User Manual

FCM-APP

Version 0.1 19/07/2024

Table of Contents

[Introduction 3](#_Toc172895879)

[Installation and running (local version) 4](#_Toc172895880)

[FCM Construction and Inference 4](#_Toc172895881)

[FCM Design Tab 5](#_Toc172895882)

[a. Design Manually (Manual construction) 5](#_Toc172895883)

[b. File Upload (Automatic construction) 6](#_Toc172895884)

[FCM visualization 9](#_Toc172895885)

[Inference Tab 10](#_Toc172895886)

[FCM Construction and Inference (Linguistic) 12](#_Toc172895887)

[FCM Design Tab 13](#_Toc172895888)

[a. Design Manually (Manual construction) 13](#_Toc172895889)

[b. File Upload (Automatic construction) 16](#_Toc172895890)

[c. Knowledge Aggregation 18](#_Toc172895891)

[FCM Visualization 21](#_Toc172895892)

[Inference Tab 22](#_Toc172895893)

[FCM Learning 22](#_Toc172895894)

[User Interface 22](#_Toc172895895)

[Data upload 23](#_Toc172895896)

[Data visualization 25](#_Toc172895897)

[Data Preprocessing 26](#_Toc172895898)

[Learning 27](#_Toc172895899)

[Demo (Classification) 27](#_Toc172895900)

[Demo (Regression) 27](#_Toc172895901)

[FAQs 27](#_Toc172895902)

[Contact Information 27](#_Toc172895903)

# Introduction

FCM-APP is designed to offer a flexible and intuitive user interface for Fuzzy Cognitive Map (FCM) construction, analysis, visualization and learning. The software is completely written in Python language with open-source packages such as Streamlit, TensorFlow, Pandas, Numpy, NetworkX, and Matplotlib.

In the current version, the software offers the following features:

1. Manual construction of both numeric and linguistic FCMs
2. Automatic construction of FCMs
3. Knowledge aggregation of linguistic FCMs
4. FCM graph visualization
5. FCM learning for classification tasks (Neural-FCM classifier)
6. FCM learning for regression tasks (Neural-FCM regressor)

In this manual a demo for each of these features is presented\*\*. For further assistance you can find contact information, as well as FAQs, later in this manual.

The app is structured as a four-page app. These pages are: 1) **Home**, 2) **FCM Construction and Inference**, 3) **FCM Construction and Inference (Linguistic)** and 4) **FCM Learning**. Use the sidebar to navigate at each page as it is shown in the Figure 1.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, ιστοσελίδα

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 1 The Home page. All pages are accessible from the sidebar.

*\*\*Important note: This manual assumes that the user is familiarized with FCM aspects and algorithms. If not, we recommend the following articles:*

1. *Kosko, B. (1986). Fuzzy cognitive maps. International journal of man-machine studies, 24(1), 65-75.*
2. *E. Papageorgiou, J. Salmeron: A review of Fuzzy Cognitive Maps research during the last decade. IEEE Transactions on Fuzzy Systems 21, 2012*
3. *G. Felix, G. Nápoles, R. Falcon, K. Vanhoof, W. Froelich, R. Bello: A review on methods and software for Fuzzy Cognitive Maps. Artificial Intelligence Review, 2017*
4. *E. I. Papageorgiou, "Learning Algorithms for Fuzzy Cognitive Maps—A Review Study," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 2, pp. 150-163, March 2012, doi: 10.1109/TSMCC.2011.2138694.*

# Installation and running (local version)

Currently, this version is running locally. To run this app locally to your PC you will need to install Python. Although later versions of Python 3.1x.x may run efficiently, it is recommended to use Python [3.10.4](https://www.python.org/downloads/release/python-3104/) (employed for development) for avoiding any running issues.

Subsequently, in the installation folder there is a “requirements.txt” file which points to the python packages that need to be installed locally. Once, Python and PIP are installed open a terminal in the installation folder (folder containing requirements.txt) and type the following command:

To install these packages, open a new terminal and type the following command:

pip install requirements.txt

*If you are facing any issues with package installation please refer to the official documentation of Python* [*here*](https://packaging.python.org/en/latest/tutorials/installing-packages/)*.*

Once the installation of packages has finished, you can run the app locally by typing the following command in the terminal (inside the installation folder):

streamlit run Home.py

A local Streamlit server will spin up and FCM-APP will open in a new tab in your default web browser in the address http://localhost:8501/.

