Possibilistic C-Means Clustering ToolBox

Hung Tran-Nam

Abstract

Keywords:

Contents

1. PCM_

The `PCM_` function performs Possibilistic C-Means (PCM) clustering on the input data.

Definition

Possibilistic C-Means (PCM) is a clustering algorithm that extends Fuzzy C-Means (FCM) to handle uncertain memberships and noisy data points. It allows each data point to belong to multiple clusters with varying degrees of membership.

Syntax

```
results = PCM_(Data, param, varargin)
```

Inputs

- `Data` - A matrix where each column represents a data point.
- `param` - A structure containing the following fields:
  - `kClust` (required) - Number of clusters.
  - `maxIter` (optional) - Maximum number of iterations (default: 100).
  - `mFuzzy` (optional) - Fuzziness parameter (default: 2.0).
  - `epsilon` (optional) - Convergence threshold (default: 1e-5).
  - `alphaCut` (optional) - Threshold for noise identification (default: 0.5).
  - `K` (optional) - Scaling factor for eta calculation (default: 0.5).

    - x (optional) - Support points for the PDFs.

- varargin - Optional parameters for visualization:
    - 'Visualize' - Visualization type: 'None', 'CDF', or 'CDE' (default: 'None').

Outputs

- results - A structure containing clustering results:
    - Cluster.U - Final partition matrix.
    - Data.fv - Representative PDFs of clusters.
    - iter - Number of iterations performed.
    - ObjFun - Final value of the objective function.
    - Data.Data - Input data.
    - Cluster.IDX - Cluster indices for each data point.
    - Dist.D - Distance matrix between representative PDFs and data.
    - isnoise - Data points identified as noise.

Algorithm Steps

1. **Initialization**: Initialize cluster centers and membership values.
2. **Membership Update**: Update memberships based on distance to cluster centers and fuzziness parameter.
3. **Cluster Center Update**: Update cluster centers based on new memberships.
4. **Convergence Check**: Check convergence based on objective function change or maximum iterations.
5. **Noise Identification**: Identify noise data points based on alpha cut threshold.
6. **Output**: Return final cluster assignments, representative PDFs, and other diagnostic information.

Example

Consider a dataset where each data point represents measurements in a two-dimensional space. We apply PCM clustering to this dataset with the following parameters:

```
% Define the input data
x =

Data = normpdf(x, mu, sigma)

% Define the parameters
param.kClust = 3;        % Number of clusters
param.maxIter = 100;     % Maximum number of iterations
param.mFuzzy = 2.0;     % Fuzziness parameter
param.epsilon = 1e-5;   % Convergence threshold
param.alphaCut = 0.5;   % Threshold for noise identification
param.K = 0.5;          % Scaling factor for eta calculation
param.x = linspace(0, 1, 100); % Support points for the PDFs

% Call the PCM_ function
results = PCM_(Data, param, 'Visualize', 'None');

% Display the results
disp('Cluster Indices:');
disp(results.Cluster.IDX);

disp('Representative PDFs:');
disp(results.Data.fv);

disp('Number of Iterations:');
disp(results.iter);

disp('Objective Function Value:');
disp(results.ObjFun);
```

In this example, we generate random data points in a two-dimensional space and apply PCM clustering to identify clusters in the data. The `PCM_` function computes cluster memberships, updates cluster centers, and identifies noise points based on specified parameters.

2. ExtractKernel

The MATLAB function `ExtractKernel` computes the kernel density estimate (pdf) for images in an `ImageDatastore`.

Description

`ExtractKernel` filters grayscale images from the `ImageDatastore` `imds`, computes the bandwidth (h) for kernel density estimation, and returns the pdf values and corresponding x values.

Syntax

```
[pdf, x] = ExtractKernel(imds)
[pdf, x] = ExtractKernel(imds, Name, Value)
```

Input Arguments

- `imds` - A `matlab.io.datastore.ImageDatastore` object containing images.

Optional Name-Value Pair Arguments

- `'numPoints'` - Number of points for kernel density estimation (default: 1000).
- `'extensions'` - File extensions of images in `imds` (default: {'.png'}).

Output Arguments

- `pdf` - Kernel density estimate values for each image.
- `x` - Points at which the kernel density estimate is evaluated.

Example

```
folderPath = 'link/to/folder/data'
imds = imageDatastore(folderPath);
[pdf, x] = ExtractKernel(imds, 'numPoints', 500);
```

See also

- `ksdensity`
- `imageDatastore`