

Lecture 12

Fuzzy C-Means
Mean Shift Clustering
OPTICS

Fuzzy C-Means (FCM) is a clustering algorithm that extends the traditional K-Means algorithm by allowing data points to belong to multiple clusters, with degrees of membership represented as fuzzy values between 0 and 1. FCM is particularly useful when data points do not clearly belong to a single cluster but may have partial membership in multiple clusters. Here's how Fuzzy C-Means clustering works:

Basic Procedure:

Initialization: Start by specifying the number of clusters (k) and the fuzziness parameter (m). The fuzziness parameter (m) determines the degree of fuzziness and typically has a value greater than 1.

Random Initialization: Initialize the cluster centroids randomly.

Membership Assignment: For each data point, calculate its degree of membership to each cluster. This is done by computing the fuzzy membership value for each cluster based on the distance between the data point and the cluster's centroid. A common method for computing the fuzzy membership value is using the Euclidean distance and the fuzzy c-means formula:

$$\mu_{ij} = \left(\sum_{k=1}^k \left(\frac{d_{ij}}{d_{ik}} \right)^{\frac{2}{m-1}} \right)^{-1}$$

Where:

μ_{ij} is the membership of data point i to cluster j .

d_{ij} is the Euclidean distance between data point i and the centroid of cluster j .

d_{ik} is the Euclidean distance between data point i and the centroid of cluster k .

m is the fuzziness parameter.

Update Cluster Centers: Calculate new cluster centroids based on the fuzzy memberships. The centroids are determined as a weighted average of all data points, with weights given by the fuzzy memberships.

Convergence Check: Check if the algorithm has converged by comparing the new cluster centroids to the previous ones. If the centroids remain largely unchanged or the change is smaller than a predefined threshold, the algorithm converges.

Termination: If the algorithm has converged, terminate. If not, repeat steps 3-5 until convergence.

Key Considerations:

- FCM allows data points to have partial membership in multiple clusters, providing a more flexible approach to clustering when compared to K-Means.
- The fuzziness parameter (m) determines the degree of fuzziness. A higher m results in more restrictive memberships, while a lower m allows data points to have more equal memberships in multiple clusters.

- The number of clusters (k) is a parameter that needs to be defined in advance, but the algorithm can be sensitive to the initial random cluster centroids.
- FCM may be more computationally intensive than K-Means due to the additional computations required for membership calculations.

Applications: FCM is used in various applications, including image segmentation, pattern recognition, medical diagnosis, and customer segmentation. It is particularly valuable in situations where data points exhibit partial memberships or when hard assignments to clusters are inadequate.

An example of Fuzzy C-Means (FCM) clustering in R using the e1071 package. In this example, we will use synthetic data and apply FCM to group data points into fuzzy clusters. We'll also visualize the results. First, make sure you have the e1071 package installed. You can install it using the following command if you haven't already:

Now, let's create a synthetic dataset and perform Fuzzy C-Means clustering:

```
# Load the e1071 library
library(e1071)
# Generate synthetic data
set.seed(123)
data <- matrix(rnorm(200, mean = c(2, 2), sd = 0.2), ncol = 2)
# Define the number of clusters (k) and fuzziness parameter (m)
k <- 3
m <- 2 # Fuzziness parameter
# Perform Fuzzy C-Means clustering
fcm_result <- cmeans(data, centers = k, m = m)
# Extract cluster assignments
cluster_assignments <- t(fcm_result$membership)
# Plot the data points with fuzzy cluster memberships
plot(data, col = cluster_assignments, pch = 19, main = "Fuzzy C-Means
Clustering Example", xlab = "X", ylab = "Y")
# Add cluster centers as points
points(fcm_result$centers, col = 1:k, pch = 3)
legend("topright", legend = 1:k, col = 1:k, pch = 3, title = "Cluster
Centers")
```

In this code, we: Load the e1071 library, which provides the cmeans function for Fuzzy C-Means clustering. Generate a synthetic dataset with one cluster using random data. Define the number of clusters (k) and the fuzziness parameter (m). Perform Fuzzy C-Means clustering using the cmeans function, specifying the data, the number of clusters, and the fuzziness parameter (m). Extract the fuzzy cluster assignments from the result. Create a scatter plot of the data points with colors representing fuzzy cluster memberships and add the cluster centers as points.

When you run this code, you will see a scatter plot of the synthetic data points, with colors representing fuzzy cluster memberships. The cluster centers are marked with special symbols (in this case, "3" for three clusters). The FCM algorithm has assigned fuzzy memberships to each data point, allowing them to belong to multiple clusters with varying degrees of membership.

You can experiment with different values of k and m to explore how they affect the FCM clustering results and the degree of fuzziness in the memberships.