# FCM Construction and Inference

This page offers widgets for constructing and simulating FCMs. You can construct an FCM **manually** by utilizing the app’s widgets, or **automatically** by uploading a .csv file with the weight matrix. These widgets are available through the **FCM Construction and Inference page.**

This page has **two tabs:**

1. **FCM design**
2. **Inference**

The **inference** tab is becoming available after the definition of weighted interconnections in the **FCM design tab**.

The construction and inference process are explained as a stepped procedure with a demonstration of screenshots in the following subsections. The example of [*Montazemi, A. R., & Conrath, D. W. (1986). The use of cognitive mapping for information requirements analysis. MIS quarterly, 45-56.*] is utilized for demonstration purposes.

## FCM Design Tab

### Design Manually (Manual construction)

1. To construct an FCM manually select the “**Design Manually**” option in the radio widget.
2. Pass the number of concepts your FCM entails, and press enter. The widget accepts any number between 3 and 50.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, ιστοσελίδα

Περιγραφή που δημιουργήθηκε αυτόματα

2)

1)

Once the number of concepts has been defined, two table-widgets appear for naming concepts and defining weighted relations. In more detail:

1. A table widget appears for naming each concept. You can double click at each cell and change the default *“name of concept 1”* value. ***Name all concepts prior to the definition of weighted interconnections****, as naming a concept triggers a re-run and all weights are re-initialized with zero (0.0) values.*

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

3)

Figure 2 Widget for naming the concepts

1. The weight relation table-widget allows the user to **design the map architecture entirely from scratch**. Each cell accepts only values in [-1, 1] and represents the weighted relation among concepts. The comma “,” symbol is used for decimals. In addition, the widget offers an option for downloading the FCM\*\*.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Download button

4)

Figure 3 Widget for defining weighted relations and the download button.

*\*\*Limitation notes: Each re-run (e.g. page reload, page changing)* ***discards all progress****. When working with large FCMs, it is advised to frequently download the weight matrix progress to avoid losing its already defined relations. The next section describes how to upload a .csv file containing an FCM weight matrix.*

### File Upload (Automatic construction)

Automatic construction refers to the utilization of a “.csv” file, containing an FCM weight matrix, for defining an FCM in the **FCM design** tab.

The .csv file is expected to have the **concept names** in the **first row** and the **concept names as** **index** (first column). The dot “.” symbol is used for decimals and comma “,” for separating columns (delimiter). The weights values are expected to be in the [-1, 1]. **FCM weight matrices constructed and downloaded within the app (section 1a), already meet these requirements**. An example of such file, opened with a simple text viewer (*Windows 10* notepad), is presented in Figure 4. The user can choose custom .csv parameters with the reading widgets as it is described later in this section.

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Index

decimal

Concept names

delimiter

Figure 4 An example of .csv file that is expected for automatic FCM construction.

1. Select the **File Upload** option in the radio widget inside the FCM design tab. Use the **upload button widget** to **browse** in the local folders and to upload a weight matrix in .csv format. Currently, you can upload a **.csv file < 10 MB**.

Εικόνα που περιέχει κείμενο, γραμματοσειρά, αριθμός, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

1)

1. Once a .csv file has been selected, csv **parameter widgets** are becoming available for reading any csv file. In detail these are:
   1. The **delimiter** symbol
   2. The **decimal** symbol
   3. Whether the first column contains **numeric index** **instead of the concept names** (usually matrices initialized with other apps such as Microsoft Excel contain a numeric index column).

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

2)

1. If the proper parameters have been selected the weight matrix appears below the widgets (Figure 5). Otherwise, an **error** is displayed (Figure 6).

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 5 Properly reading a .csv file containing the weight matrix

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 6 An example of an error in reading the .csv file containing the weight matrix.

1. Activate the toggle widget if you want to modify the weights in the uploaded weight matrix. This will provide an additional table-widget where the cells are editable\*\* and accept values in [-1,1].

