Sugeno-Type Fuzzy Inference

The fuzzy inference process we've been referring to so far is known as Mamdani's fuzzy inference method, the most common methodology. In this section, we discuss the so-called Sugeno, or Takagi-Sugeno-Kang, method of fuzzy inference. Introduced in 1985 [Sug85], it is similar to the Mamdani method in many respects. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant.

A typical rule in a Sugeno fuzzy model has the form

If Input 1 = x and Input 2 = y, then Output is $z = ax + by + c$

For a zero-order Sugeno model, the output level z is a constant *(a=b =0)*.

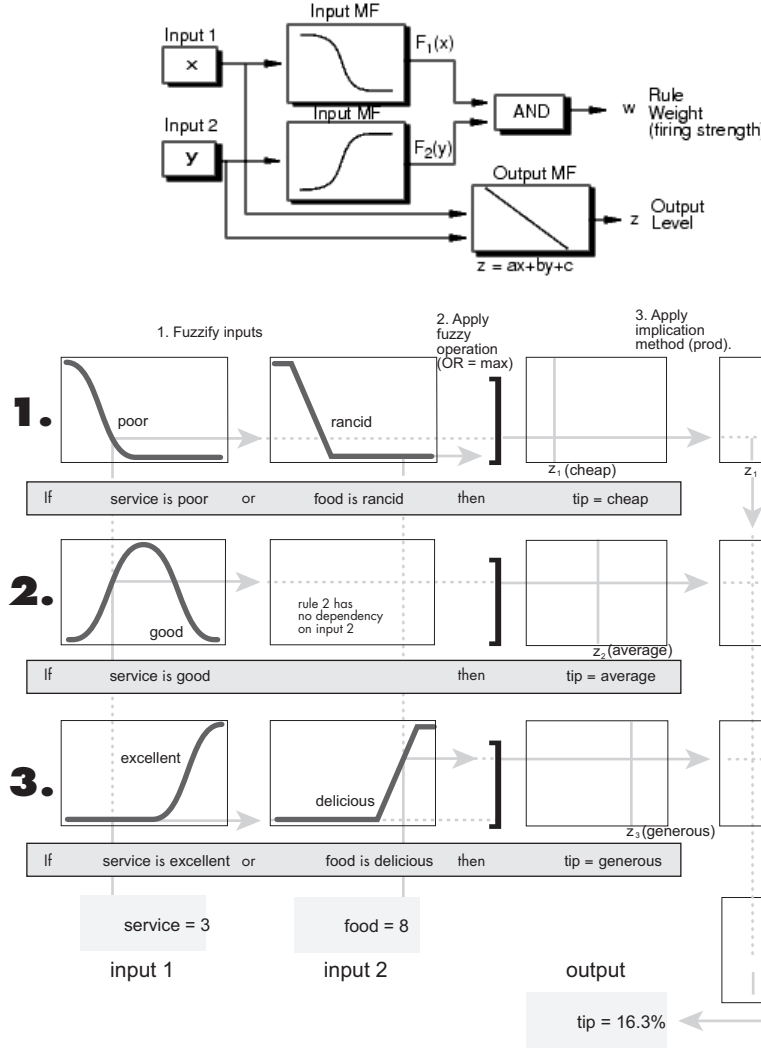
The output level z_i of each rule is weighted by the firing strength w_i of the rule. For example, for an AND rule with Input 1 = x and Input 2 = y, the firing strength is

$$w_i = AndMethod\ (F_1(x),\ F_2(y))$$

where $F_{1,2}(.)$ are the membership functions for Inputs 1 and 2. The final output of the system is the weighted average of all rule outputs, computed as

$$\text{Final Output} = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}$$

A Sugeno rule operates as shown in the following diagram.



The figure above shows the fuzzy tipping model developed in previous sections of this manual adapted for use as a Sugeno system.

anfis and the ANFIS Editor GUI

The basic structure of the type of fuzzy inference system that we've seen thus far is a model that maps input characteristics to input membership functions, input membership function to rules, rules to a set of output characteristics, output characteristics to output membership functions, and the output membership function to a single-valued output or a decision associated with the output. We have only considered membership functions that have been fixed, and somewhat arbitrarily chosen. Also, we've only applied fuzzy inference to modeling systems whose rule structure is essentially predetermined by the user's interpretation of the characteristics of the variables in the model.

In this section we discuss the use of the function `anfis` and the ANFIS Editor GUI in the Fuzzy Logic Toolbox. These tools apply fuzzy inference techniques to data modeling. As you have seen from the other fuzzy inference GUIs, the shape of the membership functions depends on parameters, and changing these parameters will change the shape of the membership function. Instead of just looking at the data to choose the membership function parameters, we will see how membership function parameters can be chosen automatically using these Fuzzy Logic Toolbox applications.

A Modeling Scenario

Suppose you want to apply fuzzy inference to a system for which you already have a collection of input/output data that you would like to use for modeling, model-following, or some similar scenario. You don't necessarily have a predetermined model structure based on characteristics of variables in your system.

There will be some modeling situations in which you can't just look at the data and discern what the membership functions should look like. Rather than choosing the parameters associated with a given membership function arbitrarily, these parameters could be chosen so as to tailor the membership functions to the input/output data in order to account for these types of variations in the data values. This is where the so-called *neuro-adaptive* learning techniques incorporated into `anfis` in the Fuzzy Logic Toolbox can help.

Model Learning and Inference Through ANFIS

The basic idea behind these neuro-adaptive learning techniques is very simple. These techniques provide a method for the fuzzy modeling procedure to *learn* information about a data set, in order to compute the membership function parameters that best allow the associated fuzzy inference system to track the given input/output data. This learning method works similarly to that of neural networks. The Fuzzy Logic Toolbox function that accomplishes this membership function parameter adjustment is called `anfis`. The `anfis` function can be accessed either from the command line, or through the ANFIS Editor GUI. Since the functionality of the command line function `anfis` and the ANFIS Editor GUI is similar, they are used somewhat interchangeably in this discussion, until we distinguish them through the description of the GUI.