Mean Shift clustering is a non-parametric and density-based clustering algorithm used to discover clusters in data by locating the modes (peaks) of the data's probability density function (PDF). It is particularly effective for datasets with irregular shapes and varying densities. Here's how Mean Shift clustering works:

Basic Procedure:

Kernel Density Estimation (KDE): Start by performing kernel density estimation on the dataset. This step involves creating a smoothed estimate of the data's underlying PDF. Each data point contributes a kernel (usually a Gaussian) to the estimation. The sum of these kernels creates the overall PDF.

Mean Shift Vector: For each data point in the dataset, calculate a mean shift vector that points toward a mode of the PDF. The mean shift vector is computed as a weighted average of the vectors from the data point to all other data points. The weights are determined by the kernel function.

Mean Shift Iteration: Update the position of each data point by shifting it in the direction of the mean shift vector. The amount of the shift is determined by the kernel bandwidth (bandwidth parameter). This process iteratively shifts data points until they converge to a mode of the PDF.

Clustering: After convergence, data points that end up close to the same mode are assigned to the same cluster. The modes represent the cluster centers.

Label Propagation: Assign a label to each data point based on the cluster it belongs to. Data points that are close to the same cluster center are assigned the same label.

Key Considerations:

- Mean Shift is a mode-seeking algorithm, which means it seeks the modes (peaks) of the data's PDF. It can identify clusters with arbitrary shapes and densities.
- The bandwidth parameter is a crucial hyperparameter that affects the size of the search window for modes. A smaller bandwidth focuses on finer details, while a larger bandwidth creates more extensive clusters.
- Mean Shift is non-parametric, so you don't need to specify the number of clusters in advance.
- It may require multiple iterations to converge to the modes. Convergence can be accelerated by setting a maximum number of iterations or a small convergence threshold.
- Outliers may be detected as separate clusters if they are far from any mode.

Applications: Mean Shift clustering is used in various applications, including image segmentation, object tracking in computer vision, and customer segmentation in marketing. It is particularly valuable when dealing with complex, non-linear data structures and non-convex clusters.

Advantages:

- Effective at identifying clusters with irregular shapes and varying densities.
- No need to specify the number of clusters in advance.

Disadvantages:

- Can be computationally intensive, especially for large datasets.
- Sensitive to the choice of the bandwidth parameter, which may require experimentation to find the optimal value.
- Outliers can be problematic if they are distant from any mode, as they may be assigned to their own clusters.

An example of Mean Shift clustering in R using the `meanshift` package. In this example, we'll generate synthetic data and apply Mean Shift clustering to discover clusters in the data. We'll also visualize the results. First, make sure you have the `meanshift` package installed. You can install it using the following command if you haven't already:

Now, let's create a synthetic dataset and perform Mean Shift clustering:

```
# Load the meanshift library
library(meanshift)
# Generate synthetic data
set.seed(123)
centers <- matrix(c(1, 1, 3, 3, 6, 2), ncol = 2)
data <- rbind(matrix(rnorm(100, mean = centers[1,], sd = 0.6), ncol =
2), # Cluster 1
matrix(rnorm(100, mean = centers[2,], sd = 0.6), ncol = 2), # Cluster 2
matrix(rnorm(100, mean = centers[3,], sd = 0.6), ncol = 2)) # Cluster 3
# Perform Mean Shift clustering
ms_result <- meanshift(data, bandwidth = 1)
# Extract cluster centers and labels
cluster_centers <- ms_result$cluster_centers cluster_labels <-
ms_result$cluster_labels
# Get the number of clusters
n_clusters <- length(unique(cluster_labels))
# Display cluster centers and data points
print("Cluster Centers:")
print(cluster_centers)
print(paste("Number of Clusters:", n_clusters))
# Visualize the clustering results
plot(data, col = cluster_labels, pch = 19, main = "Mean Shift
Clustering Example", xlab = "X", ylab = "Y")
points(cluster_centers, col = 1:n_clusters, pch = 3)
legend("topright", legend = 1:n_clusters, col = 1:n_clusters, pch = 3,
title = "Cluster Centers")
```

In this R code, we: Load the `meanshift` library, which provides Mean Shift clustering. Generate a synthetic dataset with three clusters using random data. Specify the bandwidth parameter (`bandwidth`) to control the size of the search window for modes. Perform Mean Shift clustering using the `meanshift` function, specifying the data and bandwidth. Extract the cluster centers and cluster labels. Get the number of clusters. Display the cluster centers and the number of clusters. Visualize the clustering results by plotting the data points with different colors for each cluster and marking the cluster centers with "x."

When you run this code, you will see a scatter plot of the synthetic data points, with different colors representing the identified clusters. The "x" markers represent the cluster centers discovered by Mean Shift clustering. You can experiment with different synthetic datasets and bandwidth values to observe how Mean Shift clustering adapts to various data distributions and densities.

OPTICS, which stands for **Ordering Points To Identify the Clustering Structure**, is a density-based clustering algorithm used for discovering clusters and hierarchical structures in datasets. It was designed to address some limitations of traditional density-based clustering algorithms like DBSCAN. Here's how OPTICS works:

Basic Procedure:

Reachability Distance: OPTICS uses the concept of "reachability distance" to measure the density of data points. The reachability distance of a data point A with respect to a data point B is the distance between A and B, provided that B is a core point (a point with a sufficient number of neighboring points within a specified radius). If B is not a core point, the reachability distance is defined as an undefined or "infinite" value.

