Clustering with KMeans

Sebastian JORDAN-MONTAÑO

sebastian.jordan@inria.fr

Inria, Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL

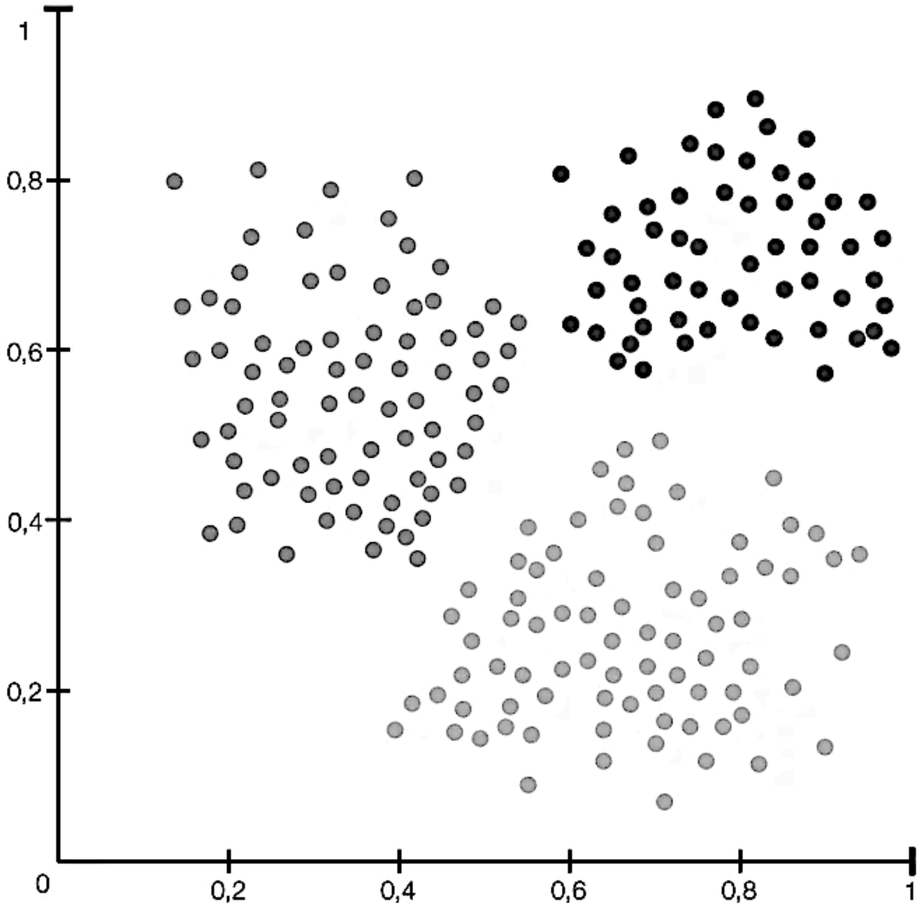
September 2023

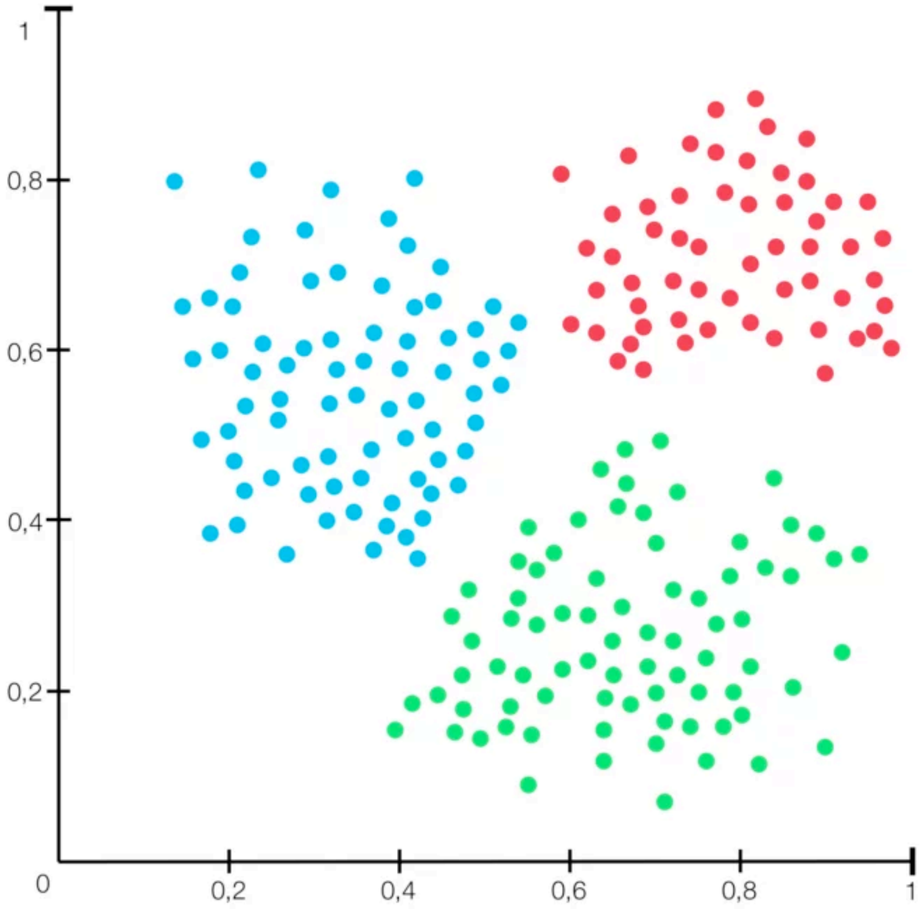
Agenda

- What is clustering
- K-means
- Implementing k-means
- Elbow method
- K-means ++
- Image segmentation using k-means

Clustering

Clustering





K-means

K-Means (simple) algorithm

1. We randomly initialize k points, called cluster centroids.
2. We assign each data-point to its closest centroid and we update the centroid’s coordinates, which are the averages of the points assigned to that centroid so far.
3. We repeat the process for a given number of iterations

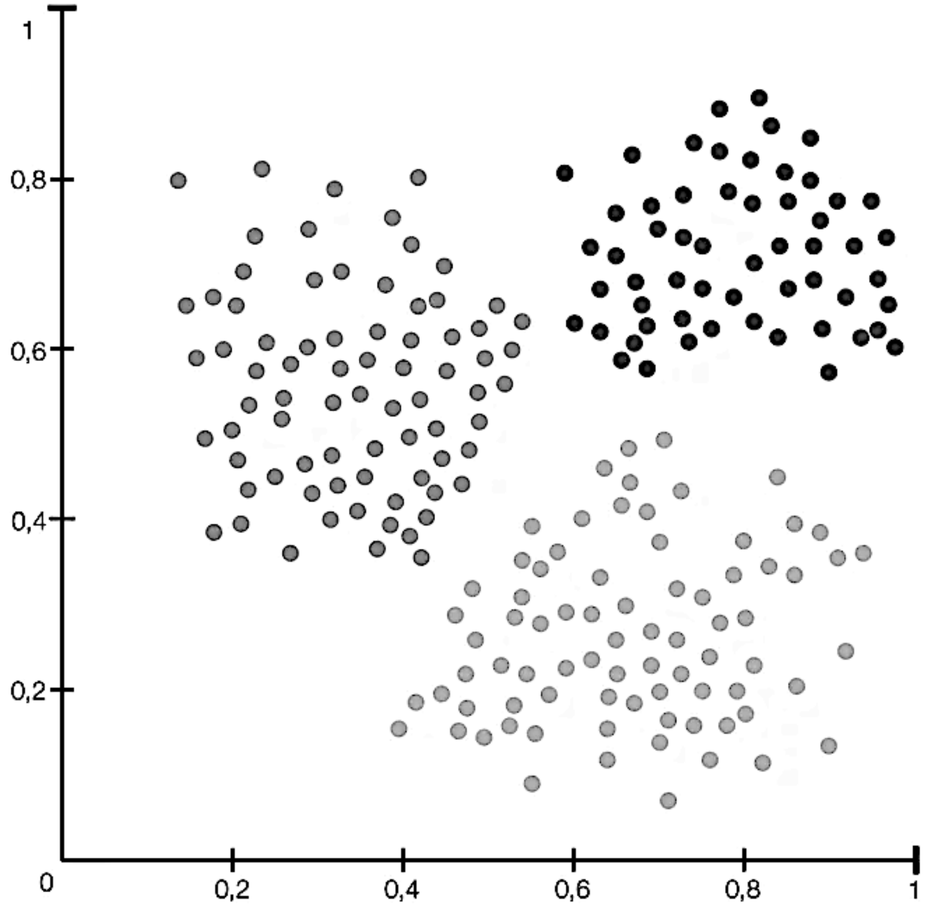
Improving K-means simple algorithm

Random centroid initialization

Option 1:

Take k data points randomly

Random centroid initialization



Improving random centroid initialization

Option 2:

1. Take the min and max value for each dimension of the point. If the point is a normal X,Y point, take the min and max X value, and the min and max Y value.
2. Choose a random number between the min and max range for each point dimension.
3. Repeat until having k centroids

Stop when it converged

Instead of stoping when we reached max iteration, we can stop early if the algorithm converged. How to know if it converged ? By comparing of the centroids changed between iterations

Running it several times

- Run k-means algorithm N-times
  - Each time that we run the algorithm, we keep the best centroids
- Set the best centroids as the centroids