What Is ANFIS?

The acronym ANFIS derives its name from *adaptive neuro-fuzzy inference system*. Using a given input/output data set, the toolbox function `anfis` constructs a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using either a backpropagation algorithm alone, or in combination with a least squares type of method. This allows your fuzzy systems to learn from the data they are modeling.

FIS Structure and Parameter Adjustment

A network-type structure similar to that of a neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map.

The parameters associated with the membership functions will change through the learning process. The computation of these parameters (or their adjustment) is facilitated by a gradient vector, which provides a measure of how well the fuzzy inference system is modeling the input/output data for a given set of parameters. Once the gradient vector is obtained, any of several optimization routines could be applied in order to adjust the parameters so as to reduce some error measure (usually defined by the sum of the squared difference between actual and desired outputs). `anfis` uses either back propagation or a combination of least squares estimation and backpropagation for membership function parameter estimation.

Familiarity Breeds Validation: Know Your Data

The modeling approach used by `anfis` is similar to many system identification techniques. First, you hypothesize a parameterized model structure (relating inputs to membership functions to rules to outputs to membership functions, and so on). Next, you collect input/output data in a form that will be usable by `anfis` for training. You can then use `anfis` to *train* the FIS model to emulate the training data presented to it by modifying the membership function parameters according to a chosen error criterion.

In general, this type of modeling works well if the training data presented to `anfis` for training (estimating) membership function parameters is fully representative of the features of the data that the trained FIS is intended to model. This is not always the case, however. In some cases, data is collected using noisy measurements, and the training data cannot be representative of all the features of the data that will be presented to the model. This is where *model validation* comes into play.

Model Validation Using Checking and Testing Data Sets

Model validation is the process by which the input vectors from input/output data sets on which the FIS was not trained, are presented to the trained FIS model, to see how well the FIS model predicts the corresponding data set output values. This is accomplished with the ANFIS Editor GUI using the so-called *testing data set*, and its use is described in a subsection that follows. You can also use another type of data set for model validation in `anfis`. This other type of validation data set is referred to as the *checking data set* and this set is used to control the potential for the model overfitting the data. When checking data is presented to `anfis` as well as training data, the FIS model is selected to have parameters associated with the minimum checking data model error.

One problem with model validation for models constructed using adaptive techniques is selecting a data set that is both representative of the data the trained model is intended to emulate, yet sufficiently distinct from the training data set so as not to render the validation process trivial. If you have collected a large amount of data, hopefully this data contains all the necessary representative features, so the process of selecting a data set for checking or testing purposes is made easier. However, if you expect to be presenting noisy measurements to your model, it is possible the training data set does not include all of the representative features you want to model.

The basic idea behind using a checking data set for model validation is that after a certain point in the training, the model begins overfitting the training data set. In principle, the model error for the checking data set tends to decrease as the training takes place up to the point that overfitting begins, and then the model error for the checking data suddenly increases. In the first example in the following section, two similar data sets are used for checking and training, but the checking data set is corrupted by a small amount of noise. This example illustrates of the use of the ANFIS Editor GUI with checking data to reduce the effect of model overfitting. In the second example, a training data set that is presented to `anfis` is sufficiently different than the applied checking data set. By examining the checking error sequence over the training period, it is clear that the checking data set is not good for model validation purposes. This example illustrates the use of the ANFIS Editor GUI to compare data sets.

Constraints of anfis

`anfis` is much more complex than the fuzzy inference systems discussed so far, and is not available for all of the fuzzy inference system options. Specifically, `anfis` only supports Sugeno-type systems, and these must have the following properties:

- Be first or zeroth order Sugeno-type systems.
- Have a single output, obtained using weighted average defuzzification. All output membership functions must be the same type and either be linear or constant.
- Have no rule sharing. Different rules cannot share the same output membership function, namely the number of output membership functions must be equal to the number of rules.
- Have unity weight for each rule.

An error occurs if your FIS structure does not comply with these constraints.

