

Demonstration of Matlab Fuzzy Logic Packages

There are two different fuzzy logic schemes Mamdani and Sugeno. Matlab supports both but provides a nicer GUI for fitting sugeno based fuzzy inference systems(FIS).

Mamdani fuzzy logic involves utilizing membership functions for both the inputs and the outputs. Fuzzy logic if-then rules are formed by applying fuzzy operations to these membership functions for given inputs. The resulting output membership functions are added together using desired weights yielding a sort of probability function. This function can then be used to estimate the expected value of the output variable.

Sugeno fuzzy logic utilizes the membership functions for inputs and rules to determine the weights for output functions. The output functions are either constants or linear functions of the input variables. If a particular fuzzy logic rule is applicable, the output function gives the value of the output variable if that particular rule were true. The advantage of Sugeno FIS is that each rule produces one value rather than a distribution. Output of a Sugeno FIS is simply a weighted average of the outputs from the different rules.

For our demonstration we will have water levels, air temperatures and salinities from several stations in Nueces bay.

1. Let us begin by loading the data

```
>> load AIWorkshopData
```

Let's suppose we wish to predict the water level at the Aquarium based upon the Water level at White's point using a fuzzy logic inference system. If we were experts in dealing with water levels, we could construct the necessary membership functions and rules by hand.

However since we are not experts and this is just a demonstration, let's take advantage of the functions matlab has to offer for automatically generating a fis. The function `genfis3` is capable of generating both sugeno and Mamdani type fis. We will train our FIS using 1994 data.

2.

```
>> fismat3 = genfis3(WhitePointPWL1994', AquariumPWL1994', ...  
'mamdani', 3)
```

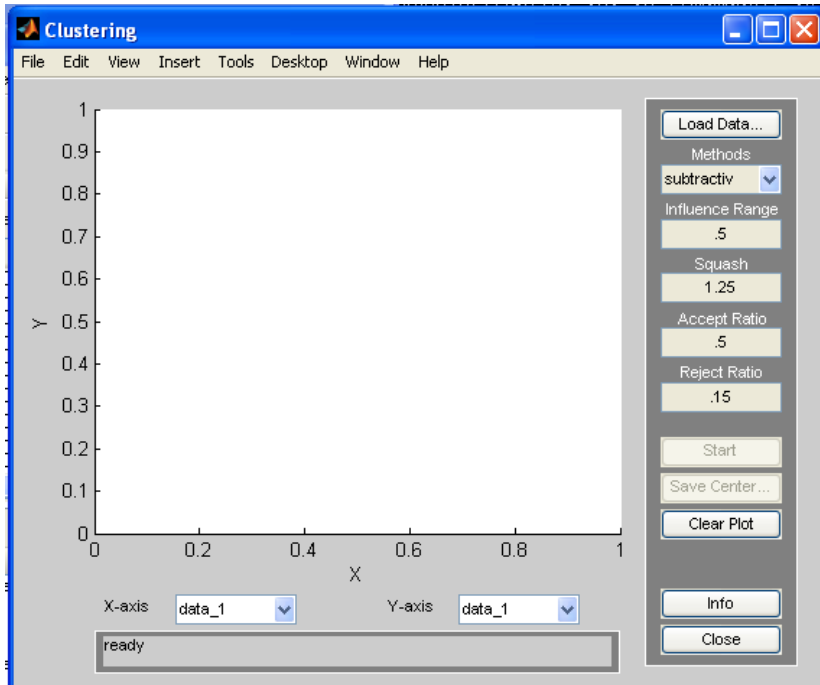
This generates a mamdani FIS utilizing 3 clusters. One way of interpreting the fuzzy logic technique is in terms of clusters. The use of membership functions and rules serves to divide the data into clusters. Particular input values would be mapped more strongly to a particular cluster and each particular cluster has a corresponding membership function for the output variables. This process can be illustrated by utilizing another matlab function.

```
>> train = [WhitePointPWL1994', AquariumPWL1994']
```

```
>> save train.mat train
```

```
>> findcluster
```

This will bring up the matlab tool for finding clusters in a dataset.



Click on **Load Data**. For the file type in `train.mat`. You should see the data plotted at this point. As we are using only two variables, this plot represents all the data. Were there more this tool could be used to view the relationship between any two inputs by changing the data used for the X-axis and the Y-axis. There are two different techniques for determining the clusters, fuzzy c-mean clustering and subtractive clustering. Set the method to `fcm`. The default number of clusters is 2. Click on **start** to have matlab determine the cluster centers. Imagine if you will the shape of the input and output membership functions as well as the rules required for concluding whether a particular data point corresponds to a particular cluster. Experiment with the number of clusters to get a feel for how matlab goes about generating clusters to be used for generating rules and input/output membership functions.

