FuzzyR: An Extended Fuzzy Logic Toolbox for the R Programming Language

Chao Chen[1], Tajul Rosli Razak[1,2], Jonathan M. Garibaldi[1]
[1]Lab for Uncertainty in Data and Decisions Making (LUCID), Intelligent Modelling & Analysis Group (IMA),
School Of Computer Science, The University of Nottingham, UK.
[2]Faculty of Computer & Mathematical Science, Universiti Teknologi MARA,
Perlis Branch, MALAYSIA.
Email: [1]{chao.chen, tajul.razak, jon.garibaldi}@nottingham.ac.uk, [2]tajulrosli@uitm.edu.my

***Abstract*—This paper presents an R package *FuzzyR* which is an extended fuzzy logic toolbox for the R programming language. FuzzyR is a continuation of the previous Fuzzy R toolboxes such as *FuzzyToolkitUoN*. Whilst keeping existing functionalities of the previous toolboxes, the main extension in the *FuzzyR* toolbox is the capability of optimising type-1 and interval type-2 fuzzy inference systems based on an extended ANFIS architecture. An accuracy function is also added to provide performance indicators featuring eight alternative accuracy measures, including a new measure UMBRAE. In addition, graphical user interfaces have been provided so that the properties of a fuzzy inference system can be visualised and manipulated, which is particularly useful for teaching and learning. Note that this paper illustrates some of the new features of the *FuzzyR* toolbox, but does not provide a complete list of all functions available. More details about the new features of *FuzzyR* and a complete description of all functions can be found in the manual of the toolbox.**

I. Introduction

Since its introduction by Zadeh in 1965 [1], fuzzy logic has become more and more popular, both as a specific field of research, as well as a tool for the modelling and computation with data applied in a myriad of disciplines [2, 3]. As mentioned by Wagner [4], while the research interest in fuzzy logic (in particular type-2 fuzzy logic) has steadily grown in the last 10 years, the number of practical applications in real-world use is still limited. There should be no doubt that the provision of free and open-source toolboxes is vital for the wider dissemination, accessibility and implementation of theoretic work and applications on fuzzy logic systems [5, 6].

In the literature, there have been many fuzzy logic toolkits available for type-1, interval type-2 and general type-2 fuzzy logic systems under different programming languages. For example, there are: MATLAB® toolboxes [7, 8, 9]; GUAJE, FisPro, Juzzy and Juzzy Online for Java [10, 11, 12, 13], and Fuzzycreator for Python [6]. On the other hand, R is an ideal choice of language for such a toolbox as it is freely available and used by a wide variety of researchers from a variety of research fields: creating a tool in R will provide a generic infrastructure for a wide range of users and will hopefully lead to an increase in the uptake of fuzzy logic systems (of all flavours). In a previous paper, Wagner et al. [4] presented a fuzzy toolbox for the R programming language. The toolbox was later released as R package called *FuzzyToolkitUoN* [14].

As already described [4], the previous Fuzzy R toolbox can provide functions to create and perform fuzzy inference in Mamdani style. However, the Takagi–Sugeno–Kang (TSK) type fuzzy inference systems, which are also widely used, were not supported. Also, in many applications such as fuzzy logic controllers, effective methods such as ANFIS [15, 16] are widely used to tune the membership functions in order to gain better performance for a pre-defined fuzzy inference system. Such optimisation functions are missing in the previous R toolbox. This paper presents a newly released R package called *FuzzyR* [17], which is an extended version of the previous R fuzzy logic toolbox.

The remainder of this paper is structured as follows. Section II briefly summarises existing features of the previous toolbox. Section III presents the newly added features in *FuzzyR*. Section IV provides some code examples for the newly added features. Finally, conclusions are drawn in Section V.

II. Summary of Previous Toolbox Features

The original fuzzy logic toolbox for the R programming language was presented by Wagner et al. [4]. It was later published in two forms. One is the R package *FuzzyToolkitUoN* [14], which is mainly for type-1 fuzzy logic. The other was an intermediate version in which the toolkit was extended to feature both type-1 and interval type-2. This version was released in source code format, available via a direct web link, but was never released as an official R package. This derived version (*fuzzy-v1_7*) is now superseded by the package described in this paper, and hence is now deprecated.

As mentioned by Wagner et al. [4], the R toolkit has been designed to maximise the transparency for people who are already familiar with the parameters and specification of fuzzy sets and systems while also making it easy to learn for people who are new to the area. Incidentally, the command line interface employed in the R language results in a high similarity compared to the command line interface operation of the MATLAB® Fuzzy Logic Toolbox. The main features of the previous fuzzy logic toolbox for the R programming language can be summarised as follows.

- Create, evaluate and plot type-1 and interval type-2 membership functions. Most of the commonly used types (e.g.

Triangular, Trapezoidal and Gaussian) of membership functions have been defined and ready-to-use. Users can also use customised membership functions.

- Create, export and import Mamdani style fuzzy inference systems with user-specified membership functions, rules and defuzzification methods etc.
- Produce a summary of the constructed fuzzy inference systems showing all the properties such as input and output membership functions, the rule antecedents and consequents.
- Print the rule base of the constructed fuzzy inference systems using IF-THEN sentences which are easy-to-understand.
- Evaluate the given fuzzy inference system with user-specified input values.
- Visualise the control surface of the given fuzzy inference system under the pre-defined domain of inputs.

