Package ‘FuzzyR’

October 12, 2022

Type Package
Title Fuzzy Logic Toolkit for R
Version 2.3.2
Depends R (>= 3.6.0)
Imports splines, shiny, plyr, grid, stats, graphics
Description Design and simulate fuzzy logic systems using Type-1 and Interval Type-2 Fuzzy Logic. This toolkit includes graphical user interface (GUI) and an adaptive neuro-fuzzy inference system (ANFIS). This toolkit is a continuation from the previous package (‘FuzzyToolkitUoN’). Produced by the Intelligent Modelling & Analysis Group (IMA) and Lab for UnCertainty In Data and decision making (LUCID), University of Nottingham. A big thank you to the many people who have contributed to the development/evaluation of the toolbox. Please cite the toolbox and the corresponding paper <doi:10.1109 FUZZ48607.2020.9177780> when using it. More related papers can be found in the NEWS.
License GPL (>= 2)
URL https://www.lucidresearch.org/
RoxygenNote 7.1.1
NeedsCompilation no
Encoding UTF-8
Repository CRAN
Date/Publication 2021-05-19 09:00:05 UTC
Author Chao Chen [aut, cre],
Jon Garibaldi [aut],
Tajul Razak [aut]
Maintainer Chao Chen <fuzzyr@cs.nott.ac.uk>

R topics documented:

  addmf .................................................. 3
  addrule ............................................... 4

1
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addvar</td>
<td>5</td>
</tr>
<tr>
<td>anfis.builder</td>
<td>6</td>
</tr>
<tr>
<td>anfis.dE.dO1</td>
<td>7</td>
</tr>
<tr>
<td>anfis.dE.dO2</td>
<td>8</td>
</tr>
<tr>
<td>anfis.dE.dO3</td>
<td>8</td>
</tr>
<tr>
<td>anfis.dE.dO4</td>
<td>9</td>
</tr>
<tr>
<td>anfis.dE.dO5</td>
<td>10</td>
</tr>
<tr>
<td>anfis.dE.dP1</td>
<td>10</td>
</tr>
<tr>
<td>anfis.dE.dP1.gbellmf</td>
<td>11</td>
</tr>
<tr>
<td>anfis.dE.dP1.it2gbellmf</td>
<td>12</td>
</tr>
<tr>
<td>anfis.dE.dP4</td>
<td>12</td>
</tr>
<tr>
<td>anfis.dMF.dP.gbellmf</td>
<td>13</td>
</tr>
<tr>
<td>anfis.dO2.dO1</td>
<td>14</td>
</tr>
<tr>
<td>anfis.dO3.dO2</td>
<td>14</td>
</tr>
<tr>
<td>anfis.dO4.dO3</td>
<td>15</td>
</tr>
<tr>
<td>anfis.dO5.dO4</td>
<td>16</td>
</tr>
<tr>
<td>anfis.eval</td>
<td>16</td>
</tr>
<tr>
<td>anfis.L1.eval</td>
<td>17</td>
</tr>
<tr>
<td>anfis.L2.eval</td>
<td>18</td>
</tr>
<tr>
<td>anfis.L2.which</td>
<td>18</td>
</tr>
<tr>
<td>anfis.L3.eval</td>
<td>19</td>
</tr>
<tr>
<td>anfis.L4.eval</td>
<td>20</td>
</tr>
<tr>
<td>anfis.L4.mf.eval</td>
<td>20</td>
</tr>
<tr>
<td>anfis.L5.eval</td>
<td>21</td>
</tr>
<tr>
<td>anfis.L1.eval</td>
<td>22</td>
</tr>
<tr>
<td>anfis.optimise</td>
<td>22</td>
</tr>
<tr>
<td>anfis.plotmf</td>
<td>24</td>
</tr>
<tr>
<td>anfis.tipper</td>
<td>25</td>
</tr>
<tr>
<td>cmp.firing</td>
<td>26</td>
</tr>
<tr>
<td>convertfis</td>
<td>27</td>
</tr>
<tr>
<td>defuzz</td>
<td>28</td>
</tr>
<tr>
<td>evalfis</td>
<td>28</td>
</tr>
<tr>
<td>evalmf</td>
<td>29</td>
</tr>
<tr>
<td>evalmftype</td>
<td>30</td>
</tr>
<tr>
<td>fis.builder</td>
<td>31</td>
</tr>
<tr>
<td>fuzzy.firing</td>
<td>32</td>
</tr>
<tr>
<td>fuzzy.optimise</td>
<td>32</td>
</tr>
<tr>
<td>fuzzy.t</td>
<td>33</td>
</tr>
<tr>
<td>fuzzy.tconorm</td>
<td>34</td>
</tr>
<tr>
<td>fuzzy.tnorm</td>
<td>35</td>
</tr>
<tr>
<td>fuzzyr.accuracy</td>
<td>35</td>
</tr>
<tr>
<td>fuzzyr.match.fun</td>
<td>36</td>
</tr>
<tr>
<td>gbell.fuzzification</td>
<td>37</td>
</tr>
<tr>
<td>gbellmf</td>
<td>37</td>
</tr>
<tr>
<td>genmf</td>
<td>38</td>
</tr>
<tr>
<td>gensurf</td>
<td>40</td>
</tr>
<tr>
<td>it2tipper</td>
<td>40</td>
</tr>
<tr>
<td>km.da</td>
<td>41</td>
</tr>
</tbody>
</table>
addmf

Description

Insert a membership function.

Usage

addmf(fis, varType, varIndex, mfName, mfType, mfParams)

Arguments

fis
A fis structure is to be provided.

varType
Should be either 'input' or 'output', which relates to the type of variable (stored on the existing fis structure) that the membership function will be added to.

varIndex
Should be an integer value representing the index value of the input or output variable that the membership function will be added to (base 1).

mfName
Membership function name to be declared, for example (Poor,Good)

mfType
Membership function type to be declared, for example (trimf, trapmf)

mfParams
The value of membership function.

Value

A fis structure with the new membership function added.
addrule

Inserts a rule

Description

Adds a rule to a fis object.

Usage

addrule(fis, ruleList)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fis</td>
<td>A fis structure is to be provided.</td>
</tr>
<tr>
<td>ruleList</td>
<td>A vector of length ( m + n + 2 ), where ( m ) is the number of input variables of a fis. Each column in ( m ) has a number which refers to the membership function of that input variable. Columns under ( n ) refer to an output variable of a fis, where the value refers to the membership function of that output variable. Finally, the ( 2 ) remaining columns refer to the weight to be applied to the rule (( m + n + 1 )) and the fuzzy operator for the rule’s antecedent (1 = AND, 2 = OR).</td>
</tr>
</tbody>
</table>

Details

For example, if one has a fis with 2 input variables, and 1 output variable, each of which have 3 membership functions (the amount of membership functions need not be the same). The following rule: 1 3 2 1 2 will mean \( m = 2 \) (for 2 input variables), \( n = 1 \) (for 1 output variable), and the last 2 columns represent weight and fuzzy operator for the rule’s antecedent respectively.