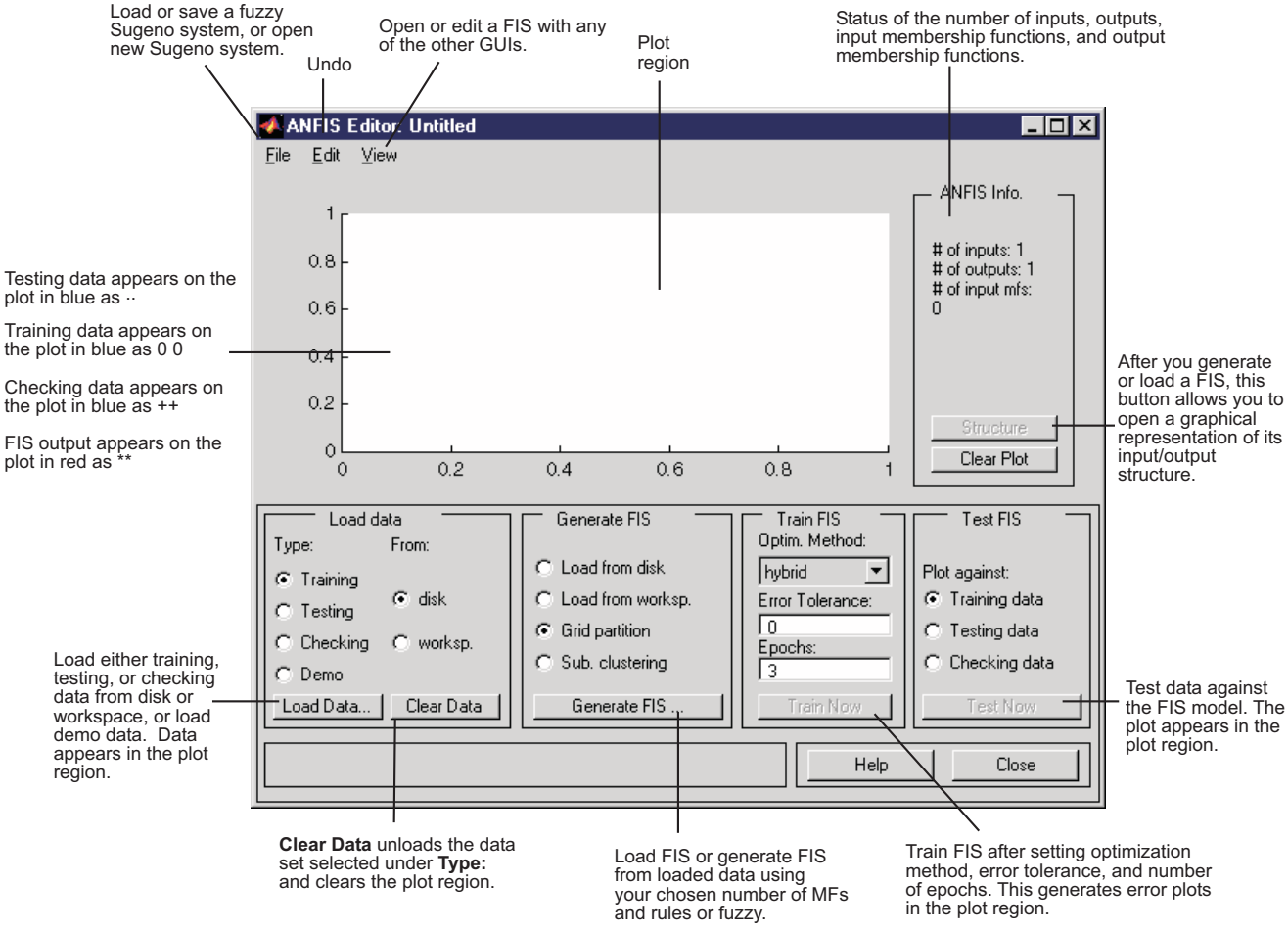
Moreover, `anfis` cannot accept all the customization options that basic fuzzy inference allows. That is, you cannot make your own membership functions and defuzzification functions; you must use the ones provided.

The ANFIS Editor GUI

To get started with the ANFIS Editor GUI, type

```
anfisedit
```

The following GUI will appear on your screen.



From this GUI you can:

- Load data (training, testing, and checking) by selecting appropriate radio buttons in the **Load data** portion of the GUI and then clicking **Load Data**

  The loaded data is plotted on the plot region.
- Generate an initial FIS model or load an initial FIS model using the options in the **Generate FIS** portion of the GUI
- View the FIS model structure once an initial FIS has been generated or loaded by clicking the **Structure** button
- Choose the FIS model parameter optimization method: backpropagation or a mixture of backpropagation and least squares (hybrid method)
- Choose the number of training epochs and the training error tolerance
- Train the FIS model by clicking the **Train Now** button

  This training adjusts the membership function parameters and plots the training (and/or checking data) error plot(s) in the plot region.
- View the FIS model output versus the training, checking, or testing data output by clicking the **Test Now** button

  This function plots the test data against the FIS output in the plot region.

You can also use the ANFIS Editor GUI menu bar to load an FIS training initialization, save your trained FIS, open a new Sugeno system, or open any of the other GUIs to interpret the trained FIS model.

Data Formalities and the ANFIS Editor GUI: Checking and Training

To start training an FIS using either `anfis` or the ANFIS Editor GUI, first you need to have a training data set that contains desired input/output data pairs of the target system to be modeled. Sometimes you also want to have the optional testing data set that can check the generalization capability of the resulting fuzzy inference system, and/or a checking data set that helps with model overfitting during the training. The use of a testing data set and a checking data set for model validation are discussed in “Model Validation Using Checking and Testing Data Sets” on page 2-86. As we mentioned previously, overfitting is accounted for by testing the FIS trained on the training data against the checking data, and choosing the membership

function parameters to be those associated with the minimum checking error if these errors indicate model overfitting. You will have to examine your training error plots fairly closely in order to determine this. These issues are discussed later in an example. Usually these training and checking data sets are collected based on observations of the target system and are then stored in separate files.

Note Any data set you load into the ANFIS Editor GUI, (or that is applied to the command-line function `anfis`) must be a matrix with the input data arranged as vectors in all but the last column. The output data must be in the last column.

ANFIS Editor GUI Example 1: Checking Data Helps Model Validation

In this section we look at an example that loads similar training and checking data sets, only the checking data set is corrupted by noise.

Loading Data

To work both of the following examples, you load the training data sets (`fuzex1trnData` and `fuzex2trnData`) and the checking data sets (`fuzex1chkData` and `fuzex2chkData`), into the ANFIS Editor GUI from the workspace. You may also substitute your own data sets.

To load these data sets from the directory `fuzzydemos` into the MATLAB workspace, type

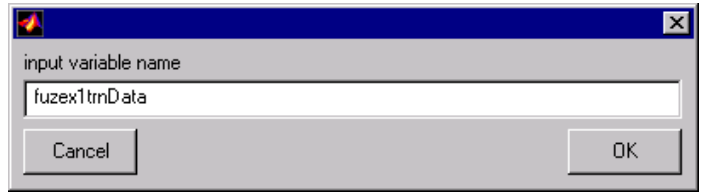
```
load fuzex1trnData.dat
load fuzex2trnData.dat
load fuzex1chkData.dat
load fuzex2chkData.dat
```

from the command line.