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 7 The widget for modifying weight values (cells).

*\*\*Limitation notes: Concept names and index cannot be modified. Please ensure that your csv file contains the right concept names in the first row and in the index column.*

### FCM visualization

Either in manually or in automatic mode, you can use the app visualization tools to render a plot of the defined FCM **by activating the toggle widget** “Generate FCM graph”. Once it is activated, a map is generated that uses a **diverging blue/red colormap** to **represent the negative and positive relations**. The user can further customize the map by selecting:

1. Different **position layouts** in the concepts:
   1. **Circular** (Position nodes on a circle),
   2. **Random** (Random position of nodes),
   3. **Shell** (Position nodes in concentric circles).
2. **Figure size**. This affects the actual size of the figure and correspondingly the node and text size.
3. **Node size**. It affects the size of the nodes, as well as the size of the text that is proportional to the node size.

The map can be directly downloaded (predefined [DPI](https://en.wikipedia.org/wiki/Dots_per_inch) = 900) with the “Download figure” button.

Εικόνα που περιέχει κείμενο, διάγραμμα, γραμμή, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

3)

2)

1)

Figure 8 The widgets for custom visualization of the FCM

## Inference Tab

Once an FCM has been defined (sections [1a](#_1A._Manual_construction), [1b](#_1B._Automatic_construction)), the inference widgets are becoming available. To access the **inference widgets**, navigate to the **Inference tab** inside the FCM Construction and Inference page.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

5.

4.

6.

3.

2.

1.

Figure 9 The FCM inference widgets inside the **Inference tab**.

These widgets allow to run custom experiments/simulations with different parameters and formulas derived from the FCM theory. More explicitly, the app offers experimentation with the following (Figure 9):

1. **Maximum number of iterations**: Slider widget that defines the upper limit of iterations to be executed before terminating the process, **even if equilibrium has not been reached**.
2. **Equilibrium threshold**: A predefined float (*h*) that is used as an inference stopping criterion. After each iteration the algorithm compares the concepts values (concepts state) ***a***t at the current iteration *t* with the values ***a***t-1 of the previous iteration. This termination criterion can be formally defined as: .
3. **Activation (inference) rule**: The formula for updating the concepts’ values. The available rules the following\*\*:
   * **Kosko**
   * **Modified Kosko**
   * **Rescaled**
4. **Activation function**: A non-linear function that is applied to the updated concepts’ state and affect the convergence of the FCM. The user can select one of\*\*:
   * **Sigmoid**
   * **Bivalent**
   * **Trivalent**
   * **Tanh (Hyperbolic)**
5. **λ, b sigmoid parameters**: These are parameters that are only visible when the Sigmoid activation function is selected. They affect the steepness and the shifting of the sigmoid curve respectively. λ widget accepts integer values in [1, 10], whereas b widget accepts integer values in [-10, 10].
6. **Initial concept state vector**: This table-widget allows the user to provide the initial state by double clicking at each concept and typing a value. For consistency, the accepted value range are based on the activation functions, meaning that:
   * Values in [-1,1] are accepted for **Tanh,** with accuracy of two decimals (0.01 step).
   * Values in [0, 1]are accepted for **Sigmoid,** with accuracy of two decimals (0.01 step).
   * {0, 1} values can be passed with **Bivalent.**
   * {-1, 0, 1} values can be passed with **Trivalent**,

*\*\*Note: For practicality, these formulas are exhibited below the widgets as a text that is dynamically updated based on user’s selection*.

Once the **initial concept state has been defined** (at least one concept has value ≠ 0)**,** a **button** “Inference” is becoming visible that initializes the inference process when pressed. In Figure 10, an inference scenario (simulation) is considered where C1, C2, C3 have initial values of 1. We employ Kosko rule and Bivalent activation function, and we press the “Inference” button to initialize the simulation.