The first column refers to the first input variable’s membership function at index 1.

The second column refers to the second input variable’s membership function at index 3.

The third column refers to the first output variable’s membership function at index 2.

The fourth column refers to the weight to be applied to the rule.

The fifth column refers to the fuzzy operator for the rule’s antecedent (in this case it represents ‘OR’).

Value

A fis structure with the new rule added.
Examples

```r
fis <- tipper()
ruleList <- rbind(c(1,1,1,1,2), c(2,0,2,1,1), c(3,2,3,1,2))
fis <- addrule(fis, ruleList)
```

addvar

**Description**

Adds an input or output variable to a fis object.

**Usage**

```r
addvar(
  fis, 
  varType, 
  varName, 
  varBounds, 
  method = NULL, 
  params = NULL, 
  firing.method = "tnorm.min.max"
)
```

**Arguments**

- **fis**: A fis must be provided.
- **varType**: Should be either 'input' or 'output' which represents the type of variable to be created and added.
- **varName**: A string representing the name of the variable.
- **varBounds**: Also known as the 'range', this should be a vector giving a range for the variable, such as 1:10.
- **method**: fuzzification or defuzzification method.
  - fuzzification: 'gauss', 'gbell', 'tri', or user-defined.
  - defuzzification: 'centroid', 'cos', 'coh', 'csum' or user-defined.
- **params**: the required parameters for the corresponding fuzzification or defuzzification method. For example, the required parameters for `gbell.fuzzification` are c(a,b)
- **firing.method**: the chosen method for getting the firing strength (for non-singleton fuzzification).
  - 'tnorm.min.max' - minimum t-norm with maximum membership grade as the firing strength
  - 'tnorm.prod.max' - product t-norm with maximum membership grade as the firing strength
- `tnorm.min.defuzz.[method]` - the firing strength is based on minimum t-norm, and the chosen defuzzification method (e.g. tnorm.min.defuzz.centroid)
- `tnorm.prod.defuzz.[method]` - the firing strength is based on product t-norm, and the chosen defuzzification method (e.g. tnorm.prod.defuzz.bisector)
- 'similarity.set' - Set-theoretic similarity: the ratio between the intersection and the union of two fuzzy sets

**Value**

A fis with the new variable added.

**Examples**

```r
fis <- newfis('tipper')
fis <- addvar(fis, 'input', 'service', c(0, 10))
fis <- addvar(fis, 'input', 'service', c(0, 10), 'gauss', 0.5, 'tnorm.min.max')
```

**Description**

To build an ANFIS model from an existing FIS model

**Usage**

`anfis.builder(fis)`

**Arguments**

- `fis` A fuzzy inference system model initialised by `newfis`.

**Value**

An ANFIS model

**Author(s)**

Chao Chen

**References**


Examples

```r
fis <- anfis.tipper()
anfis <- anfis.builder(fis)
```

Description

To calculate the derivatives of output error with respect to output.L1.

Usage

```r
anfis.dE.dO1(anfis, output.L1, de.do2, do2.do1)
```

Arguments

- `anfis`: The given ANFIS model
- `output.L1`: The output of nodes in Layer 1
- `de.do2`: The derivatives of output error with respect to output.L2

Details

This function is not recommended for external use, but can be used for debugging or learning.

Value

The derivatives of output error with respect to output.L1.

Author(s)

Chao Chen
**Description**

to calculate the derivatives of output error with respect to output.L2.

**Usage**

```
anfis.dE.dO2(de.do3, do3.do2)
```

**Arguments**

- `de.do3`: The derivatives of output error with respect to output.L3

**Details**

This function is not recommended for external use, but can be used for debugging or learning.

**Value**

The derivatives of output error with respect to output.L2.

**Author(s)**

Chao Chen

---

**Description**

to calculate the derivatives of output error with respect to output.L3.

**Usage**

```
anfis.dE.dO3(de.do4, do4.do3, output.L3)
```

**Arguments**

- `de.do4`: The derivatives of output error with respect to output.L4
Details

This function is not recommended for external use, but can be used for debugging or learning.

Value

The derivatives of output error with respect to output.L3.

Author(s)

Chao Chen

Description

to calculate the derivatives of output error with respect to output.L4.

Usage

anfis.dE.dO4(anfis, de.do5, do5.do4)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>anfis</td>
<td>The given ANFIS model</td>
</tr>
<tr>
<td>de.do5</td>
<td>The derivatives of output error with respect to output.L5</td>
</tr>
<tr>
<td>do5.do4</td>
<td>The derivatives of output.L5 with respect to output.L4.</td>
</tr>
</tbody>
</table>

Details

This function is not recommended for external use, but can be used for debugging or learning.

Value

The derivatives of output error with respect to output.L4.

Author(s)

Chao Chen
**Description**

To calculate the derivatives of output error with respect to output.L5. NOTE: currently, only single output in L5 is supported.

**Usage**

`anfis.dE.dO5(output.L5, y)`

**Arguments**

- **output.L5**: the model outputs
- **y**: the target outputs

**Details**

This function is not recommended for external use, but can be used for debugging or learning.

**Value**

The derivatives of output error with respect to output.L5

**Author(s)**

Chao Chen

---

**Description**

To calculate the derivatives of output error with respect to parameters in Layer 1.

**Usage**

`anfis.dE.dP1(anfis, de.do1, input.stack)`

**Arguments**

- **anfis**: The given ANFIS model
- **de.do1**: The derivatives of output error with respect to output.L1
- **input.stack**: The input data pairs.
Details

This function is not recommended for external use, but can be used for debugging or learning.

Value

The derivatives of output error with respect to parameters in Layer 1.

Author(s)

Chao Chen

Description

To calculate the derivatives of E versus mf.params.L1 for gbellmf: 1 / (1 + ((x - c)/a)^2)^b) NOTE: only singleton fuzzification is supported

Usage

anfis.dE.dPL1.gbellmf(de.do1, x, mf.params)

Arguments

de.do1  The derivatives of output error with respect to output.L1
x  The crisp input
mf.params  parameters for membership functions

Details

This function is not recommended for external use, but can be used for debugging or learning.

Author(s)

Chao Chen
Description

to calculate the derivatives of $E$ versus $\text{mf.params.L1}$ for $\text{it2gbellmf}$ NOTE: only singleton fuzzification is supported

Usage

\texttt{anfis.dE.dP1.it2gbellmf(de.do1, x, mf.params)}

Arguments

\begin{itemize}
\item \texttt{de.do1} \hspace{1em} The derivatives of output error with respect to output.L1
\item \texttt{x} \hspace{1em} The crisp input
\item \texttt{mf.params} \hspace{1em} parameters for membership functions
\end{itemize}

Details

This function is not recommended for external use, but can be used for debugging or learning.

