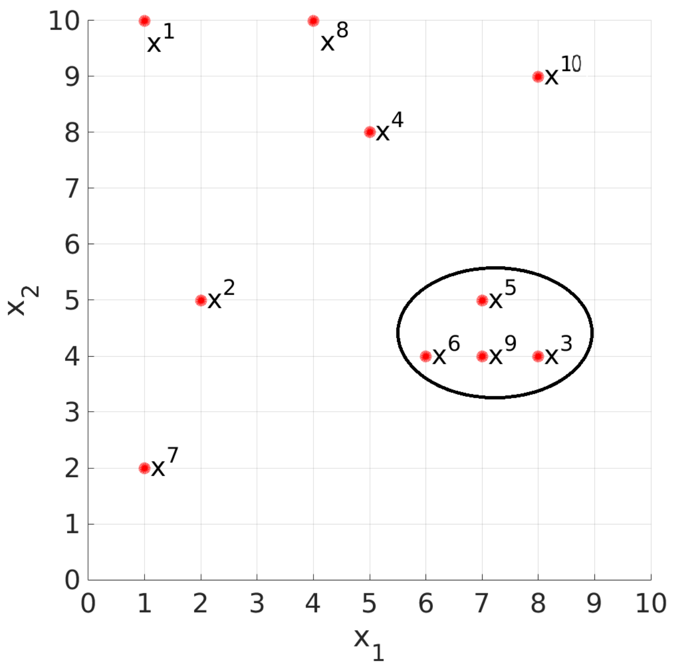


Figure 12: Clusters obtained from the application of DBSCAN with $\epsilon = 3.5$ and *MinPoints* $= 4$

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.

Solutions

A) In each iteration of K-means, you have to compute the distance for each centroids to all the points in order to be able to assign each point to its cluster and then compute the new position of each centroid. Moreover, this needs to be done for each dimension, so the time complexity of K-means is $O(K * M * N)$ (where K is the number of clusters).

For Soft-K-means, you need to compute in addition the responsbility of each cluster for each point. So you have the same number of operations as for K-means multiplied by K (the responsabilities computations). Hence the time complexity of Soft-K-means is $O(K^2 * M * N)$.

B) For DBSCAN, you have to compute for each point the distance to all the other points in order to know if they are part of the neighborhood. So the time complexity of DBSCAN is $O(M^2 * N)$.

However, we can think about a clever way of storing the data points to avoid having to compute several times the distance from a point to points which are close together. If we know that one of these points is not in the neighborhood of the considered point then we don't need to compute the distance to the points in the same partition. A particular interesting structure is the KD-tree which enables to partition the space and helps finding the nearest neighbors of a point quickly. Thanks to this kind of structure we can reduce the time complexity of DBSCAN to $O(M * log(M) * N)$.

C) K-means is the cheapest method. However it can fit solely clusters with globular distributions.

The soft-K-means algorithm has the same disadvantage but it enables to have a better understanding of the clusters (position of the centroids) in case of clusters with some overlap since the points that are difficult to assign to only one of the clusters will have a lower weight.

DBSCAN costs the most but it first can detect clusters which are not globular. Moreover, it is robust to noise since points which are far from the clusters will be considered as outliers.