Hands-On Exercises in Data-Driven Fuzzy Modeling

M. Setnes and R. Babuŝka

Control Laboratory
Faculty of Information Technology and Systems
Delft University of Technology
P.O. Box 5031, 2600 GA Delft, the Netherlands.
Tel: +31 15 278 3371, E-mail: {setnes, babuska}@its.tudelft.nl

Contents

1 Introduction

The following fuzzy modeling exercises run in the MATLAB environment. The interaction required by the user is guided by instructions and editable £elds menus. The user can change various parameters of a system, and execute modeling commands to verify the resulting system. In some exercises, it is also possible to manually edit the fuzzy system initially generated with the parameters given by the user, in order to enhance and £ne tune the performance of the model.

There are a few important notes concerning the execution of the exercises:

1. **Start an exercise** by typing its name in the MATLAB command window. **End an exercise** by pressing the `[EXIT]` push-button in the exercise menu.
2. **Avoid** closing any windows yourself as this may mess up the graphs in the exercise.
3. There are no default operations. You have to use the mouse to press the buttons to activate changes, and not the `[ENTER]` key.
4. **To change any parameters of a system**, move the mouse pointer to the current parameter value and press the left mouse button (`[left click]`). Delete the current value using the `[Delete]` or `[BackSpace]` key, and £ll in a new value. (Alternatively, you can highlight the current value using the mouse, and replace it with a new value.) Take care that the new value is within the allowed range of the parameter, see Figure 1. When you have changed a parameter, use the mouse pointer to move on to the next parameter or to press one of the **push buttons** in the window. All updates and actions are executed by pressing the **push buttons** in the current window.
5. **In case of any errors**, just restart the exercise by typing its name in the MATLAB command window.

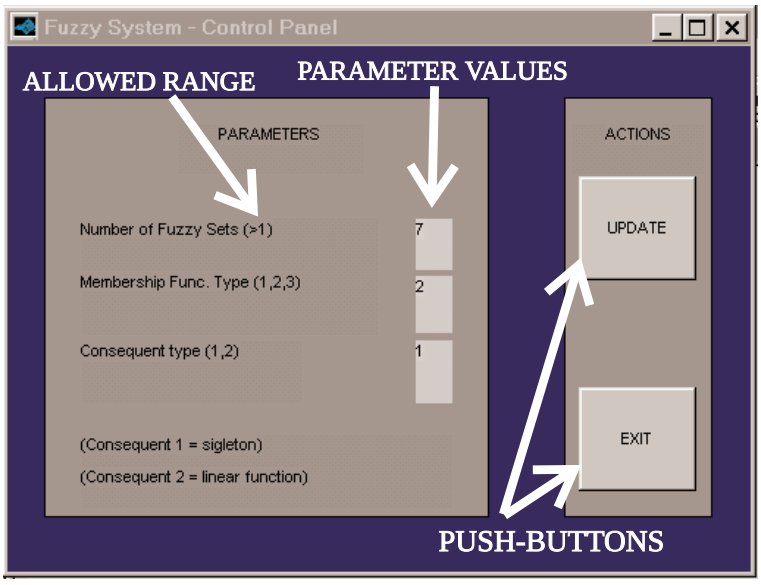

Figure 1: Example of MATLAB exercise interface.

2 Simple function approximation using fuzzy sets

- Exercise name: `fuzzsine`
- Description: function approximation with one-input-one-output using a TS fuzzy model.

System to approximate

The following simple one-input-one-output static function is to be approximated:

$$y = \sin(0.0015x^2)\frac{x^{2.9}}{10000}\,,\ x = 0, 1, 2, \ldots, 100\,. \tag{1}$$

A total of 101 samples of input-output pairs (x, y) equally distributed in x are used as training data.

Fuzzy system

A Takagi-Sugeno fuzzy model is used to approximate the function (1). You can de£ne various fuzzy systems by varying:

1. the number of fuzzy sets for the partitioning of the input x,
2. the membership function type (1=trapezoidal, 2=exponential, 3=sigmoidal), and
3. the type of consequents used by the fuzzy rules, *singleton* (one-£xed value) or *linear functions* ($f(x) = a * x + b$).

Given that this is a one-input system, the number of rules in the fuzzy model equals the number of fuzzy sets used for the partitioning of the input variable x. The output of each rule is a local model that is used to approximate the output of the real function for the range of the input x that is de£ned by the fuzzy set in the premise of the rule.

Instructions

Top left window shows the parameters of the current fuzzy system. You can change these parameters, and view the effect by pressing the `[UPDATE]` button.