Author(s)

Chao Chen

Description

To calculate the derivatives of output error with respect to parameters in Layer 4.

Usage

\texttt{anfis.dE.dP4(anfis, de.do4, output.L3, input.stack)}

Arguments

\begin{itemize}
\item \texttt{anfis} \hspace{1em} The given ANFIS model
\item \texttt{de.do4} \hspace{1em} The derivatives of output error with respect to output.L4
\item \texttt{output.L3} \hspace{1em} The output of nodes in Layer 3
\item \texttt{input.stack} \hspace{1em} The input data pairs.
\end{itemize}
Details

This function is not recommended for external use, but can be used for debugging or learning.

Value

The derivatives of output error with respect to parameters in Layer 4.

Author(s)

Chao Chen

Description

to calculate the derivatives of membership grades with respect to its parameters

Usage

`anfis.dMF.dP.gbellmf(x, mf.params)`

Arguments

- `x` The crisp input
- `mf.params` parameters for membership functions

Details

This function is not recommended for external use, but can be used for debugging or learning.

Author(s)

Chao Chen
anfis.dO2.dO1

Description
To calculate the derivatives of output.L2 with respect to output.L1.

Usage
anfis.dO2.dO1(anfis, output.L2, output.L1)

Arguments
- anfis: The given ANFIS model
- output.L2: The output of nodes in Layer 2
- output.L1: The output of nodes in Layer 1

Details
This function is not recommended for external use, but can be used for debugging or learning.

Value
The derivatives of output.L2 with respect to output.L1.

Author(s)
Chao Chen

anfis.dO3.dO2

Description
To calculate the derivatives of output.L3 with respect to output.L2.

Usage
anfis.dO3.dO2(anfis, output.L2, output.L2.which)

Arguments
- anfis: The given ANFIS model
- output.L2: The output of nodes in Layer 2
- output.L2.which: A list of matrix indicating which output (w.lower, w.upper) in layer 2 should be used by the ekm algorithm
Details

This function is not recommended for external use, but can be used for debugging or learning.

Value

The derivatives of output.L3 with respect to output.L2. \( \text{do3.left}[j].\text{do2}[i] \leftarrow \text{do3.d2}[[i]][[1]][[j]] \)

Author(s)

Chao Chen

Description

To calculate the derivatives of output.L4 with respect to output.L3.

Usage

\texttt{anfis.d4.d3(output.L4, output.L4.mf)}

Arguments

- \texttt{output.L4} The output of nodes in Layer 4
- \texttt{output.L4.mf} The membership grades of the membership functions of nodes in Layer 4

Details

This function is not recommended for external use, but can be used for debugging or learning.

Value

The derivatives of output.L4 with respect to output.L3.

Author(s)

Chao Chen
Description
To calculate the derivatives of output.L5 with respect to output.L4. NOTE: currently, only single output in L5 is supported.

Usage
anfis.dO5.dO4(output.L4)

Arguments
output.L4 The output of nodes in Layer 4.

Details
This function is not recommended for external use, but can be used for debugging or learning.

Value
The derivatives of output.L5 with respect to output.L4.

Author(s)
Chao Chen

Description
To evaluate an ANFIS model with input data.

Usage
anfis.eval(anfis, input.stack)

Arguments
anfis The given ANFIS model
input.stack The input data

Value
The output of the anfis for given input data.
Author(s)
Chao Chen

Examples
fis <- anfis.tipper()
anfis <- anfis.builder(fis)
data.num <- 5
input.num <- length(fis$input)
input.stack <- matrix(rnorm(data.num*input.num), ncol=input.num)
y <- matrix(rnorm(data.num))
data.trn <- cbind(input.stack, y)
anfis.eval(anfis, input.stack)

anfis.L1.eval  The evaluator for nodes in Layer 1

Description
To evaluate the antecedent layer (L1) of anfis

Usage
anfis.L1.eval(anfis, output.LI, input.stack)

Arguments
anfis The given ANFIS model
output.LI The output of nodes in Layer I
input.stack The input data

Details
This function is not recommended for external use, but can be used for debugging or learning. See the source code of anfis.eval for usage.

Value
The output of nodes in Layer I

Author(s)
Chao Chen
**Description**

To evaluate the nodes in Layer 2 of the given ANFIS model

**Usage**

```r
anfis.L2.eval(anfis, output.L1)
```

**Arguments**

- `anfis`: The given ANFIS model
- `output.L1`: The output of nodes in Layer 1

**Details**

This function is not recommended for external use, but can be used for debugging or learning. See the source code of `anfis.eval` for usage.

**Value**

The output of nodes in Layer 2

**Author(s)**

Chao Chen

---

**Description**

To determin which output (w.lower, w.upper) to be used by the ekm algorithm

**Usage**

```r
anfis.L2.which(anfis, output.L2, output.L4.mf)
```

**Arguments**

- `anfis`: The given ANFIS model
- `output.L2`: The output of nodes in Layer 2
- `output.L4.mf`: The linear membership grades of nodes in Layer 4
anfis.L3.eval

Details
This function is not recommended for external use, but can be used for debugging or learning. See the source code of anfis.eval for usage.

Value
A list of matrix indicating which output (w.lower, w.upper) in layer 2 should be used by the ekm algorithm

Author(s)
Chao Chen
The evaluator for nodes in Layer 4

Description
To evaluate the nodes in Layer 4

Usage

Arguments
- **output.L3**: The output of nodes in Layer 3
- **output.L4.mf**: The membership grades of the membership functions of nodes in Layer 4

Details
This function is not recommended for external use, but can be used for debugging or learning. See the source code of **anfis.eval** for usage.

Value
The output of nodes in Layer 4

Author(s)
Chao Chen

The evaluator for membership functions of nodes in Layer 1

Description
To evaluate the membership functions of nodes in Layer 4

Usage
anfis.L4.mf.eval(anfis, input.stack)

Arguments
- **anfis**: The given ANFIS model
- **input.stack**: The input data
Details

This function is not recommended for external use, but can be used for debugging or learning. See the source code of `anfis.eval` for usage.

Value

The membership grades of the membership functions of nodes in Layer 4

Author(s)

Chao Chen

---

`anfis.L5.eval`  
*The evaluator for nodes in Layer 5*

Description

To evaluate the nodes in Layer 5

Usage

`anfis.L5.eval(output.L4)`

Arguments

- `output.L4`  
  The output of nodes in Layer 4

Details

This function is not recommended for external use, but can be used for debugging or learning. See the source code of `anfis.eval` for usage.