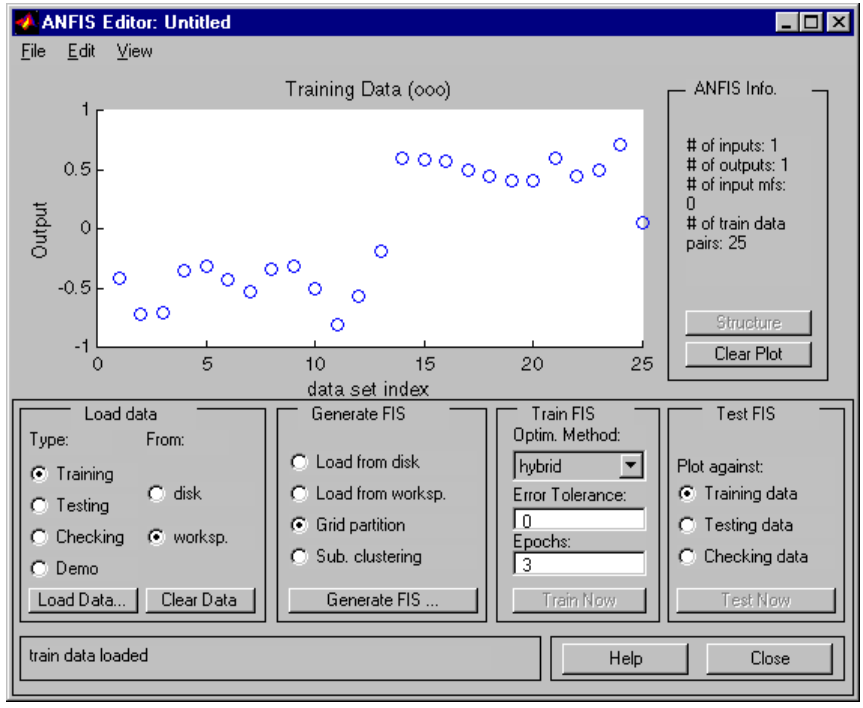
Note You may also want to load your data set from the `fuzzydemos` or any other directory on the disk, using the ANFIS Editor GUI, directly.

Open the ANFIS Editor GUI by typing `anfisedit`. To load the training data set, click **Training worksp.** and then **Load Data**.

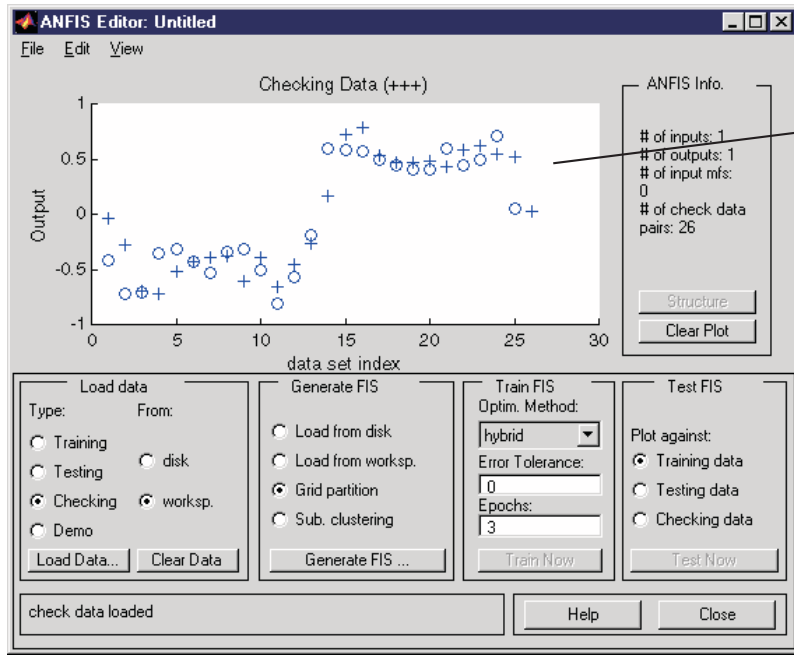
The small GUI window that opens allows you to type in a variable name from the workspace. Type in `fuzex1trnData`, as shown below.



The training data appears in the plot in the center of the GUI as a set of *circles*.



Notice the horizontal axis is marked **data set index**. This index indicates the row from which that input data value was obtained (whether or not the input is a vector or a scalar). Next select the **Checking** check box in the **Type** column of the **Load data** portion of the GUI to load `fuzex1chkData` from the workspace. This data appears in the GUI plot as *plusses* superimposed on the training data.



This data set will be used to train a fuzzy system by adjusting the membership function parameters that best model this data. The next step is to specify an initial fuzzy inference system for `anfis` to train.

Initializing and Generating Your FIS

You can either initialize the FIS parameters to your own preference, or if you do not have any preference for how you want the initial membership functions to be parameterized, you can let `anfis` do this for you.