The top right window shows the function to be approximated (red), the output of the model (green) and the local models (yellow) between which the fuzzy system interpolates to produce its output.

In the bottom right window you can edit the current membership functions of the fuzzy sets de£ned for the input x. After editing, view the effect by pressing the `[UPDATE]` button.

Questions

1. How does the number of membership functions used effect the approximation?
2. What is a reasonable number of membership functions for this problem, and why?
3. What is the effect of increased overlap of the membership functions on
    a) the quality of the £t (approximation accuracy)?
    b) the interpretation of the consequent parameters?
4. What is the effect of the membership function shape on the output of the model?
5. How important is the position of the membership functions for the approximation accuracy?

3 Function approximation with fuzzy clustering

- Exercise name: `fuzzcl`
- Description: function approximation with one-input-one-output using a TS fuzzy model identi£ed from data by fuzzy clustering.

System to approximate

In this exercise the same simple one-input-one-output static function as in the previous exercise is to be approximated, and the same 101 samples of input-output pairs (x, y) equally distributed in x are used as training data.

Fuzzy system

A Takagi-Sugeno fuzzy model is used to approximate the function (1). The partition of the input space is determined by clustering in the Cartesian product space of x and y. You can identify various fuzzy systems by choosing:

1. the number of clusters to be sought by the clustering algorithm in the product space $x \times y$,
2. the fuzziness of the clusters,
3. the membership function type (1=trapezoidal, 2=exponential, 3=sigmoidal), and
4. the type of consequents used by the fuzzy rules, *singleton* (one-£xed value) or *linear functions* ($f(x) = a * x + b$).

The number of rules in the fuzzy model equals the number of clusters in the product space. The fuzzy sets used to partition the input variable x also equals the number of clusters and are identi£ed by £tting a parametric function (trapezoidal, exponential or sigmoidal) to the projection of the clusters found in $x \times y$ onto x.

The consequent of each rule is a local model that approximates the output of the real function for the range of x for which the rule is applicable. If *singleton* consequents are used, the rules are identi£ed by fuzzy c-means clustering, and the rule consequents are taken to be the y coordinate of the cluster centers. If *linear functions* are chosen as consequents, the Gustafson-Kessel fuzzy clustering algorithm is used. In this case, the rule consequents are derived from the corresponding clusters as the largest of the two axes de£ning the ellipsoid shaped cluster.

Local vs global model

The rule consequents identi£ed from the clusters represent optimal local models. They describe the behavior of the system in the corresponding region. By pressing `[RECALC]`, the rule *consequents* of the current system are recalculated with respect the global (overall) error using Least-Squares.

Instructions

Top left window shows the parameters of the fuzzy system. To identify a system with these parameters, press the [CLUSTER] button.

By pressing the [RECALC], the rule *consequents* of the current system are recalculated without repeating the clustering, using LS estimation. (Changes made to the Membership Function- and Consequent Type parameters are taken into account by [RECALC].)

The top right window shows the function to be approximated (red) and the local models (colors corresponding to the rules respective fuzzy antecedent sets).

The bottom right window shows the function to be approximated (red) and the overall output (yellow) of the fuzzy model (interpolation between local models), and the fuzzy sets de£ned on x. During clustering, this window shows the progress of the cluster algorithm (the cluster centers), and the projection of the clusters onto x.

Questions

1. How does the type of membership functions used in¤uence the position of the cluster centers?
2. What is a reasonable number of clusters for this problem, and why?
3. Cluster with various values for the fuzziness parameter (e.g. 1.2, 2, 3). What is the effect of this parameter on
   a) the overlap of the membership functions?
   b) the interpretation of the consequent parameters?
4. Can you say anything about which consequent type (singleton or linear) is the more sensitive to the variations in the fuzziness parameter?
5. Identify a few different models with a varying number of clusters. Inspect the local models (rule consequents) obtained from the clusters and the approximation quality. Compare with the local models and the approximation obtained when you *recalculate* the consequents of the current model using LS. What can you say in general about the effect of this recalculation on
   a) the local models (consequent parameters)?
   b) the approximation capability of the model?

4 Fuzzy modeling of fermenter pressure dynamics

- Exercise name: `fuzzpres`
- Description: modeling of a two-input-one-output dynamic system with TS and Mamdani fuzzy models using fuzzy grid initialization and manual tuning.

System to approximate

A fuzzy model of the pressure dynamics of a fermenter is to be built. An initial system is identi£ed from sampled data using a fuzzy grid to partition the two dimensional input space. The goodness of the model can be veri£ed in simulations using unseen data.