Value

The output of nodes in Layer 5

Author(s)

Chao Chen
anfis.LI.eval

*The evaluator for nodes in Layer I*

**Description**
To evaluate the input Layer (LI) of anfis

**Usage**
anfis.LI.eval(anfis, input.stack)

**Arguments**
anfis The given ANFIS model
input.stack The input data

**Details**
This function is not recommended for external use, but can be used for debugging or learning. See the source code of `anfis.eval` for usage.

**Value**
The output of nodes in Layer I

**Author(s)**
Chao Chen

anfis.optimise

*ANFIS optimiser*

**Description**
To optimise the performance of a given ANFIS model by learning the parameters in L1 and L4.

**Usage**
anfis.optimise(
anfis,
data.trn,
data.chk = NULL,
epoch.total = 100,
stepsize = 0.1,
rate.inc = 1.1,
rate.dec = 0.9,
)
method = c("gradient", "lse"),
err.log = F,
online = 0,
lambda = 1,
opt.by = "err.opt",
err.trn.fix = T
)

Arguments

anfis The given ANFIS model
data.trn The input and output data pairs as training data
data.chk The input and output data pairs as checking (validation) data
epoch.total The total training epochs.
stepsize The initial stepsize
rate.inc increasing rate of the stepsize
rate.dec decreasing rate of the stepsize
method The learning algorithms for Layer 1 and Layer 4 respectively. default method = c("gradient", "lse")
err.log T or F, the flag indicate whether to save the error log.
online 0 – batch; 1 – online; 2 – semi-online
lambda The forgetting rate for the LSE algorithm
opt.by To optimise the ANFIS model by: err.opt – optimisation error; err.trn – training error; err.chk – checking (validation) error.
err.trn.fix T or F. When KM defuzzification is used for IT2 ANFIS, err.trn is not equal to err.opt. Hence, this flag is used for users to choose whether to fix this issue. The default value is set to T for the compatibility with previous built IT2 models. For T1 ANFIS, this flag can be set to F for speed improvement.

Value

The optimised ANFIS model.

Author(s)

Chao Chen

References


Examples

```r
fis <- anfis.tipper()
anfis <- anfis.builder(fis)
data.num <- 5
input.num <- length(fis$input)
input.stack <- matrix(rnorm(data.num*input.num), ncol=input.num)
y <- matrix(rnorm(data.num))
data.trn <- cbind(input.stack, y)
anfis.eval(anfis, input.stack)
anfis.final <- anfis.optimise(anfis, data.trn, epoch.total=500,
    stepsize=0.01, rate.inc=1.1, rate.dec=0.9)
```

---

**anfis.plotmf**  
Plot membership functions for an ANFIS object

**Description**

Plots a 2D graph of all membership functions from the specified variable which must be part of an anfis object.

**Usage**

```r
anfis.plotmf(  
anfis,
  varType,
  varIndex,  
  xx = NULL,
  timelimit = 0,
  xlab = NULL,
  ylab = NULL,
  main = NULL
)
```

**Arguments**

- **anfis**  
  Requires an existing anfis as an argument.

- **varType**  
  Can be either 'input' or 'output', representing the type of variable.

- **varIndex**  
  A numerical integer, representing the index of the input or output variable whose membership functions shall be plotted (base 1).

- **xx**  
  primary inputs for extra lines

- **timelimit**  
  for perturbation

- **xlab**  
  X axis label using font, size and color

- **ylab**  
  Y axis label, same font attributes as xlab

- **main**  
  The main title (on top)
**anfis.tipper**

**Value**

A two dimensional graph displaying all the membership functions of a given variable.

**Examples**

```r
fis <- anfis.tipper()
anfis <- anfis.builder(fis)
data.num <- 5
input.num <- length(fis$input)
input.stack <- matrix(rnorm(data.num*input.num), ncol=input.num)
y <- matrix(rnorm(data.num))
data.trn <- cbind(input.stack, y)
anfis.eval(anfis, input.stack)
anfis.final <- anfis.optimise(anfis, data.trn, epoch.total=500,
                            stepsize=0.01, rate.inc=1.1, rate.dec=0.9)
anfis.plotmf(anfis, 'input', 1)
anfis.plotmf(anfis.final, 'input', 1)
```

---

**anfis.tipper**  
*Produces an example fis object which can be used for ANFIS.*

**Description**

A function used primarily for example purposes, it creates a fis with two input (service & food), output variables (tip) and their membership functions.

**Usage**

```r
anfis.tipper()
```

**Value**

A fis is return

**Examples**

```r
fis <- anfis.tipper()
```
cmp.firing  

*Plot firing strength with different inference method*

**Description**

Plots a 2D graph of the firing strength for an antecedent produced by different inference method

**Usage**

```r
cmp.firing(
  IP,
  mfType,
  mfPara,
  fuzMethod,
  fuzPara,
  SFLS = TRUE,
  STD = TRUE,
  CEN = FALSE,
  SIM = FALSE,
  step = 100,
  fisRange = NULL
)
```

**Arguments**

- **IP**
  A matrix representing the input stack, number of inputs (columns) by number of outputs (rows).

- **mfType**
  The type of fuzzy membership function

- **mfPara**
  The parameters for the given type of membership function

- **fuzMethod**
  The type of fuzzy membership function for non-singleton fuzzification

- **fuzPara**
  The parameters for the given fuz.type of membership function

- **SFLS**
  When TRUE, shows the firing strength produced by SFLS

- **STD**
  When TRUE, shows the firing strength produced by std-NSFLS

- **CEN**
  When TRUE, shows the firing strength produced by cen-NSFLS

- **SIM**
  When TRUE, shows the firing strength produced by sim-NSFLS

- **step**
  For discrete fuzzification

- **fisRange**
  Field of definition, for example, c(1,10)

**Value**

A two dimensional graph displaying all the firing strength produced by given method.
convertfis

Author(s)
Yu Zhao

Examples

cmp.firing(1, 'gaussmf', c(1, 2.5, 1), 'gbell', c(0.4, 2), step=100)

convertfis  

Convert a fis

Description
Convert a fis object from one type to another (e.g. from singleton to non-singleton)

Usage
convertfis(fis, option = "s2n", ...)

Arguments
fis the fis object to be converted
option the convert option.'s2n': singleton to non-singleton
... For 's2n': fuzzification.method, fuzzification.params, firing.method. See details below for more information.

Details
- fuzzification.method, fuzzification.params, firing.method - see addvar

Usage:
1. convertfis(fis, option, mf.params, fuzzification.method, fuzzification.params)
2. convertfis(fis, option, mf.params, fuzzification.method, fuzzification.params, firing.method)

Value
Membership grade(s)

Author(s)
Chao Chen

Examples
fis <- tipper()
fis.ns.1 <- convertfis(fis, option='s2n', fuzzification.method='gauss', fuzzification.params=1)
fis.ns.2 <- convertfis(fis, option='s2n', fuzzification.method='gauss', fuzzification.params=1, firing.method='tnorm.min.max')
defuzz

**Defuzzify a set of values.**

**Description**
Defuzzifies a given set of values using a specified range and defuzzification type producing a crisp value.