When inference process finishes, a table and a graph appear that entail the inference progress as it is presented in Figure 11. Both the table and the graph can be downloaded locally. It is observed that in this scenario, the FCM converged after 3 iterations. Press the “Clear” button or change the values in the inference widgets, to discard the current results and perform new simulations.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 10 The parameters of an FCM simulation

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, παράλληλα

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 11 The inference results of the FCM simulation

# FCM Construction and Inference (Linguistic)

Another key feature of FCM-APP is the ability **to construct FCMs with linguistic relations and fuzziness**. This key-feature levitates the utilization of expertise for designing powerful FCMs for decision-making. Regarding the **terminology** that Is used in this section, we refer to **linguistic variables,** as the names of the membership functions given to concepts interconnections, that define the degree of causality (fuzzy variable), in the FCM.

This page has two main tabs: 1) FCM Design and 2) Inference. The FCM design offers **three methods for constructing an FCM**:

1. **constructing manually** by defining the fuzzy interconnections from scratch,
2. **constructing automatically** by uploading a **pair of csv and json files** that contain the weight matrix and membership function parameters respectively,
3. **knowledge aggregation**, where multiple pairs of csv and json files are uploaded to construct an FCM based on multiple experts/stakeholders.

These are further explained in the following subsections.

## FCM Design Tab

### Design Manually (Manual construction)

1. To construct an FCM with linguistic variables manually select the **“Design Manually**” option in “FCM Design” tab as it presented in Figure 12.

Εικόνα που περιέχει κείμενο, γραμματοσειρά, αριθμός, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

1)

Figure 12 The “Design Manually” option

1. Once “Design Manually” has been selected, a widget for selecting **the number of linguistic variables,** as well as an **expander for modifying them** is becoming available. You can currently use **5, 7,** or **11 linguistic variables** with **predefined names** as it is presented in Figure 13.

Εικόνα που περιέχει κείμενο, γραμματοσειρά, αριθμός, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

2)

Figure 13 UI widgets for fuzzy variables.

Using the expander, you can **modify the type of MF and choose one of Triangular, Trapezoidal and Gaussian**. Note that the Universe of Discourse [-1, 1] cannot be modified. However, you can modify the parameters of each MF. In Figure 14 we present the widgets for modifying Triangular MFs. There are **two columns of slider widgets** and a plot that is dynamically updated and depicts the **MFs parameters.** Each row of the slider widgets corresponds to one MF. Considering the sliders on the first column, they have **two points** that can be adjusted and correspond to **the start** and **the end values** **of the triangles**. The sliders on the second column are dynamically updated based on the range provided by the sliders in the first column and correspond to the value of the **top corner of the triangle**. Finally, **a “Download JSON” button** is located at the bottom right of the menu with which the user can download the defined fuzzy parameters. This JSON can be used within the app for automatic construction and is further explained in the [next section](#_Automatic_construction_(File).

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Use this button to download the MFs parameters in a JSON file format.

The modifications of the **-High MF** will be visible real time in the graph

This modifies the **top corner** of the **-High MF triangle**

This modifies the **start** and the **end** of the **-High MF triangle**

Figure 14 The UI widgets for modifying the MFs parameters.

1. Pass the number of concepts your FCM entails, and press enter. The widget accepts any number between 3 and 50.

Once the number of concepts has been defined, two table-widgets appear for naming concepts and defining the linguistic relations (Figure 15). In more detail:

1. A table widget appears for naming each concept. You can double click at each cell and change the default *“name of concept 1”* value. ***Name all concepts prior to the definition of linguistic relations****, as naming a concept triggers a re-run and all weights are re-initialized with “None” values*

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

3)

4)

Figure 15 The UI widgets for defining the number of concepts, naming them and the initialization of a matrix with fuzzy interconnections

1. The fuzzy relation table-widget allows the user to **design the map architecture entirely from scratch**. Each cell accepts only the defined linguistic variables providing a dropdown selection menu for selection.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

5)

Figure 16 The widget for defining fuzzy relations

1. After the provision of **fuzzy relations**, a selection widget for **defining the defuzzification method** is becoming available and the **weighted interconnections are automatically calculated** and presented (Figure 17). The user can select one of the **Centroid**, **Bisector** and **Mean of Maximum (MoM) defuzzification methods**.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

6)

Figure 17 The defuzzification step and the result using the centroid method

### File Upload (Automatic construction)

Use this option to upload files for automatic construction of FCMs. For maximum compatibility, we recommend users to **exploit this app to construct FCMs with linguistic interconnections**, with the UI widgets that described in the [previous section](#_Automatic_construction_(File).