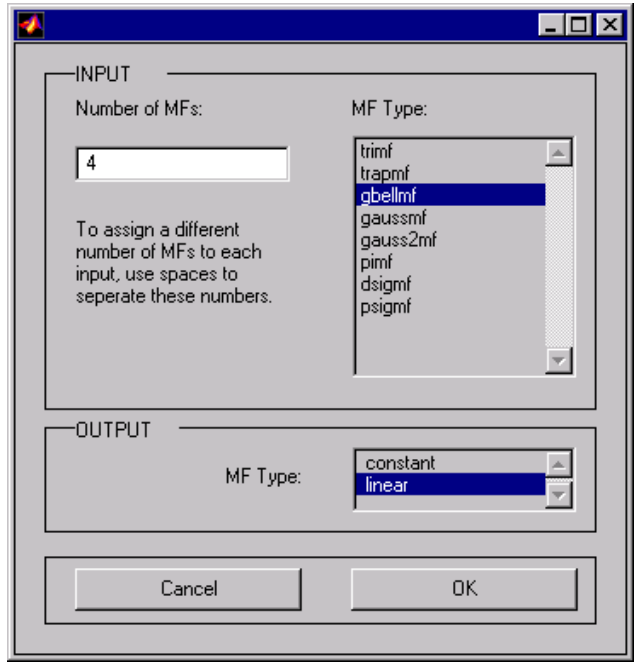
Automatic FIS Structure Generation with ANFIS

To initialize your FIS using `anfis`,

1. Choose **Grid partition,** the default partitioning method. The two partition methods, grid partitioning and subtractive clustering, are described later in “Fuzzy C-Means Clustering” on page 2-114, and in “Subtractive Clustering” on page 2-117.

2 Click on the **Generate FIS** button. This displays a menu from which you can choose the number of membership functions, **MFs**, and the type of input and output membership functions. Notice there are only two choices for the output membership function: `constant` and `linear`. This limitation of output membership function choices is because `anfis` only operates on Sugeno-type systems.

3 Fill in the entries as we've done below, and click **OK.**



You can also implement this FIS generation from the command line using the command `genfis1` (for grid partitioning) or `genfis2` (for subtractive clustering). A command line language example illustrating the use of `genfis1` and `anfis` is provided later.

Specifying Your Own Membership Functions for ANFIS

Although we don't expect you to do this for this example, you can choose your own preferred membership functions with specific parameters to be used by `anfis` as an initial FIS for training.

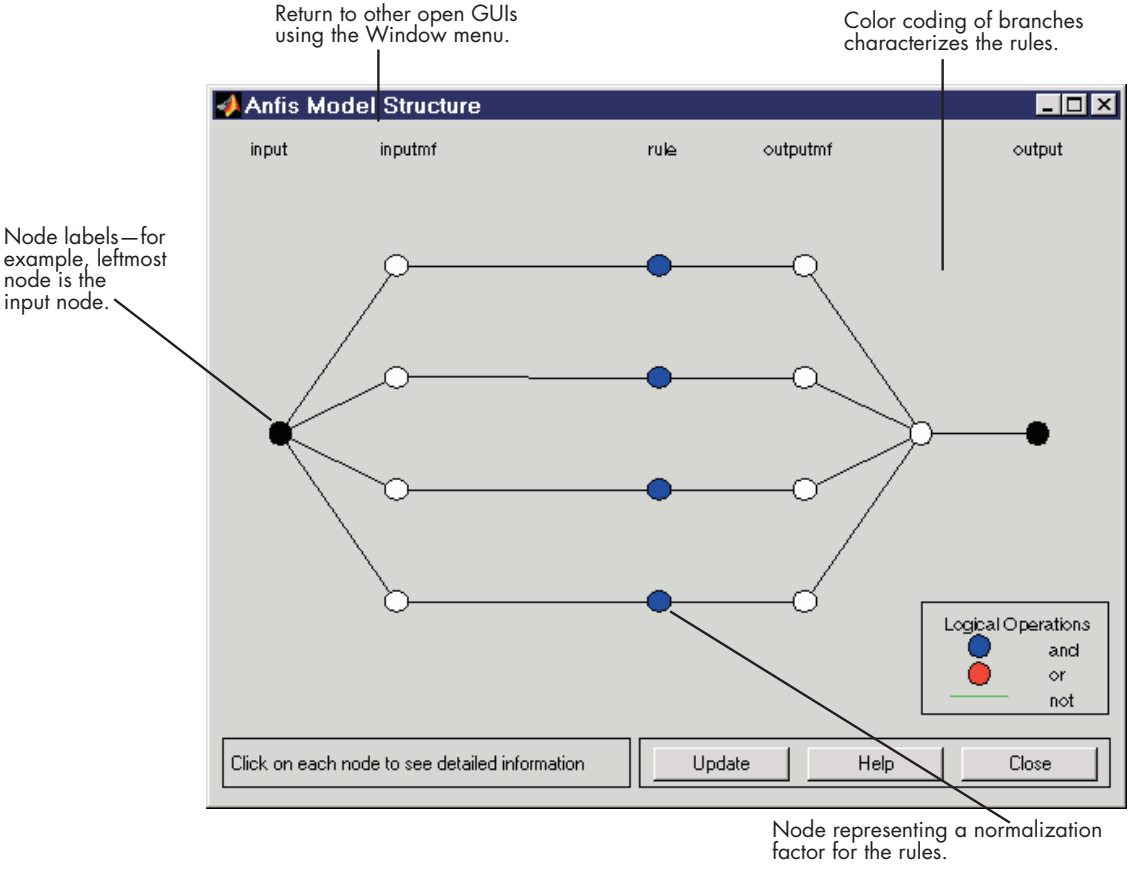
To define your own FIS structure and parameters:

1. Open the **Edit membership functions** menu item from the **View** menu.
2. Add your desired membership functions (the custom membership option will be disabled for `anfis`). The output membership functions must either be all constant or all linear. For carrying out this and the following step, see "The FIS Editor" on page 2-34 and "The Membership Function Editor" on page 2-38.
3. Select the **Edit rules** menu item in the **View** menu. Use the Rule Editor to generate the rules (see "The Rule Editor" on page 2-42).
4. Select the **Edit FIS Properties** menu item from the **View** menu. Name your FIS, and save it to either the workspace or the disk.
5. Use the **View** menu to return to the ANFIS Editor GUI to train the FIS.