**Usage**
defuzz(x, mf, type)

**Arguments**
- **x**: The range to be applied in the function (numeric vector).
- **mf**: The values to be applied in the function (numeric vector).
- **type**: The defuzzification method type, which should be either 'centroid', 'bisector', 'mom', 'som' or 'lom'.

**Value**
Returns a defuzzified crisp value (double).

**Examples**
Crisp_value = defuzz(1:10, c(1.5, 5), "centroid")

---

evalfis

**Evaluate a Fuzzy Inference System (fis)**

**Description**
Returns an evaluated crisp value for a given fis structure.

**Usage**
evalfis(input_stack, fis, time = 1, point_n = 101, draw = FALSE)

**Arguments**
- **input_stack**: A matrix representing the input stack, number of inputs (columns) by number of outputs (rows).
- **fis**: A fis must be provided.
- **time**: default 1
- **point_n**: number of discretised points, default 101
- **draw**: whether to draw, TRUE or FALSE
evalmf

Value

Returns a matrix of evaluated values.

Examples

Input_data <- matrix((1:2),1,2)
fis <- tipper()
evalfis(Input_data, fis)

evalmf  Evaluate fuzzy membership function

Description

To obtain the corresponding membership grade(s) for given crisp input(s) x

Usage

evalmf(...)

Arguments

... For singleton fuzzification: x, mf.type, mf.params; x, mf.
Four additional parameters need to be used for non-singleton fuzzification: fuzzification.method, fuzzification.params, firing.method and input.range. See details below for more information.

Details

- x - the crisp input(s) on the universe of discourse for corresponding antecedent membership function
- mf.type - The type of fuzzy membership function
- mf.params - The parameters for the given type of membership function
- mf - the membership function generated by genmf
- fuzzification.method, fuzzification.params, firing.method and input.range - see addvar

Usage:

1. evalmf(x, mf.type, mf.params)
2. evalmf(x, mf)
3. evalmf(x, mf.type, mf.params, fuzzification.method, fuzzification.params, firing.method, input.range)
4. evalmf(x, mf, fuzzification.method, fuzzification.params, firing.method, input.range)
Value

Membership grade(s)

Author(s)

Chao Chen

Examples

evalmf(5, mf.type=gbellmf, mf.params=c(1,2,3))
evalmf(1:10, mf.type=gbellmf, mf.params=c(1,2,3))
evalmf(1:10, mf.type=gbellmf, mf.params=c(1,2,3), fuzzification.method='gauss',
       fuzzification.params=1, firing.method='t_norm.min.max', input.range=c(0,10))

mf <- genmf('gbellmf', c(1,2,3))
evalmf(5, mf)
evalmf(1:10, mf)
evalmf(1:10, mf, fuzzification.method='gauss', fuzzification.params=1,
       firing.method='t_norm.min.max', input.range=c(0,10))

---

evalmftype  Evaluate fuzzy membership function with membership function type and parameters

Description

To obtain the corresponding membership grade(s) for crisp input(s) x

Usage

evalmftype(x, mf.type, mf.params)

Arguments

x              A generic element of U, which is the universe of discourse for a fuzzy set
mf.type        The member function type
mf.params      The parameters for a member function

Value

Membership grade(s)

Author(s)

Chao Chen
Examples

```r
evalmftype(5, mf.type=gbellmf, mf.params=c(1,2,3))
evalmftype(1:10, mf.type=gbellmf, mf.params=c(1,2,3))
```

Description

To build a one-output TSK FIS by automatically generating the input membership functions and the fuzzy rules

Usage

```r
fis.builder(  
  x.range,  
  input.num,  
  input.mf.num,  
  input.mf.type,  
  rule.num = prod(input.mf.num),  
  rule.which = NULL,  
  defuzzMethod = "default",  
  params.ante,  
  params.conse  
)
```

Arguments

- `x.range`: a vector/matrix as the range of input(s)
- `input.num`: the number of inputs
- `input.mf.num`: a list of the number of membership functions for all inputs
- `input.mf.type`: designed for different membership function types, however, currently, ’T1’ for gbellmf, else ’it2gbellmf’
- `rule.num`: the number of rules
- `rule.which`: selected rules to be used in the full rule list, for example, c(1,2,3) specify the first three rules
- `defuzzMethod`: "default"
- `params.ante`: parameter settings for initialising antecedent membership functions
- `params.conse`: parameter settings for initialising consequent membership functions

Author(s)

Chao Chen
Description

To get the firing strength for the given input fuzzification membership function and the antecedent membership function in the domain of [lower, upper]

Usage

fuzzy.firing(operator, x.mf, ante.mf, lower, upper)

Arguments

- operator: t-norm operator
- x.mf: the fuzzy input membership function
- ante.mf: the antecedent membership function
- lower: lower bound of the input
- upper: upper bound of the input

Value

the rule firing strength

Author(s)

Chao Chen

Examples

x.mf <- x.fuzzification(gbell.fuzzification, 3, c(1,2))
ante.mf <- genmf(gbellmf, c(1,2,6))
firing.strength <- fuzzy.firing(min, x.mf, ante.mf, lower=0, upper=10)
firing.strength

Description

Fuzzy optimisation

to get an approximation of the maximum membership grade for a given membership function in the domain of [lower, upper]

Usage

fuzzy.optimise(fuzzy.mf, lower, upper)
fuzzy.t

Arguments

fuzzy.mf  fuzzy member function
lower     lower bound of the input
upper     upper bound of the input

Value

an approximation of the maximum membership grade in the given domain

Author(s)

Chao Chen

Examples

mf <- genmf(gbellmf, c(1,2,3))
x <- seq(4, 5, by=0.01)
max(evalmf(x, mf))
fuzzy.optimise(mf, 4, 5)

Description

To conduct t-norm or t-conorm operation for given fuzzy member functions

Usage

fuzzy.t(operator, ...)

Arguments

operator  The supported t-norm/t-conorm operators are min, prod, max
...       fuzzy membership functions

Value

A membership function, which is the t-norm/t-conorm of membership functions

Author(s)

Chao Chen
Examples

mf1 <- genmf(gbellmf, c(1,2,3))
mf2 <- genmf(gbellmf, c(4,5,6))
mf3 <- fuzzy.t(max, mf1, mf2)
tmp1 <- evalmf(1:10, mf1)
tmp2 <- evalmf(1:10, mf2)
tmp3 <- evalmf(1:10, mf3)
identical(tmp3, pmax(tmp1, tmp2))
tmp3

fuzzy.tconorm

Fuzzy t-conorm

Description

To conduct t-conorm operation for given fuzzy member functions

Usage

fuzzy.tconorm(operator, ...)