1. To construct an FCM directly from files you need to provide a pair of csv/json files. The .csv file is expected to have the **concept names** in the **first row** and the **concept names as** **index** (first column). The values are expected to be in linguistic form (text)\*\*. The .json file provides the information to define the membership functions attributes of the fuzzy relation. Examples of proper .csv and .json files, opened with Windows 10 notepad, are presented in the following figures.

Index

Εικόνα που περιέχει κείμενο, γραμματοσειρά, γραμμή, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Concept names

delimiter

Figure 18 An example of a .csv file containing the relations matrix

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 19 An example of a .json file containing information regarding MFs

1. The UI provides two different upload buttons inside an expander container for uploading csv/json files separately as it is presented in Figure 20. When a .csv is selected, widgets appear for selecting the delimiter symbol and the [index parameters](#_1B._Automatic_construction).

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

.json upload

.csv upload

Figure 20 The UI widgets for uploading the necessary files

1. If both the csv and the json have been uploaded correctly, **the relation matrix**, a **“Modify uploaded matrix” option** and an **expander** for modifying MFs will appear in the UI (Figure 21). An error is displayed otherwise. The **“Modify uploaded matrix” option** allows the user to modify the defined relations among concepts. Both the matrix and the MFs modification, as well as the next step of **defuzzification** have been described in the [previous section](#_Automatic_construction_(File).

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Modification tools

Uploaded matrix

.csv widgets

.json file

.csv file

Figure 21 An example of a correct upload procedure and the UI

\*\*Limitation notes: Currently, compatibility is asserted with predefined linguistic variables (names) that have been presented within the [previous section](#_Automatic_construction_(File).

### Knowledge Aggregation

The “knowledge Aggregation” option allows **multiple pairs of .csv/.json to be uploaded**, thus, **combining knowledge from multiple experts**. For maximum compatibility, It is recommended to use the “Design Manually” option to obtain such pairs.

1. The process, the file characteristics and the UI widgets (Figure 22) for uploading files is similar to what has been already described in [the previous section](#_Automatic_construction_(File). However, it is further required that pairs of .csv/.json files **have the same name** e.g. file1.csv/file1.json. This allows the backend to understand and combine the information of both files.
2. In the following figures we present an example in which three pairs of csv/json files have been utilized. Moreover, when pairs are uploaded, the UI renders new widgets, inside an “Aggregation info” expander container, such as:
   1. A table that presents the **total concepts and their names** (Figure 24).
   2. The **aggregation matrix** that exhibits what linguistic variables are passed at each interconnection (cell) (Figure 24).
   3. The defined MFs that found in jsons (Figure 25).
   4. And two dropdown widgets to select a **specific interconnection for visualizing the MFs aggregation**. For aggregation we employthe max method.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

.csv upload

.json upload

Figure 22 The UI of Knowledge Aggregation option and the upload widgets

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, εικονίδιο υπολογιστή

Περιγραφή που δημιουργήθηκε αυτόματα

An “Aggregation info” expander appears after uploading files

Files have the same name

Figure 23 An example of uploading three pairs of csv/json files

a)

Εικόνα που περιέχει κείμενο, γραμματοσειρά, αριθμός, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

b)

Figure 24 The widgets first widgets a,b, inside the “Aggregation info” expander

Εικόνα που περιέχει γραμμή, διάγραμμα, γράφημα, παράλληλα

Περιγραφή που δημιουργήθηκε αυτόματα

c)

Figure 25 The plots of the MFs (c) that have been found in jsons

Εικόνα που περιέχει γραμμή, κείμενο, στιγμιότυπο οθόνης, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει γραμμή, γράφημα, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει γραμμή, γράφημα, διάγραμμα, πλαγιά

Περιγραφή που δημιουργήθηκε αυτόματα

Aggregated result

Opinion of “expert 3” (pair3)

Opinion of “expert 2” (pair2)

Opinion of “expert 1” (pair1)

d)

Figure 26 The aggregated MFs regarding the degree of causality of concept 2 towards concept 1.