Running it several times

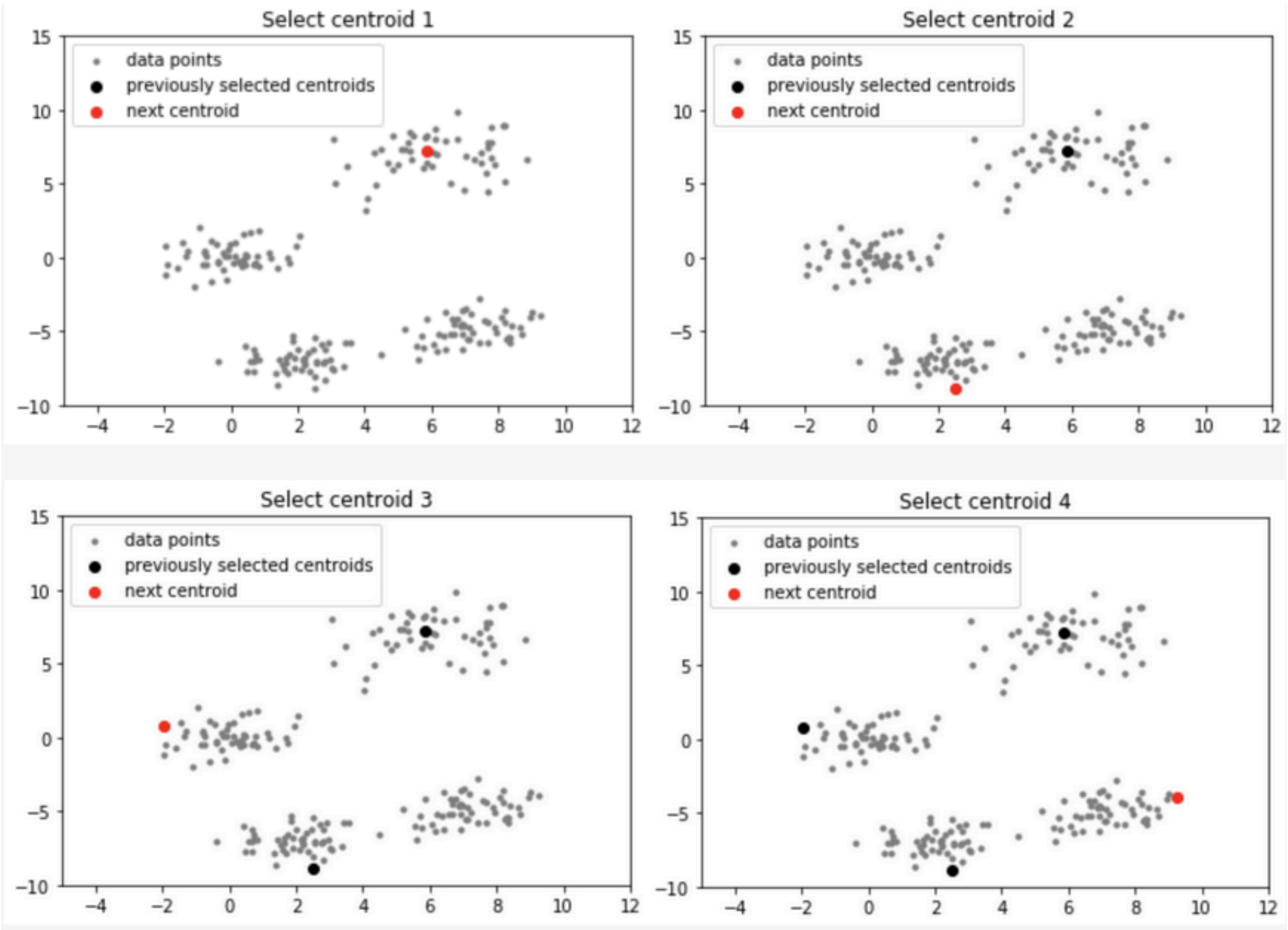
```
fit: aCollectionOfPoints
  | score bestScore bestCentroids |
  timesToRun timesRepeat: [
    self kMeansAlgorithm: aCollectionOfPoints.
    "The best centroid points are the ones that minimize the score.
    The score is the sum of the mean square errors of the points and its cluster."
    self updateBestScoreAndCentroids ].

  centroids := bestCentroids.
  self assignClusterToPoints: aCollectionOfPoints
```

Centroid initialization: K-Means++

1. Randomly select the first centroid from the data points
2. For each data point, compute its distance to the **nearest** centroid.
3. Select the next centroid from the data points such as the point with the largest distance it’s more likely to be chosen.
4. Repeat steps 2 and 3 until k centroids have been sample

Centroid initialization: K-Means++



Source: geeksforgeeks-org

Visualizing each iteration

Look at the class `AIKMeansVisualizer`. Extend the inspector with a presenter that contains the visualizer to visualize the points and the clusters at each iteration.

- `<inspectorPresentationOrder: anInteger title: aTitle>`
- Use `SpRoassalInspectorPresenter` for the inspector presenter.
- The class `AIKMeansVisualizer` returns a chart. You need to ask the chart for its canvas.