Arguments

operator The t-conorm operator such as max
...

fuzzy membership functions

Value

A membership function, which is the t-conorm of membership functions

Author(s)

Chao Chen

Examples

mf1 <- genmf(gbellmf, c(1,2,3))
mf2 <- genmf(gbellmf, c(4,5,6))
mf3 <- fuzzy.tconorm(max, mf1, mf2)
tmp1 <- evalmf(1:10, mf1)
tmp2 <- evalmf(1:10, mf2)
tmp3 <- evalmf(1:10, mf3)
identical(tmp3, pmax(tmp1, tmp2))
tmp3
fuzzy.tnorm

Description
To conduct t-norm operation for given fuzzy member functions

Usage
fuzzy.tnorm(operator, ...)

Arguments
operator
The t-norm operator such as min, prod
...
 fuzzy membership functions

Value
A membership function, which is the t-norm of membership functions

Author(s)
Chao Chen

Examples
mf1 <- genmf(gbellmf, c(1,2,3))
mf2 <- genmf(gbellmf, c(4,5,6))
mf3 <- fuzzy.tnorm(prod, mf1, mf2)
tmp1 <- evalmf(1:10, mf1)
tmp2 <- evalmf(1:10, mf2)
tmp3 <- evalmf(1:10, mf3)
identical(tmp3, tmp1*tmp2)
tmp3

fuzzyr.accuracy

Description
This function is to provide performance indicators by using eight different accuracy measures including a new measure UMBRAE.

Usage
fuzzyr.accuracy(f, y, f.ref = 0, scale.mase = NULL)
Arguments

\begin{verbatim}
f  A vector of forecasting values produced by a model to be evaluated.
y  A vector of observed values.
f.ref  A vector of forecasting values produced by a benchmark method to be compared.
scale.mase  A single value which is the scaling factor of the measure MASE.
\end{verbatim}

Value

A vector of results by each measure.

Author(s)

Chao Chen

References


Examples

f <- rnorm(10)
y <- rnorm(10)
fuzzyr.accuracy(f, y)

Description

This is a modification of the original match.fun, where parent.frame(2) is changed to parent.env(environment()).

Usage

fuzzyr.match.fun(FUN, descend = TRUE)

Arguments

\begin{verbatim}
FUN  item to match as function: a function, symbol or character string.
descend  logical; control whether to search past non-function objects.
\end{verbatim}

Details

See match.fun.
gbell.fuzzification  Generalised bell fuzzification

Description
To generate a fuzzy membership function based on generalised bell fuzzification for the given crisp input x

Usage
gbell.fuzzification(x, mf.params)

Arguments
x  the crisp input, which will be the parameter c for a generalised bell membership function
mf.params  the parameters c(a, b) or c(a, b, h) for a generalised bell membership function

Value
The gbell MF centred at the crisp point x

Author(s)
Chao Chen

Examples
mf <- gbell.fuzzification(3, c(1,2))
# This is the same as:
mf <- genmf('gbellmf', c(1,2,3))
evalmf(1:10, mf)

gbellmf  Generalised bell membership function

Description
To specify a generalised bell membership function with a pair of particular parameters

Usage
gbellmf(mf.params)
Arguments

mf.params The parameters c(a, b, c) for a generalised bell membership function

Details

This is not an external function. It should be used through genmf.

Value

The generalised bell membership function of x for a given pair of parameters, where x is a generic element of U, which is the universe of discourse of a fuzzy set X

Author(s)

Chao Chen

Examples

mf <- gbellmf(c(1,2,3))
# This is the same as:
mf <- genmf('gbellmf', c(1,2,3))

evalmf(5, mf)
**genmf**

**Details**

Built-in membership function types are: 'gbellmf', 'it2gbellmf', 'singletonmf', 'linearmf', 'gaussmf', 'trapmf', 'trimf'.

mf.params for

- 'gbellmf' is c(a, b, c), where a denotes the width, b is usually positive and c locates the center of the curve.
- 'it2gbellmf' is c(a.lower, a.upper, b, c), where a.upper > a.lower when b > 0 and a.upper < a.lower when b < 0
- 'singletonmf' is c(c), where c is the location where the membership grade is 1.
- 'linearmf' is c(...), which are the coefficients of the linear membership function.
- 'gaussmf' is c(sig, c), which are the parameters for exp(-(x - c)^2/(2 * sig^2)).
- 'trapmf' is c(a, b, c, d), where a and d locate the "feet" of the trapezoid and b and c locate the "shoulders".
- 'trimf' is c(a, b, c), where a and c locate the "feet" of the triangle and b locates the peak.

Note that users are able to define their own membership functions.

**Value**

The desired type of membership function f(x), where x is a generic element of U, which is the universe of discourse for a fuzzy set

**Author(s)**

Chao Chen

**Examples**

```r
mf <- genmf('gbellmf', c(1,2,3))
evalmf(1:10, mf)
```
**gensurf** Produce a graphical evaluated fuzzy inference system.

**Description**

Produces a three dimensional graphical view of a specific fis object. This function is only works for FIS structures with 3 variables. It will only work for 2 inputs, and 1 output.

**Usage**

gensurf(fis, ix1 = 1, ix2 = 2, ox1 = 1)

**Arguments**

- **fis**
  - A fis must be provided.
- **ix1**
  - Optional input (1)
- **ix2**
  - Optional input (2)
- **ox1**
  - Optional output

**Value**

A three dimensional graphical model generated from the fis and other optional parameters.

**Examples**

```r
fis <- tipper()
gensurf(fis)
```

**it2tipper** Produces an example it2fis object for Waiter-Tipping.

**Description**

A function used primarily for example purposes, it creates a it2 fis with two input (service & food), output variables (tip) and their membership functions.

**Usage**

```r
it2tipper()
```

**Value**

A fis object

**Examples**

```r
it2fis <- it2tipper()
```
Description

A Direct Approach for Determining the Switch Points in the Karnik-Mendel Algorithm.

Usage

\[
\text{km.da}(wl, \text{wr}, f, \text{maximum} = \text{T}, \text{w.wh..ich} = \text{T}, \text{sorted} = \text{T}, \text{k..which} = \text{T})
\]

Arguments

- \text{wl} A vector of lower membership grades.
- \text{wr} A vector of upper membership grades.
- \text{f} A vector of the primary values in the discrete universe of discourse \( X \).
- \text{maximum} T, to calculate the maximum centroid; F, to calculate the minimum centroid.
- \text{w.wh..ich} T, to show which membership grade to be used to calculate maximum/minimum centroid for each primary value.
- \text{sorted} T, to indicate that the primary values have already been put in ascending order.
- \text{k..which} T, to show the index of the switch point selected by the algorithm.