Finally, the user can select defuzzification methods to acquire a numeric matrix and to proceed to FCM simulations. The defuzzification procedure and UI have been described in [a previous section](#_Manual_construction_(Design).

### FCM Visualization

This page provides visualization tools for both the **fuzzy interconnections** and the **weighted interconnections** (after the defuzzification**)**. The latter tool has been already described in a [previous section](#_FCM_visualization) and we prompt the reader to visit this section to understand visualization widgets’ behavior. The former tool utilizes the colors of *cyan* and *purple* to depict the **positive** and **negative** relations respectively.

Εικόνα που περιέχει κείμενο, διάγραμμα, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

## Inference Tab

After the initialization and the defuzzification of the weight matrix the inference process is becoming accessible. The reader is prompt to use the [Section 3 of the arithmetic FCM Construction](#_3._Inference), as in the current app version simulation accepts only numerical initial state values and not linguistic. This may change in future versions.

# FCM Learning

Another key feature of the FCM-APP is the provision of learning algorithms for extracting knowledge and constructing FCM from data. Currently, this version supports the novel Neural-FCM algorithm (To be published) for supervised learning. Future versions will include population-based algorithms such as the Particle Swarm Optimization (PSO), the Real Coded Genetic Algorithm (RCGA) etc.

We begin by providing general information regarding the learning UI and its abilities. Finally, this section includes two demonstrations regarding the learning application in a classification and in a regression task.

## User Interface

The FCM learning page utilizes **four tabs** and the **sidebar**, as it is presented in Figure 27. These tabs are:

1. Data upload
2. Data visualization
3. Data preprocessing
4. Learning

This structure aims to assist non-experienced users as it mimics the data-driven methodology that in general is composed of the following four steps 1) **Data gathering**, 2) **Data understanding**, 3) **Data preparation** and 4) **model selection and learning**. More information is provided in the following sections.

The sidebar is used for information purposes regarding the uploaded dataset and for selecting **the learning task** option, **classification** or **regression**, as it is observed with Figure 27 and Figure 29.

Εικόνα που περιέχει κείμενο, γραμματοσειρά, λογισμικό, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

The four main tabs

Figure 27 The FCM learning page

### Data upload

This tab provides the tools for uploading and reading custom datasets. The UI widgets provide a variety of options for reading .csv files. However, the followings are expected:

1. The dataset is a **two-dimensional table**.
2. Variables in the dataset **must be column-wised** and not row-wised, meaning that the horizontal axis of the table contains the total variables and the vertical axis the total observations.
3. The first row in .csv files is expected to have the **names of the variables**. The following rows must contain the data observations.
4. Both **input** and **output** variables (columns) must **be provided within the same file**.
5. A **single continuous dataset** is expected which can be later divided within the app into **train/validation/test sub datasets**.
6. The maximum size of the file must not exceed the size of 10 MB.

#### Upload widgets

As it is presented in Figure 27, initially, only an **upload button “Browse Files” is available** for selecting and uploading .csv files to the app. When a **file is selected a variety of other widgets are dynamically becoming available,** as it is presented in Figure 28. These are:

* The **delimiter symbol**: The symbol used to separate the columns (variables) in the csv. The predefined value is the *comma* “,”. Other options are available as well.
* The **decimal symbol**: The symbol used to indicate float (decimal) numbers. The predefined value is the *dot* “.”. Comma “,” is available as well.
* The **NAN values**: Symbols for defining missing or [NAN](https://en.wikipedia.org/wiki/NaN) values. Several options are available\*\*.
* The **Advanced options**: This dropdown allows to select multiple columns for datetime index. This is usually beneficial when working with timeseries data and the dataset splits the datetime information into multiple columns.
* The **dataset table**: This widget dynamically exhibits how the dataset will be read inside the app with the provided parameters. If the app cannot read the table an error is presented.
* The **“show dataset info” button**: This button renders two new tabs that provide information regarding the datasets, the statistics and the data type of variables. This information is particularly useful as it indicates columns that:
  + contain non-numeric data (object, boolean, datetime) and must be transformed,
  + contain missing values which need to be processed prior to learning.
* The **“Import data” button**: That imports the dataset in the runtime memory.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα Εικόνα που περιέχει κείμενο, απόδειξη, γραμματοσειρά, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 28 The upload widgets