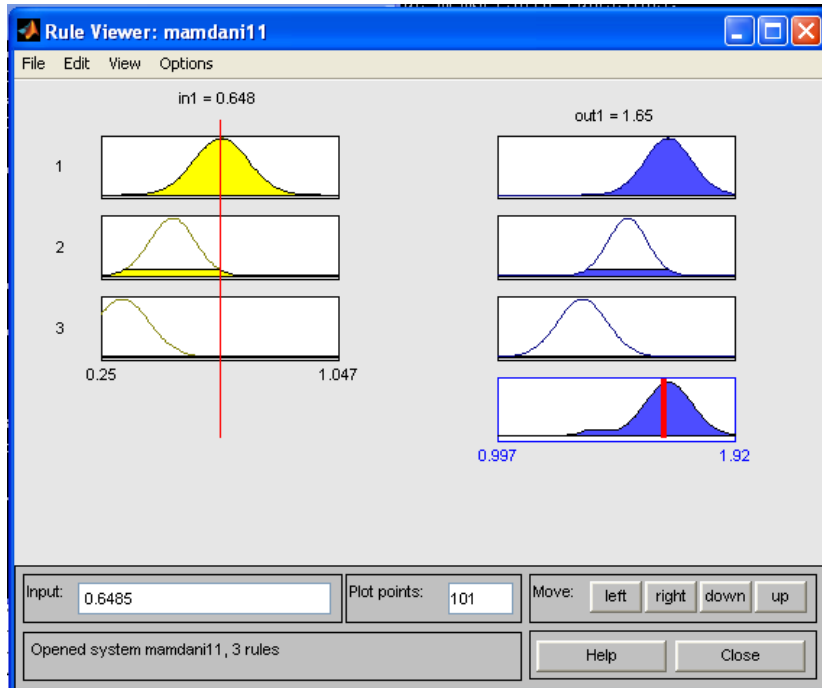
3. Having developed a feel for how matlab generates an FIS let's examine the FIS we generated above.

Close the Clustering window.

Return to the matlab command prompt and enter the following.

```
>> ruleview(fismat3)
```

The following window should open



The ruleview window allows direct simulation of the input to the FIS showing the resulting input and output membership functions. The graphs on the left correspond to the input membership functions, the red line indicates the current input value. It can be moved to simulate different input values. Where this line crosses the membership functions are the values which represent the degree to which the input is believed to belong to that particular membership. The graphs on the right are the output membership functions. Notice that they have been AND'd with the value of the input membership function. The bottom right output function represents the output of the FIS. The red bar on this graph indicates where the estimated value for the output variable lies. Experiment with the FIS to see how the value of the input function effects the output.

4. How well is the FIS performing? Enter the following into the Matlab command prompt:

```
>>
```

```
plot(train(:,1),[train(:,2),evalfis(train(:,1),fismat3)],'.')
```

The blue dots are the actual data. The green is the relationship between the input variable (water level at Whitepoint) and the estimate of the water level at the Aquarium from the FIS. Keep in mind for each input value there is a distribution of output values and the green line only represents the weighted average of this distribution.

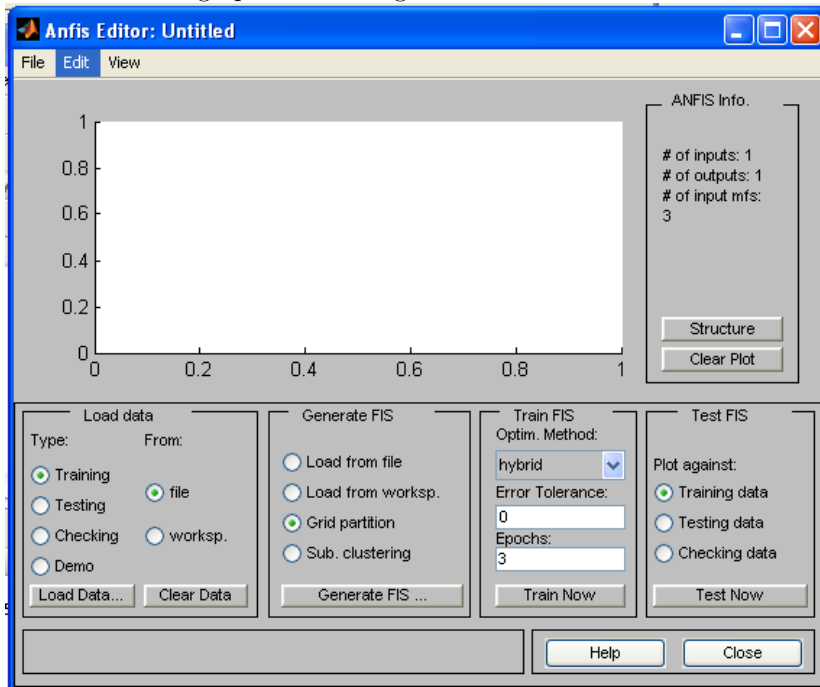
5. Further tweaking. The membership functions as well as other parameters