Value

- w.wh..ich=T, a two-column matrix indicating which membership grades to be used; w.wh..ich=F and k..which=T, a vector of the centroid and the switch point; w.wh..ich=F and k..which=F, a single value of the centroid.

Author(s)

Chao Chen

References


Examples

```r
wr <- runif(100, 0, 1)
w1 <- wr * runif(100, 0, 1)
f <- abs(runif(100, 0, 1))
f <- sort(f)
km.da(wl, wr, f)
```

---

**linearmf**

*Linear membership function*

**Description**

To specify a 1st order linear membership function with given parameters

**Usage**

```r
linearmf(mf.params)
```

**Arguments**

- `mf.params`: The linear parameters, which is a vector of the size of input numbers plus 1

**Value**

A linear membership function

**Author(s)**

Chao Chen

---

**newfis**

*Create a fis using newfis function*

**Description**

Creates a fis object.

**Usage**

```r
newfis(
  fisName,
  fisType = "mamdani",
  mfType = "t1",
  andMethod = "min",
  orMethod = "max",
  impMethod = "min",
  aggMethod = "max",
  defuzzMethod = "centroid"
)
```
Arguments

- `fisName` String representing the fis name.
- `fisType` Type of the fis, default is 'mamdani'.
- `mfType` Type of membership functions, 't1' or 'it2'
- `andMethod` The AND method for the fis, default is 'min'.
- `orMethod` The OR method for the fis, default is 'max'.
- `impMethod` The implication method for the fis, default is 'min'.
- `aggMethod` The aggregation method for the fis, default is 'max'.
- `defuzzMethod` The defuzzification method for the fis, default is 'centroid'.

Value

A new fis structure.

Examples

```r
fis <- newfis("fisName")
```

Description

Plots a 2D graph of all membership functions from the specified variable which must be part of a fis object.

Usage

```r
plotmf(
  fis,
  varType,
  varIndex,
  xx = NULL,
  timelimit = 0,
  xlab = NULL,
  ylab = NULL,
  main = NULL
)
```
Arguments

- **fis**
  Requires an existing fis as an argument.

- **varType**
  Can be either 'input' or 'output', representing the type of variable.

- **varIndex**
  A numerical integer, representing the index of the input or output variable whose membership functions shall be plotted (base 1).

- **xx**
  Primary inputs for extra lines

- **timelimit**
  For perturbation

- **xlab**
  X axis label using font, size and color

- **ylab**
  Y axis label, same font attributes as xlab

- **main**
  The main title (on top)

Value

A two dimensional graph displaying all the membership functions of a given variable.

Examples

```r
fis <- tipper()
plotmf(fis, "input", 1)
```

---

**Description**

Reads a fis object from a file with the .fis extension, and converts it into a data structure to be used within the environment.

**Usage**

```r
readfis(fileName)
```

**Arguments**

- **fileName**
  Should be an absolute path given as a string to the file to be read, with escaped backslashes.

**Value**

A fis structure with its values generated from that of the files.
showfis  

*Show a fis object.*

**Description**

Shows a fis and all its data in an ordered format on the console.

**Usage**

```r
showfis(fis)
```

**Arguments**

- `fis`  
  Requires a fis structure to be displayed.

**Value**

Returned the organised text regarding the fis is output to console.

**Examples**

```r
fis <- tipper()
showfis(fis)
```

---

showGUI  

*Show a Graphic User Interface of fis object*

**Description**

Show a Graphic User Interface to display membership function plots for input and output, rules and evaluate the fis.

**Usage**

```r
showGUI(fis, advancedGUI = FALSE)
```

**Arguments**

- `fis`  
  Requires a fis structure to display a GUI.

- `advancedGUI`  
  TRUE/FALSE; if TRUE, an advanced GUI with more features is provided (provided by science@sboldt.com).
showrule

Details
This function is purposed to display all the membership plots and rules of fis object in Graphic User Interface (GUI). It also provide a function to evaluate the fis object.

showGUI(fis) will display the GUI of fis object.

Value
Return the GUI to display membership function for input and output together with rules.

Author(s)
Tajul Razak

Examples
fis <- tipper()
fis <- showGUI(fis)

-------------------------
showrule  Showing rule from fis object
-------------------------

Description
All the rule is showing from fis object

Usage
showrule(fis)

Arguments
fis  A fis must be provided.

Value
Show the total of rules inside fis object

Examples
fis <- tipper()
ruleList <- rbind(c(1,1,1,1,2), c(2,0,2,1,1), c(3,2,3,1,2))
fis <- addrule(fis, ruleList)
showrule(fis)
singleton.fuzzification

Singleton Fuzzification

Description

To generate a fuzzy membership function based on singleton fuzzification for the given crisp input \( x \).

Usage

singleton.fuzzification(x, mf.params = NULL)

Arguments

- **x**: the crisp input
- **mf.params**: NULL or a single value

Value

The singleton MF at the crisp point \( x \)

Author(s)

Chao Chen

Examples

```r
mf <- singleton.fuzzification(3)
evalmf(1:10, mf)
```

singletonmf

Singleton membership function

Description

To specify a singleton membership function at the particular point

Usage

singletonmf(mf.params)

Arguments

- **mf.params**: the particular singleton point
Details

This is not an external function. It should be used through `genmf`.

Value

The singleton membership function of x at the particular point, where x is a generic element of U, which is the universe of discourse of a fuzzy set X.

Author(s)

Chao Chen

Examples

```r
mf <- singletonmf(3)
# This is the same as:
mf <- genmf('singletonmf', 3)

evalmf(1:10, mf)
```

Description

A function used primarily for example purposes, it creates a fis with two input (service & food), output variables (tip) and their membership functions.

Usage

`tipper()`

Value

A fis is return

Examples

```r
fis <- tipper()
```
**tipper.ns**

*Produces an example non-singleton fis object for Waiter-Tipping.*

**Description**

A function used primarily for example purposes, it creates a nsfis with two input (service & food), output variables (tip) and their membership functions.

**Usage**

```r
tipper.ns()
```

**Value**

A non-singleton fis object

**Author(s)**

Yu Zhao

**Examples**

```r
fis <- tipper.ns()
```

---

**tipper.tsk**

*Produces an example fis object (TSK type), which can also be optimised by ANFIS.*

**Description**

A function used primarily for example purposes, it creates a fis with two input (service & food), output variables (tip) and their membership functions.

**Usage**

```r
tipper.tsk()
```

**Value**

A fis is return

**Examples**

```r
fis <- tipper.tsk()
```
tipperGUI

**Description**

Graphic User Interface for Waiter-Tipping to display the membership function (input & output) and rules.