Once the dataset is imported, three other widgets are rendered that can be utilized for modifying the existing dataset. These widgets are:

* **Change column names popover**: It provides a two-column table-widget, where the user can rename columns. Double click to the cells of the second column to provide a new (text) value and press enter.
* **Change index popover**: This popover provides two options. 1) The user can specify **a column to be used for indexing observations**. 2) The user can convert the current index into a datetime datatype. *Datetime index is used in timeseries data*.
* **Delete column popover:** The user can select a column to be deleted.

Use the **“Submit” button** inside the popovers to apply changes. After a modification is applied, a “**restore changes**” is becoming visible in the UI of the page for discarding all changes and preprocessing methods that were applied in the dataset.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 29 Widgets for dataset modifications

### Data visualization

The app provides **four types of graphs** for data visualization. These are:

* [**Line**](https://en.wikipedia.org/wiki/Line_chart): A graph that uses lines to connect individual data points. A single column is used for visualization each time.
* [**Area**](https://en.wikipedia.org/wiki/Area_chart): Similar to the line graph, it uses lines to connect individual data points and highlights the area below (for positive values) or above (for negative values) the line. A single column is used for visualization each time.
* [**Bar**](https://en.wikipedia.org/wiki/Bar_chart): Graphical display of data using bars of different heights. A single column is used for visualization each time.
* [**Boxplot**](https://en.wikipedia.org/wiki/Box_plot)**:** Demonstrates graphically the locality, spread and skewness groups of numerical data. Multiple columns can be used in a single graph.
* [**Histogram**](https://en.wikipedia.org/wiki/Histogram): A visual representation of the distribution of quantitative data. A single column is used for visualization each time. The user can customize the plot by defining the number of bins and the size of the graph (height, width).

Use the first selection box **“Select chart type”** to choose the graph type. Subsequently, use the column(s) to be plotted and a toggle option **“Generate graph”** appears. **Line, Area and Bar graphs are interactive**, meaning that you can zoom and shift the graph. All graphs can be expanded for a full screen view and downloaded. An example of area chart is displayed in Figure 30.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Expand and download options

Figure 30 An example of an area chart

### Data Preprocessing

Data preprocessing is an essential step in any data-driven method. The app provides several tools for **preparing the data prior to FCM learning**. These tools are grouped into the following tabs (Figure 31):

* **Data Cleansing**: Is used for handling missing data\*\* and for processing outlier data**.**
* **Data Transformation**: Is used for transforming columns. It can be used to either transform text variables\*\* to numerical values, or for [one-hot encoding](https://en.wikipedia.org/wiki/One-hot).
* **Data** [**Normalization**](https://en.wikipedia.org/wiki/Normalization_(statistics)): Changes the range of the values in the dataset.
* **Data split**: Split the dataset into train/validation/test sub datasets and define inputs/output variables\*\*.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, ιστοσελίδα

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 31 The preprocessing structure

\*\****Important notes****: These steps are essential, as missing data and non-numerical columns will crush the learning procedure. Warnings are displayed in the UI when such issues exist in the working dataset (*Figure 32*). The “Learning” tab will be available after defining input/output variables.*

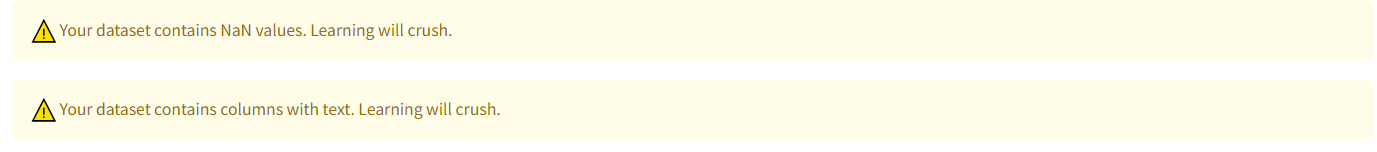


Figure 32 UI warnings for dataset incompatibility

#### Data Cleansing Widgets

Use these widgets to **handle missing data** and to process **outliers**. The UI offers widgets (Figure 33) for:

* Imputation of values in missing data
* Deletion of columns or rows containing missing data
* Outlier processing
* Manual processing

Εικόνα που περιέχει κείμενο, λογισμικό, ιστοσελίδα, τοποθεσία web

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 33 The Data Cleansing UI

##### Handling missing data

1. If missing values do not exist in the dataset, an information message and the dataset information appear on the UI, for more user-friendly experience.

Εικόνα που περιέχει κείμενο, γραμματοσειρά, αριθμός, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 34 UI when no missing data

1. Two widgets are rendered, an **“Imputation”** and a **“Deletion”** expander, in case missing data do exist, as is presented in Figure 33.
2. Imputation expander contains:
   1. a widget for selecting column for processing
   2. a widget for selecting the imputation method. The user can accept one of the following methods:
      1. [**Interpolation**](https://docs.scipy.org/doc/scipy/tutorial/interpolate/1D.html#tutorial-interpolate-1dsection) (Linear)
      2. **Ffil** (Forward fill): Fill values by propagating the last valid observation to next valid.
      3. **Bfil** (Bacward fill): Fill values by using the next valid observation to fill the gap.
      4. **Value**: Provision of a specific value by the user.
      5. **Statistics**: This will render widgets for selecting a statistic value (mean, median, min, max).
   3. Once a column and a method are selected, the UI renders a graph that visualizes the actual values and the imputed values, as it is depicted in Figure 35. Press “Submit” to apply changes.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γράφημα, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 35 An example of filling missing values using the interpolation method.

1. The “Delete rows/column with NaN values” expander entails two widgets (Figure 36):
   1. A widget for selecting a whole column for deletion.
   2. A button for discarding all rows, where at least one column contains a missing value.



Figure 36 Widgets for deleting missing rows/columns with missing data

##### Outlier Removal

It offers widgets for processing outliers based on the **standard score (z-score) method**. Future versions may provide more methods based on clustering algorithms. This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean. More specifically, Z score tells how many standard deviations away a data point is from the mean. Z score = (x -mean) / std. deviation.

As it is presented in Figure 37, the user must provide the column\*\* and the z-score threshold.

A graph then it appears showing which outliers are detected. Subsequently, two tabs are rendered, and the user is prompt to select a) a replacing method or b) the deletion of these rows (the whole row in the dataset is going to be deleted).

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 37 The UI of the z-score outlier removal method.

##### Manual Processing

Finally, the user has the option to manually modify values by enabling the “Manual processing” which renders a table-widget.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

#### Data Transformation Widgets

Data transformation widgets can be employed for categorical data that may be either text or numeric variables. “Integer Encoding” and “One Hot Encoding” methods are offered, as it is presented in Figure 38.

* When **One Hot Encoding\*\*** is selected, the user is prompt to specify a column for encoding. The algorithm identifies the unique values within this column and generates a new table. In this new table, the selected column is removed and replaced by multiple new columns, corresponding to the number of unique values found. Each new column contains binary values (0 or 1), where 1 indicates the presence of the original value in that specific column. Both the current dataset and the encoded dataset are rendered in the UI to assist users understand the transformation (Figure 39). Press “Submit changes” to apply changes.
* When **Integer Encoding** (Ordinal Encoding) is selected the user is prompt to provide a column **containing text values** which will be mapped and transformed to **integer values**. The UI provides a table that shows the transformation (Figure 40). Press “Replace” to apply changes.

***\*\*Important note: Transform the output variable with One Hot Encoding when working with categorical (classification) data for efficient FCM learning*.**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 38 Data transformation options.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 39 One Hot Encoding

Εικόνα που περιέχει κείμενο, γραμμή, λογισμικό, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Figure 40 Integer Encoding

#### Data Normalization Widgets

#### Data Split Widgets

### Learning

## Demo (Classification)

## Demo (Regression)

# FAQs

# Contact Information