of the FIS can be examined by using the edit menu from the ruleview window.

6. Sugeno neuro-fuzzy fit. Matlab offers a pretty nifty tool for fitting Sugeno fuzzy logic inference systems. The command is `anfis` which stands for 'Adaptive Neuro-Fuzzy training of Sugeno-type FIS'. The command fits a Sugeno type FIS in much the same manor one would fit a neural network. Given a training set the command goes through several training epochs during which it adjusts the FIS's parameters to better match the desired output. The fitting technique used is a combination of back propagation and least squares. The GUI interface for `anfis` is `anfisedit`. First let's setup the input data.

```
>> train = [WhitePointPWL1994', AquariumPWL1994']
>> test = [WhitePointPWL1995', AquariumPWL1995']
>> check = [WhitePointPWL1997', AquariumPWL1997']
>> anfisedit
```

This should bring up the following:



Load the data respective datasets from the workspace specifying the variables created above. Note the check data set is optional. The check dataset if loaded is used to minimize the amount of overfitting. As `anfis` is training the FIS, beyond a certain point improvements are made in the fit for the training set that result in a poorer fit for the check set. ANFIS attempts to minimize the the model error for the while at the same time minimizing the model error for the training set.

Grid Partition is the default means of generating the FIS. This simply means that the membership functions will be spread across the range of the input variables in a regular grid. Feel free to experiment with creating such an FIS by clicking on **Generate FIS . . .**, followed by **Train Now**. Verify the FIS's performance by selecting the respective data set and then click on **Test Now**. **View the Rules** to verify that the membership functions are in fact spread in a regular manner across the input variable's range. **View the Surface** to see a plot of the FIS output as a function of the input variable. If there were more than one input variables this plot would indeed be a surface.

An FIS generated on a Grid is useful primarily for engineering problems where the input variables can be controlled. For modeling natural systems there are combinations of the input and output variable values that simply are never seen. Also some values are more common than others. A better means of generating an FIS would be to take advantage of any clustering that may exist in the variable space. Select **Sub.clustering** for generating the FIS. There are several parameters which control how the clustering is done of which we will focus upon **Range of Influence**. The algorithm rescales the input and output variables so that they each have a range from 0 to 1, the **Range of Influence** is the desired size of the clusters in the rescaled variable space. For a smaller **Range of Influence** of the algorithm will generate more clusters. Keep in mind that each cluster is characterized by the input membership functions as well as a rule and an output function.

7. Two Inputs. Let's experiment with using the previous days water level from Whitepoint and the Aquarium to predict today's water level at the Aquarium. Enter the following command commands at the Matlab command prompt:

```
>> train =
[WhitePointPWL1994(1:8734)', AquariumPWL1994(1:8734)', AquariumPWL1994(25:8760)']
>> test =
[WhitePointPWL1995(1:8734)', AquariumPWL1995(1:8734)', AquariumPWL1994(25:8760)']
>> check =
[WhitePointPWL1997(1:8734)', AquariumPWL1997(1:8734)', AquariumPWL1994(25:8760)']
```

Return to the anfis GUI. Reload the data, generate the FIS, train the FIS and test the FIS. **View the Rules** to see how two variables are handled. **View the Surface** to see the overall output of the FIS keeping in mind that there are input configurations that do not occur in the dataset.

8. Salinity Data

Let's attempt to predict the salinity at one station based upon the salinity at adjacent stations. Enter the following commands at the matlab prompt to load the dataset.

```
>> train = [Salt01SAL1994', Salt04SAL1994', Salt03SAL1994']
>> test = [Salt01SAL1995', Salt04SAL1995', Salt03SAL1995']
>> check = [Salt01SAL1997', Salt04SAL1997', Salt03SAL1997']
```

```
>> save train.mat train'
```

Get a feel for the data using `findcluster` loading `train.mat` for the data.

Fit a Sugeno FIS to the data set in the same manner as was done for the Water Level data.