Core Distance: For each data point, the core distance is defined as the distance to its nearest neighbor that has at least a specified number of data points within its radius (a user-defined parameter known as "MinPts").

Ordering of Data Points: OPTICS processes data points and computes their reachability distances and core distances. It orders data points based on their reachability distances, forming a reachability plot. This plot reveals a density-based hierarchical structure.

Cluster Identification: By examining the reachability plot, you can identify clusters of different densities. Clusters are formed based on local maxima in the reachability distances. The local maxima correspond to cluster centers.

Extracting Hierarchical Clusters: The reachability plot provides a hierarchical structure, allowing you to extract clusters at various density levels, from dense core clusters to less dense clusters.

Key Considerations:

- OPTICS is similar to DBSCAN in that it groups data points based on density. However, it doesn't require you to specify the number of clusters in advance.
- OPTICS provides a more comprehensive view of the data's hierarchical clustering structure by identifying clusters of varying densities and shapes.
- The MinPts parameter is essential in OPTICS. It defines the minimum number of data points required within a radius for a point to be considered a core point.
- Noise points, which do not belong to any cluster, are also identified in the output of OPTICS.
- OPTICS is particularly useful for datasets with varying densities and complex hierarchical structures.

Applications: OPTICS has applications in various fields, including data mining, image analysis, and spatial databases. It is valuable when you want to discover clusters in data without needing to specify the number of clusters in advance and when you are interested in understanding the hierarchical relationships between clusters of different densities.

Advantages:

- No need to predefine the number of clusters.
- Reveals hierarchical structures in the data.
- Suitable for datasets with varying densities.

Disadvantages:

- Can be computationally intensive, especially for large datasets.
- Sensitivity to parameter settings, such as MinPts and radius.
- Reachability plots can be challenging to interpret in some cases.

OPTICS clustering in R is not available as a built-in function in standard R packages. However, you can use external libraries or tools to perform OPTICS clustering in R. Here, I will explain the high-level steps to use the `dbscan` package, which includes an OPTICS implementation, to perform clustering.

Here's a high-level overview of how you would use the `dbscan` package for OPTICS clustering:

Load Data: Prepare your dataset. For this example, we'll assume you have a dataset in the form of a data frame named `data`.

Optimize Parameters: OPTICS requires tuning parameters, such as `eps` (the maximum distance between two samples for one to be considered as in the neighborhood of the other) and `MinPts` (the minimum number of data points to form a dense region). You need to estimate these parameters based on your dataset.

Run OPTICS: Use the `dbscan` function from the `dbscan` package to perform OPTICS clustering on your dataset. You can set the `eps` and `MinPts` parameters accordingly.

```
library(dbscan)
# Example parameter values; adjust according to your data
eps <- 0.5 MinPts <- 5
# Run OPTICS clustering
optics_result <- dbscan(data, eps = eps, MinPts = MinPts, method =
"optics")
```

Extract Clusters: You can extract the clusters from the `optics_result` object. This object contains information about the clusters and the core points identified by OPTICS.

```
clusters <- optics_result$cluster
```

Visualization: You can visualize the results as needed, e.g., by creating scatter plots with different colors for each cluster.

Remember that parameter tuning, especially for `eps` and `MinPts`, is crucial in OPTICS clustering. You may need to experiment with different values to identify meaningful clusters in your dataset.

Please note that this example provides a general overview of the steps, and you should adapt the parameters and the data preparation steps to your specific dataset and clustering goals. The quality of the results will depend on the data and parameter settings.

Resources:

1. <https://cran.r-project.org/web/packages/ppclust/vignettes/fcm.html>
2. <https://rpubs.com/rahulSaha/Fuzzy-CMeansClustering>
3. <https://www.kaggle.com/code/ysthehurricane/fuzzy-c-means-clustering-tutorial-r>
4. <https://www.datanovia.com/en/lessons/fuzzy-clustering-essentials/cmeans-r-function-compute-fuzzy-clustering/>
5. <http://meanmean.me/meanshift/r/cran/2016/08/28/meanShiftR.html>
6. <http://cran.nexr.com/web/packages/MeanShift/vignettes/MeanShift-clustering.html>
7. <https://rpubs.com/AnnYang/389433>
8. <https://www.geeksforgeeks.org/ml-mean-shift-clustering/>
9. <https://medium.com/@shruti.dhumne/mean-shift-clustering-a-powerful-technique-for-data-analysis-with-python-f0c26bfb808a>
10. <https://www.kaggle.com/code/pmcgovern/optics-example-in-r>
11. <https://datarundown.com/optics-clustering/>
12. <https://www.geeksforgeeks.org/ml-optics-clustering-explanation/>