Fuzzy system

Both Takagi-Sugeno and Mamdani fuzzy models can be used to model the pressure dynamics. The initial partitioning of the input space is determined by the number of membership functions de£ned for the two inputs $u(k)$ and $y(k)$. If n and m fuzzy sets are used for $u(k)$ and $y(k)$, respectively, the number of rules in the initial model becomes $n * m$.

In the TS fuzzy model, the consequent of each rule is a linear function of the inputs $u(k)$ and $y(k)$. The TS consequents are identi£ed by LS £tting using training data. In the Mamdani model, the rule consequents are fuzzy sets. The Mamdani model is a linguistic model, and the best consequent for each rule is selected using the training data.

Step-by-step instructions

After starting the exercise, read the instructions on screen. When the Parameter Menu appears, you are ready to start modeling. Follow the steps below:

1. **select 2 for the partition of valve $u(k)$,**
2. **select 2 for the partition of pressure $y(k)$,**
3. **select trapezoidal membership functions (type 1),**
4. **select Mamdani fuzzy system,**
5. **press `[NEW GRID]` to make a new fuzzy system with the parameters above.**

You now have the *rules* in the lower-left window, the used *membership functions* in the upper-right window and the *input-output mapping* of the model in the lower-right window. Continue by simulating your Mamdani system:

6. **go to the simulation menu by pressing** `[Simulate]`**,**
7. **select one-step-ahead simulation.**

During simulation, the upper-right window shows the *input space* (premise) of the model, with each "mountain" corresponding to the premise of a fuzzy rule. The white ball shows the current state of the system. In the lower-right window, the real systems output is shown in yellow, while the *prediction* by the model is shown in magenta. This simple Mamdani model is able to roughly predict the pressure at the next time step. Now, simulate the model in free-run (dynamic simulation):

8. **go to simulation menu by pressing** `[Simulate]`**,**
9. **select free-run simulation.**

The Mamdani model is not capable of simulating the system dynamically. Improve the dynamic capabilities of the model by de£ning a TS model:

10. **select exponential membership functions (type 2),**
11. **select Takagi-Sugeno fuzzy system type,**
12. **press** `[NEW GRID]` **to make a fuzzy system with the new parameters.**

Notice the *rules* with the functional consequents in the lower-left window, and the new *input-output mapping* of the model in the lower-right window. Continue by simulating your TS system:

13. **go to simulation menu by pressing** `[Simulate]`**,**
14. **select free-run simulation.**

The simple TS model with four rules is capable of dynamically simulating the pressure in the fermenter. Let us study this model in more detail:

15. **press** `[Edit]` **to open a menu for manual tuning,**
16. **press** `[View premise]` **to study the premise space.**

In the upper-right window, you will now see the *premise space* of the model, with the identi£cation data (yellow circles) and the fuzzy regions in which the various rules are applicable.

Due to the grid used to initialize the system, a rule has been located in a region of the input space never visited by the system: *open valve and high pressure*. This situation cannot occur, and the corresponding rule can be removed from the rule base. (Before you continue, notice the mapping of the current system in the lower-right window).

17. **select to remove rule nr. 3,**
18. **press** `[UPDATE]` **to see the effect.**

Observe the effect on the *input-output mapping* of the system. Verify the goodness of the current system by simulation:

19. **press** `[MAIN MENU]` **to exit the edit menu,**
20. **press** `[Simulate]` **and select free-run simulation.**

Even with three rules, the TS model is capable of simulating the system dynamically with a high degree of accuracy.

Try to improve the model by £ne-tuning the membership functions:

21. **press** `[Edit]` **to open the edit menu,**
22. **press** `[View premise]` **to see the location of the rules in the premise space,**
23. **press** `[Edit mfs]` **to open the membership function editors.**

You will see the editors for the membership functions on the lower half of the screen. For instance, try to increase the overlap of the fuzzy sets de£ned for *pressure*. For the *valve*, make the `CLOSED` region bigger and the `OPEN` region smaller. When you are done, press `[UPDATE]`. Observe the resulting input-output mapping. If you are pleased with the result, simulate the model to verify its goodness.

Assignment

Try to obtain a compact and transparent fuzzy model with good prediction capabilities. Here are some tips to get you started:

1. Initialize a new fuzzy model using 3 fuzzy sets to partition the domain of the pressure, and 3 fuzzy sets to partition the domain of the valve.
2. Check its performance by simulation.
3. Go to the edit menu, and remove unnecessary rules.
4. Tune the membership functions to get a reasonable coverage of the input space.
5. Inspect the input-output mapping and the rules.
6. Simulate in free-run to verify the goodness.