Subtractive clustering. Modelling of a two-variable nonlinear function based on a data set.

Objectives: understanding the concept and methods for data clustering, understanding the meaning of class and class centre, understanding how the precision of a model is determined.

Note: MATLAB/Simulink is accessed online (https://matlab.mathworks.com/), by logging in with the MS Teams student credentials (surname.name@student.utcluj.ro).

Terms and abbreviations: *crisp clustering, subtractive clustering, Fuzzy C-Means.*

○ Data clustering

Data clustering is a fundamental method for data analysis, where the purpose is to identify the natural data partitions, when having a large set of data. Data clustering is an unsupervised method, as the desired output (number of clusters, membership of an object to a certain cluster) is not known. Typical applications of data clustering include: shape recognition, vector quantization, image segmentation, function approximation, data mining.

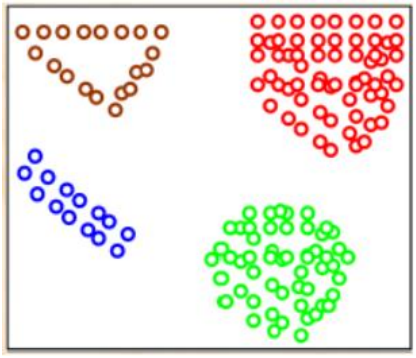
Data clustering is done using a set of features that describe each object of the data set. The result of data clustering is a fixed structure, made of clusters (classes). For each cluster, the following information is known:

- the cluster's location – the coordinates of the cluster's centre
- the cluster's shape
- the membership degree of each object to the cluster

The data partition has the following properties:

- homogeneity inside a cluster - the objects of a cluster should be as resembling as possible
- heterogeneity between clusters - objects belonging to different clusters should be as different as possible

A common measure for the resemblance between objects is the Euclidian distance, illustrated in the image below, where objects belonging to a certain cluster have the same colour.



The data to be clustered is represented as *N*-dimensional vectors:

$$X_i = [x_{i1}, x_{i2}, \dots, x_{iN}], X_i \in \mathbb{R}^N, i = \overline{1, \dots, M}$$

where *N* – number of features for each object, *M* – number of objects (size of dataset).

$$X = \begin{bmatrix} x_{11} & x_{12} \dots & x_{1N} \\ x_{21} & x_{22} \dots & x_{2N} \\ \dots & \dots & \dots \\ x_{M1} & x_{M2} \dots & x_{MN} \end{bmatrix}$$

The objective of clustering is to find *K* vectors that are the centres of the *K* clusters resulted after data clustering:

$$c_k = [x_{k1}, x_{k2}, \dots, x_{kN}], k = \overline{1, \dots, C}$$

○ Types of data clustering

In ***crisp*** or ***hard*** or ***all-or-nothing clustering***, the membership degree of any object to a cluster is either 0 or 1. Each object belongs to a cluster, there are no empty clusters and no cluster can contain all the objects.

In real life, hard clustering is rarely suitable, thus a more "relaxed" clustering method is used, where each object can belong to several different clusters, with a membership degree between 0 and 1. This type of clustering can be:

- fuzzy – Fuzzy C-Means, subtractive clustering
- probabilistic.

○ Subtractive clustering

Subtractive clustering finds the number of clusters and their centres in a set of data. The influence area of the centre of the cluster must be specified, for each dimension.

The steps for subtractive clustering are:

1. Assume that every data point is a potential cluster centre. Compute the probability of this assumption based on the density of the surrounding points.
2. Select the point with the highest potential as the first cluster centre.
3. Eliminate all the points around the centre found in the previous step (according to its influence area), to find the next cluster and its centre.
4. Repeat step 3 until all the points are part of at least one cluster.

○ Modelling a two variable, non-linear function. Application - functional modelling of a transconductance amplifier

Download "*Data_Templates.zip*" and place the archive (using *drag-and-drop*) in the current directory of MATLAB. Double click to unzip and view the contents of the folder.

http://www.bel.utcluj.ro/dce/didactic/sf/lab_eng/10FunctionalModeling/Data_Templates.zip

The archive contains 6 files: 3 files with the datasets (*.mat* file extension), and 3 scripts (*.m* file extension).

To build the model that implements the *gain = f(frequency, temperature)* function, 3 datasets are used:

- "*gendata*" – 179 samples, to generate the initial fuzzy system
- "*antdata*" – 35717 samples, to train the initial fuzzy system
- "*verdata*" – 497 samples, to verify the fuzzy system during training and detect overfitting

The initial ranges of values for the input variables are:

frequency_ini: [1 Hz, 10 MHz]

temperature_ini: [-55, +125]℃

To reduce the range of values and to work with positive values, two transformations are performed:

frequency = *log(frequency_ini)*, so the frequency is in [0, 7]

temperature = *temperature_ini* + 60, so the temperature is in [5, 185]℃

The datasets are matrices, each row represents the values for [*frequency, temperature, gain*].

Exercise 1

Analyse the structure of the 3 datasets, by loading them into the workspace (double-click on each *.mat* file).

Exercise 2

To **generate the initial fuzzy system,** use "*generate_aft.m*" template, in which you add code snippets, according with the instructions in the file.

Generate the initial fuzzy system, using the *genfis2* function. Examine the properties of the fuzzy system, using *Fuzzy Logic Designer*.

What is the value of *radii*?

How many clusters were determined?

How many rules are in the rule base?

What is the connection between the number of clusters, number of rules and the structure of each rule?

What type of fuzzy sets are used for the input variables? What are the expressions of the output fuzzy sets?

What are the centres of the clusters? Visualise the control surface of the initial fuzzy system.

Exercise 3

To **evaluate the precision of the initial fuzzy model,** use the "*errors.m*" template, in which you add code snippets, according with the instructions in the file.

Compute and save in a file:

- the maximum value of the absolute error:

$$absolute\_error_i = |reference\_gain_i - fuzzy\_gain_i|$$

- the maximum value of the relative error:

$$relative\_error_i = \frac{|reference\_gain_i - fuzzy\_gain_i|}{reference\_gain_i}$$

- mean percentual error

$$mean\_percentual\_error = \frac{1}{N}\sum_{i=1}^{N}\frac{|reference\_gain_i - fuzzy\_gain_i|}{refernce\_gain_i}$$

Exercise 4

To **train the initial fuzzy system**, use "*training_aft.m*" template, in which you add code snippets, according with the instructions in the file.

Train the initial fuzzy system, using the *anfis* function.

What type of error does *anfis* use?

Analyse the evolution of errors for the training and test datasets. Does overfitting occur?

Do you think increasing the number of training epochs is necessary? Justify your answer.

Visualise the control surface of the trained fuzzy system. Compare it against the control surface of the initial system.

Identify the changes in the input and output fuzzy sets.

Exercise 5

To **evaluate the precision of the final fuzzy model**, use the "*errors.m*" template, in which you add code snippets, according with the instructions in the file.

Compute and save in a file:

- the maximum value of the absolute error:

$$absolute\_error_i = |reference\_gain_i - fuzzy\_gain_i|$$

- the maximum value of the relative error:

$$relative\_error_i = \frac{|reference\_gain_i - fuzzy\_gain_i|}{reference\_gain_i}$$

- mean percentual error

$$mean\_percentual\_error = \frac{1}{N}\sum_{i=1}^{N} \frac{|reference\_gain_i - fuzzy\_gain_i|}{refernce\_gain_i}$$

Compares these values against the ones obtained for the initial fuzzy system.