To load an existing FIS for ANFIS initialization, in the **Generate FIS** portion of the GUI, click **Load from worksp.** or **Load from disk**. You will load your FIS from the disk if you have saved an FIS previously that you would like to use. Otherwise you will be loading your FIS from the workspace. Either of these radio buttons toggle the **Generate FIS** button to **Load...**. Load your FIS by clicking this button.

Viewing Your FIS Structure

After you generate the FIS, you can view the model structure by clicking the **Structure** button in the middle of the right side of the GUI. A new GUI appears, as follows.



The branches in this graph are color coded to indicate whether or not *and*, *not*, or *or* are used in the rules. Clicking on the nodes indicates information about the structure.

You can view the membership functions or the rules by opening either the Membership Function Editor, or the Rule Editor from the **View** menu.

ANFIS Training

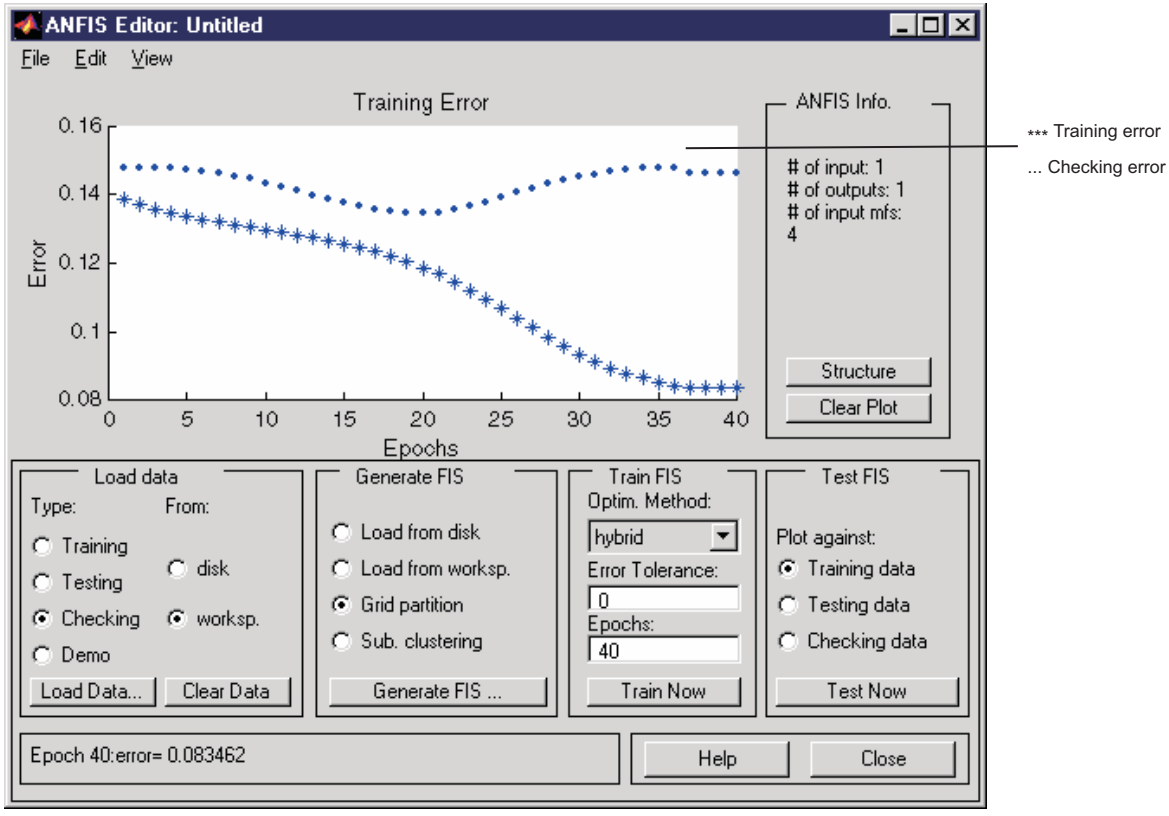
The two `anfis` parameter optimization method options available for FIS training are `hybrid` (the default, mixed least squares and backpropagation) and `backpropa` (backpropagation). **Error Tolerance** is used to create a

training stopping criterion, which is related to the error size. The training will stop after the training data error remains within this tolerance. This is best left set to 0 if you don't know how your training error is going to behave.

To start the training,

- Leave the optimization method at `hybrid`.
- Set the number of training epochs to 40, under the **Epochs** listing on the GUI (the default value is 3).
- Select **Train Now**.

