For use with Matlab

In case of publication of any application of this method, please, cite the work:

Jérôme Mendes, Francisco Souza, Rui Araújo, and Saeid Rastegar. Neo-fuzzy neuron learning using backfitting algorithm. Neural Computing and Applications, 2018.
DOI: https://doi.org/10.1007/s00521-017-3301-4

http://www.isr.uc.pt/~jermendes/
http://www.isr.uc.pt/~rui/

How to Run:

Main.m – Just run the Main file.

Main Functions:

Main.m – This is the main function

C_BestNumberRules.m - This function returns the best number of rules from the candidate ones.

C_FAM.m - This function represents the learning algorithm (Algorithm 2)

Initialization.m - This function initializes the struct x, i.e. the initial parameters (Algorithm 2 Step 2)

AdditiveModels_C.cpp – This mex function is responsible to the Consequent Design, Algorithm 2 Step 3

FinalyModel_C.cpp - This mex function gives the estimation of the learned neo-fuzzy neuro system.

PlotMFs.m - This function presents the plots of the membership functions of all input variables.

Main Configuration Files:

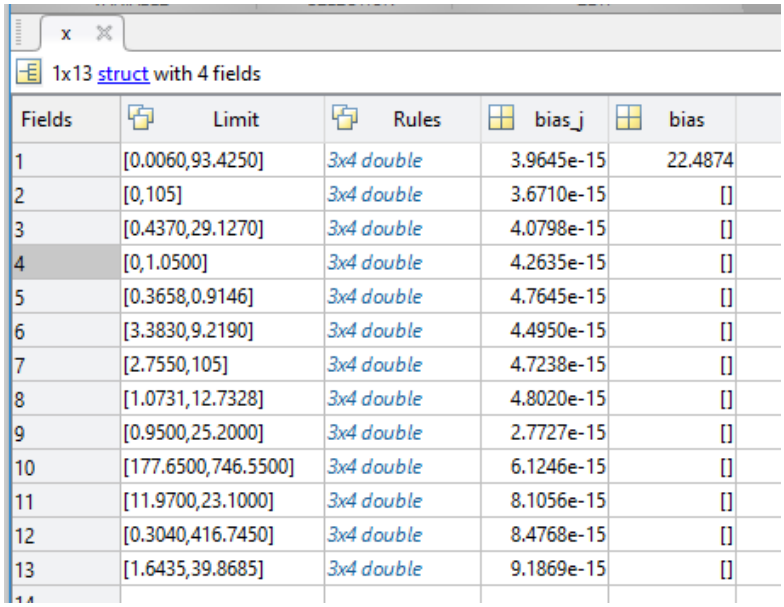
ParametersConfig.m – This file contains the main variables of the algorithm:

- lim_{it} - Maximal number of learning interactions.
- ε - Termination condition defined on Algorithm 2 Step 3.c.ii.
- Number of rules per input variable - The learning algorithm will run varying the fuzzy rules as defined by the user. On the paper as from 2 to 20 rules.

DatasetConfig.m – This file contains the definition of the dataset. Change it to test on another dataset.

Definition of the Struct x:

x is a struct, in which, contains all parameters of the neo-fuzzy neuron system.



x

1x13 struct with 4 fields

Fields	Limit	Rules	bias_j	bias
1	[0.0060,93.4250]	3x4 double	3.9645e-15	22.4874
2	[0,105]	3x4 double	3.6710e-15	[]
3	[0.4370,29.1270]	3x4 double	4.0798e-15	[]
4	[0,1.0500]	3x4 double	4.2635e-15	[]
5	[0.3658,0.9146]	3x4 double	4.7645e-15	[]
6	[3.3830,9.2190]	3x4 double	4.4950e-15	[]
7	[2.7550,105]	3x4 double	4.7238e-15	[]
8	[1.0731,12.7328]	3x4 double	4.8020e-15	[]
9	[0.9500,25.2000]	3x4 double	2.7727e-15	[]
10	[177.6500,746.5500]	3x4 double	6.1246e-15	[]
11	[11.9700,23.1000]	3x4 double	8.1056e-15	[]
12	[0.3040,416.7450]	3x4 double	8.4768e-15	[]
13	[1.6435,39.8685]	3x4 double	9.1869e-15	[]

Figure 1: Example of struct x. Final values of the current example, Boston Housing dataset.

Elements of struct x:

- **Rows**: contain all parameters of the zero-order T-S fuzzy system for each input variable, e.g.:
  - row 1 contains the parameters of the zero-order T-S fuzzy system for the input variable x_1
  - row *j* contains the parameters of the zero-order T-S fuzzy system for the input variable x_j
  - Figure 1 contains the parameters of the 13 input variables of the Boston Housing dataset.

- **Columns:**
  - **First column:** contains the limits of the universe od discourse of the respective input variable.
  - **Second column:** contain the matrix rules of the respective input variable:

	1	2	3	4
1	0.0060	0.0060	46.7155	0.8781
2	0.0060	46.7155	93.4250	-11.0519
3	46.7155	93.4250	93.4250	-7.9929

Figure 2: Example of the matrix rules.

-

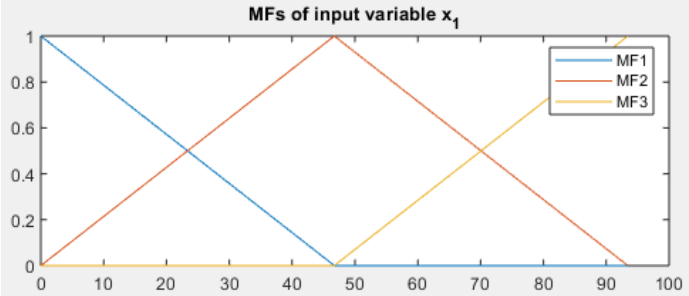


Figure 3: Plot of MFs represented on Figure 2.

    - **Rows:** each row represents a fuzzy rule;
    - **1º column:** represents parameter $a_{j,I}$ (lower limit of the respective membership function);
    - **2º column:** represents parameter $b_{j,I}$ (center value of the respective membership function);
    - **3º column:** represents parameter $c_{j,I}$ (upper limit of the respective membership function);
    - **4º column:** represents the consequent parameter of the respective rule, $\Theta_{j,I}$
  - **Third column:** represents the bias ($bias_j$) of the model of the respective input variable,
  - **Fourth column:** represents the bias (y_0) of the general model. Just the first row is used.

Script Simple_script.m

The main example (file Main.m) runs in order to find the best number of rules from the candidate ones, like the paper, in which, the learning algorithm is tested by varying the number of fuzzy rules between 2 and 20. This script is much faster since not use cross-validation to find the best number of rules.

In the script *Simple_script.m* , the number of rules is defined by the user, and it is unique (scalar, not a vector of possible candidates).