**Usage**

tipperGUI()

**Value**

Return graphic user interface for Waiter-Tipping

**Author(s)**

Tajul Razak

**Examples**

fis <- tipperGUI()

---

tipperGUI2

**Description**

Another style of Graphic User Interface for Waiter-Tipping to display the membership function (input & output) and rules.

**Usage**

tipperGUI2()

**Value**

Return graphic user interface for Waiter-Tipping

**Author(s)**

Tajul Razak

**Examples**

fis <- tipperGUI2()
writefis

Write a fis object to a .fis file.

Description

Write a fis object to a file with the .fis extension.

Usage

writefis(fis, fileName = "fuzzy.fis")

Arguments

fis The fuzzy inference system data structure to be saved.
fileName filename

x.fuzzification Fuzzification

Description

To convert the crisp input x to a fuzzy membership function with specified fuzzification method

Usage

x.fuzzification(fuzzification.method, x, mf.params)

Arguments

fuzzification.method
  The fuzzification method
x
  The required parameters for a fuzzification method
mf.params
  The parameters for a membership function

Value

The corresponding fuzzy membership function

Author(s)

Chao Chen
x <- 3
mf <- x.fuzzification(gbell.fuzzification, x, c(1,2))
# This is the same as:
mf <- genmf(gbellmf, c(1,2,x))

evalmf(1:10, mf)
<table>
<thead>
<tr>
<th>addmf</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>addrule</td>
<td>4</td>
</tr>
<tr>
<td>addvar</td>
<td>5, 27, 29</td>
</tr>
<tr>
<td>anfis.builder</td>
<td>6</td>
</tr>
<tr>
<td>anfis.dE.d01</td>
<td>7</td>
</tr>
<tr>
<td>anfis.dE.d02</td>
<td>8</td>
</tr>
<tr>
<td>anfis.dE.d03</td>
<td>8</td>
</tr>
<tr>
<td>anfis.dE.d04</td>
<td>9</td>
</tr>
<tr>
<td>anfis.dE.d05</td>
<td>10</td>
</tr>
<tr>
<td>anfis.dE.dP1</td>
<td>10</td>
</tr>
<tr>
<td>anfis.dE.dP1.gbEllmf</td>
<td>11</td>
</tr>
<tr>
<td>anfis.dE.dP1.it2gbEllmf</td>
<td>12</td>
</tr>
<tr>
<td>anfis.dE.dP4</td>
<td>12</td>
</tr>
<tr>
<td>anfis.dMf.dP.gbellmf</td>
<td>13</td>
</tr>
<tr>
<td>anfis.dO2.dO1</td>
<td>14</td>
</tr>
<tr>
<td>anfis.dO3.dO2</td>
<td>14</td>
</tr>
<tr>
<td>anfis.dO4.dO3</td>
<td>15</td>
</tr>
<tr>
<td>anfis.dO5.dO4</td>
<td>16</td>
</tr>
<tr>
<td>anfis.eval</td>
<td>16, 17–22</td>
</tr>
<tr>
<td>anfis.L1.eval</td>
<td>17</td>
</tr>
<tr>
<td>anfis.L2.eval</td>
<td>18</td>
</tr>
<tr>
<td>anfis.L2.which</td>
<td>18</td>
</tr>
<tr>
<td>anfis.L3.eval</td>
<td>19</td>
</tr>
<tr>
<td>anfis.L4.eval</td>
<td>20</td>
</tr>
<tr>
<td>anfis.L4.mf.eval</td>
<td>20</td>
</tr>
<tr>
<td>anfis.L5.eval</td>
<td>21</td>
</tr>
<tr>
<td>anfis.LI.eval</td>
<td>22</td>
</tr>
<tr>
<td>anfis.optimise</td>
<td>22</td>
</tr>
<tr>
<td>anfis.plotmf</td>
<td>24</td>
</tr>
<tr>
<td>anfis.tipper</td>
<td>25</td>
</tr>
<tr>
<td>cmp.firing</td>
<td>26</td>
</tr>
<tr>
<td>convertfis</td>
<td>27</td>
</tr>
<tr>
<td>defuzz</td>
<td>28</td>
</tr>
<tr>
<td>evalfis</td>
<td>28</td>
</tr>
<tr>
<td>evalmf</td>
<td>29</td>
</tr>
<tr>
<td>evalmftype</td>
<td>30</td>
</tr>
<tr>
<td>fis.builder</td>
<td>31</td>
</tr>
<tr>
<td>fuzzy.firing</td>
<td>32</td>
</tr>
<tr>
<td>fuzzy.optimise</td>
<td>32</td>
</tr>
<tr>
<td>fuzzy.t</td>
<td>33</td>
</tr>
<tr>
<td>fuzzy.tconorm</td>
<td>34</td>
</tr>
<tr>
<td>fuzzy.tnorm</td>
<td>35</td>
</tr>
<tr>
<td>fuzzyr.accuracy</td>
<td>35</td>
</tr>
<tr>
<td>fuzzyr.match.fun</td>
<td>36</td>
</tr>
<tr>
<td>gbell.fuzzification</td>
<td>5, 37</td>
</tr>
<tr>
<td>gbellmf</td>
<td>37</td>
</tr>
<tr>
<td>genmf</td>
<td>29, 38, 38, 48</td>
</tr>
<tr>
<td>gensurf</td>
<td>40</td>
</tr>
<tr>
<td>it2tipper</td>
<td>40</td>
</tr>
<tr>
<td>km.da</td>
<td>41</td>
</tr>
<tr>
<td>linearmf</td>
<td>42</td>
</tr>
<tr>
<td>match.fun</td>
<td>36</td>
</tr>
<tr>
<td>newfis</td>
<td>6, 42</td>
</tr>
<tr>
<td>plotmf</td>
<td>43</td>
</tr>
<tr>
<td>readfis</td>
<td>44</td>
</tr>
<tr>
<td>showfis</td>
<td>45</td>
</tr>
<tr>
<td>showGUI</td>
<td>45</td>
</tr>
<tr>
<td>showrule</td>
<td>46</td>
</tr>
<tr>
<td>singleton.fuzzification</td>
<td>47</td>
</tr>
<tr>
<td>singletonmf</td>
<td>47</td>
</tr>
<tr>
<td>tipper</td>
<td>48</td>
</tr>
<tr>
<td>tipper.ns</td>
<td>49</td>
</tr>
<tr>
<td>tipper.tsk</td>
<td>49</td>
</tr>
<tr>
<td>tipperGUI</td>
<td>50</td>
</tr>
<tr>
<td>tipperGUI2</td>
<td>50</td>
</tr>
<tr>
<td>writefis</td>
<td>51</td>
</tr>
<tr>
<td>x.fuzzification</td>
<td>51</td>
</tr>
</tbody>
</table>