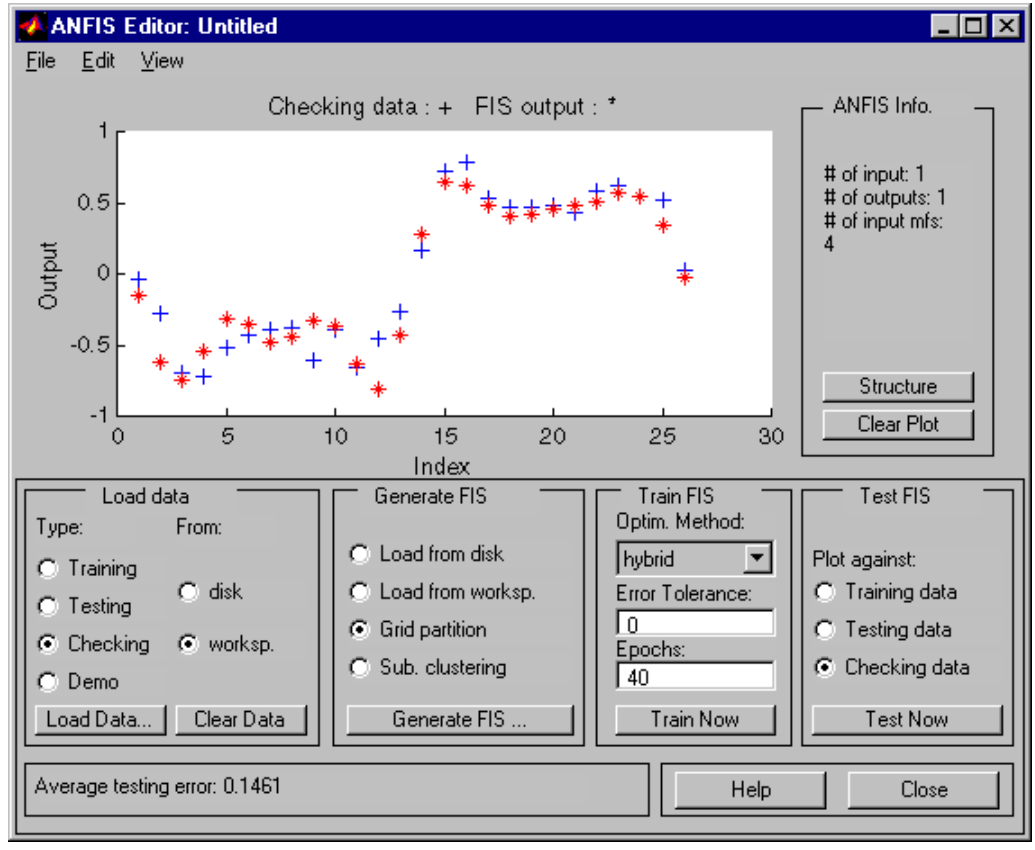
The following should appear on your screen.



Notice how the checking error decreases up to a certain point in the training and then it increases. This increase represents the point of model overfitting. `anfis` chooses the model parameters associated with the minimum checking error (just prior to this jump point). This is an example for which the checking data option of `anfis` is useful.

Testing Your Data Against the Trained FIS

To test your FIS against the checking data, select the **Checking data** check box in the **Test FIS** portion of the GUI, and click **Test Now**. Now when you test the checking data against the FIS, it looks pretty good.



Note on loading more data with anfis If you load data into `anfis` after clearing previously loaded data, you must make sure that the newly loaded data sets have the same number of inputs as the previously loaded ones did. Otherwise, you will have to start a new `anfisedit` session from the command line.

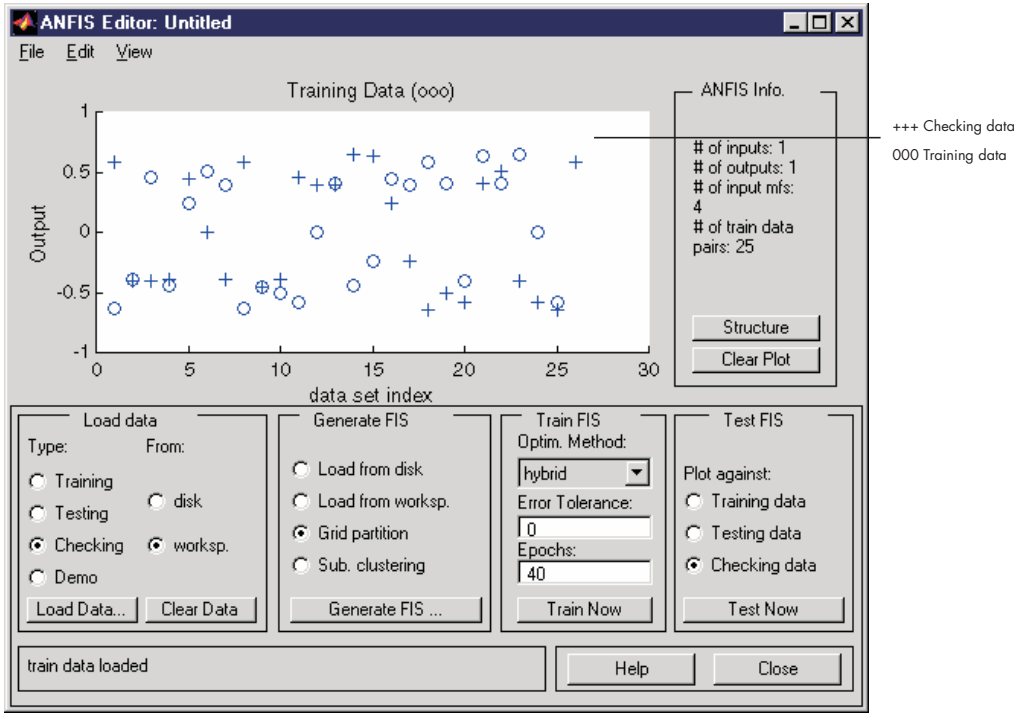
Note on the checking data option and clearing data If you don't want to use the checking data option of `anfis`, don't load any checking data before you train the FIS. If you decide to retrain your FIS with no checking data, you can unload the checking data in one of two ways. One method is to select the **Checking** radio button in the **Load data** portion of the GUI and then click **Clear Data** to unload the checking data. The other method you can use is to close the GUI and go to the command line and retype `anfisedit`. In this case you will have to reload the training data. After clearing the data, you will need to regenerate your FIS. Once the FIS is generated, you can use your first training experience to decide on the number of training epochs you want for the second round of training.

ANFIS Editor GUI Example 2: Checking Data Does not Validate Model

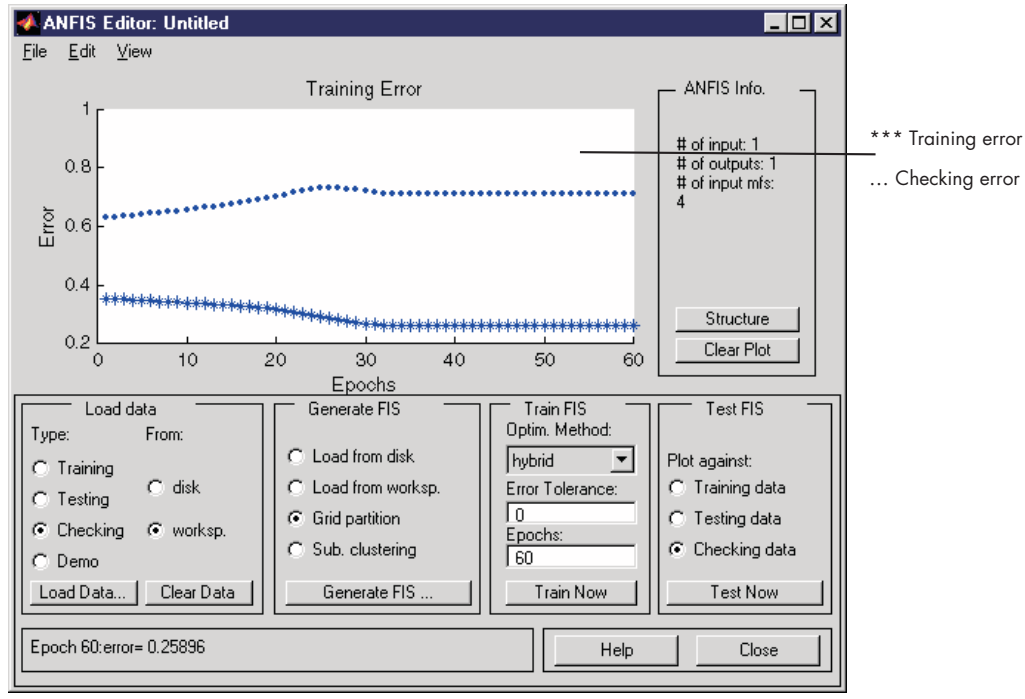
In this example, we examine what happens when the training and checking data sets are sufficiently different. To see how the ANFIS Editor GUI can be used to learn something about data sets and how they differ:

1. Clear both the training and checking data.
2. You can also click the **Clear Plot** button on the right, although you don't have to.
3. Load `fuzex2trnData` and `fuzex2chkData` (respectively, the training data and checking data) from the MATLAB workspace just as you did in the previous example.

You should get something that looks like this.



Train the FIS for this system exactly as you did in the previous example, except now choose 60 **Epochs** before training. You should get the following.

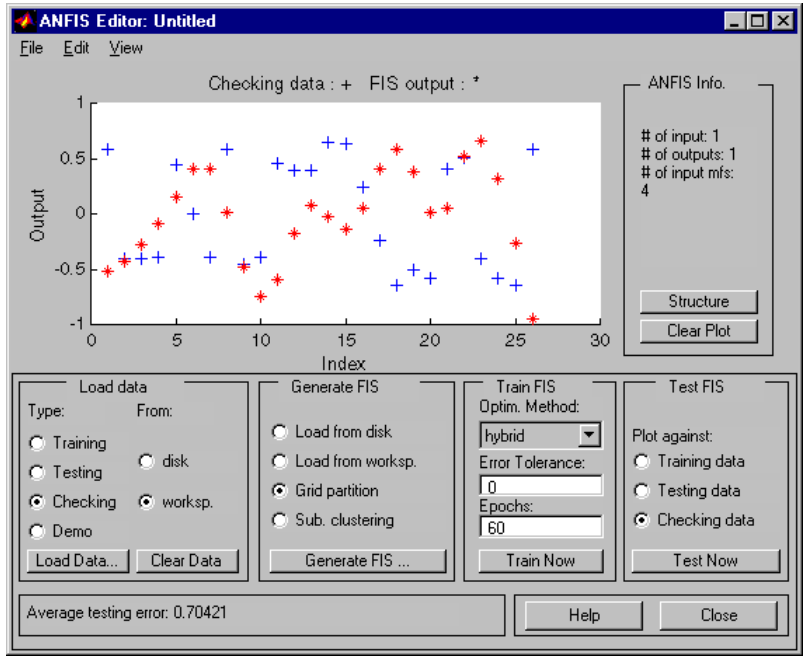


Notice the checking error is quite large. It appears that the minimum checking error occurs within the first epoch. Recall that using the checking data option with `anfis` automatically sets the FIS parameters to be those associated with the minimum checking error. Clearly this set of membership functions would not be the best choice for modeling the training data.

What's wrong here? This example illustrates the problem discussed earlier wherein the checking data set presented to `anfis` for training was sufficiently different from the training data set. As a result, the trained FIS did not capture the features of this data set very well. This illustrates the importance of knowing the features of your data set well enough when you select your training and checking data. When this is not the case, you can analyze the checking error plots to see whether or not the checking data performed sufficiently well with the trained model. In this example, the checking error is sufficiently large to indicate that either more data needs to be selected for training, or you may want to modify your membership function choices (both the number of membership functions and the type). Otherwise the system

can be retrained without the checking data, if you think the training data captures sufficiently the features you are trying to represent.

To complete this example, let's test the trained FIS model against the checking data. To do so, select **Checking data** in the **Test FIS** portion of the GUI, and click **Test Now**. The following plot in the GUI indicates that there is quite a discrepancy between the checking data output and the FIS output.



anfis from the Command Line

As you can see, generating an FIS using the ANFIS Editor GUI is quite simple. However, as you saw in the last example, you need to be cautious about implementing the checking data validation feature of `anfis`. You must check that the checking data error does what is supposed to. Otherwise you need to retrain the FIS.