III. New Toolkit Features

This sections presents the new toolkit features in *FuzzyR*.

A. Redesigned membership functions

In the previous R toolbox, specifying membership functions are actually generating discretised fuzzy sets. By calling a membership function with specified input values x and parameters p, the output will be the membership grades y for the corresponding input x. See Fig. 1 as an example.

```
x <- 1:10
p <- c(2, 5, 8)
y <- trimf(x, p)
y

##  [1] 0.00 0.00 0.33 0.67 1.00 0.67 0.33 0.00 0.00 0.00
```

Fig. 1. Specify a Triangular membership function with the previous R toolbox.

With *FuzzyR*, a membership function can be specified with only the parameters p. Internally, it generates a real function which can later be evaluated with input x values. See Fig. 2 as an example.

```
p <- c(2, 5, 8)
trimf1 <- genmf("trimf", p)
class(trimf1)

## [1] "function"

x <- 1:10
y <- trimf1(x)
y

##  [1] 0.00 0.00 0.33 0.67 1.00 0.67 0.33 0.00 0.00 0.00
```

Fig. 2. Specify and evaluate a Triangular membership function with the *FuzzyR* toolbox.

B. Fuzzy set composition

Based on the newly designed membership functions described above, *FuzzyR* provides methods for the composition of fuzzy sets based on t-norm and t-conorm operators. These methods can generate new membership functions as required (See Fig. 3). This would be quite useful for designing and implementing fuzzy inference systems.

```
p1 <- c(2, 5, 8)
trimf1 <- genmf("trimf", p1)
p2 <- c(3, 6, 9)
trimf2 <- genmf("trimf", p2)
trimf3 <- fuzzy.tnorm("min", trimf1, trimf2)
class(trimf3)

## [1] "function"

x <- 1:10
y <- trimf3(x)
y

##  [1] 0.00 0.00 0.00 0.33 0.67 0.67 0.33 0.00 0.00 0.00
```

Fig. 3. An example of using the t-norm operator with the *FuzzyR* toolbox.

C. TSK type fuzzy inference system

As mentioned above in the Introduction, the Takagi–Sugeno–Kang (TSK) type inference was not supported in the previous toolbox. It has now been added as an option (see below) in *FuzzyR* for both type-1 and interval type-2 fuzzy inference systems. A complete demo for creating a TSK model can be found in the source file of the function *tipper.tsk*.

```
fis = newfis('tipper', fisType = 'tsk')
```

D. ANFIS - TSK type fuzzy system optimisation

The architecture of ANFIS was proposed by Jang [15] to serve as a basis for constructing a set of fuzzy rules with appropriate membership functions to generate the desired fuzzy inference system. A hybrid learning algorithm, based on the gradient method and the least-squares estimate, is applied to tune the parameters of membership functions. The original ANFIS was designed for type-1 TSK models. The implementation of ANFIS in *FuzzyR* is based on an extended ANFIS architecture which can be applied to both type-1 and interval type-2 fuzzy inference systems. There are many options, as shown below, available when using ANFIS to optimise a fuzzy inference system in *FuzzyR*. Details about the usage can found in the manual of the toolbox. Some demonstration code for illustrating the use of ANFIS in *FuzzyR* will also be presented in Section IV.

```
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)
```

E. Type-reduction algorithms

Interval type-2 fuzzy logic systems have been demonstrated to have better abilities to handle uncertainties than their type-1 counterparts in many applications [18, 19, 20, 21, 22, 23, 24]. However, the high computational cost of type-reduction algorithms makes it more expensive to deploy interval type-2 systems, especially for certain cost-sensitive real-world applications. Improving the efficiency of type-reduction algorithms continues to attract research interest. In *FuzzyR*, the mostly commonly used type-reduction algorithms such as EKM [25] and EIASC [26] have been implemented (see Fig. 4). We have also implemented other efficient algorithms such as DA, DAND, COSTRWSR and SC presented in recent studies [27, 28, 29, 30, 31].

```
N <- 2^5
wr <- runif(N, 0, 1)
wl <- wr * runif(N, 0, 1)
x <- abs(rnorm(N, 0, 1))

ekm(wl, wr, x, maximum = T, w.which = F, sorted = F, k.which = F)

## [1] 0.809751

dand(wl, wr, x, maximum = T, w.which = F, sorted = F, k.which = F)

## [1] 0.809751

eiasc(wl, wr, x, maximum = T, w.which = F, sorted = F, k.which = F)

## [1] 0.809751

costrwsr(wl, wr, x, maximum = T, w.which = F)

## [1] 0.809751

sc(wl, wr, x, maximum = T, w.which = F)

## [1] 0.809751
```

Fig. 4. An example of the use of type-reduction functions in *FuzzyR*.

F. Accuracy measures

Without a suitable measure, it is not possible to carry out an objective evaluation of methodologies. Various accuracy measures have been proposed over the past decades as being the best to use in various situations [32]. However, many of these measures are not generally applicable due to issues such as sometimes generating infinite values, or being undefined under certain conditions, which may produce misleading results. In a recent study [33], a new accuracy measure called UMBRAE, which is based on bounded relative error, was proposed. Though the title of the paper may suggest an area of time series forecasting, UMBRAE is actually a good choice for general use as an alternative to other accuracy measures such as RMSE and GMRAE. It combines the best features of various alternative measures without suffering their common issues. Specifically, the new accuracy measure has the following properties:

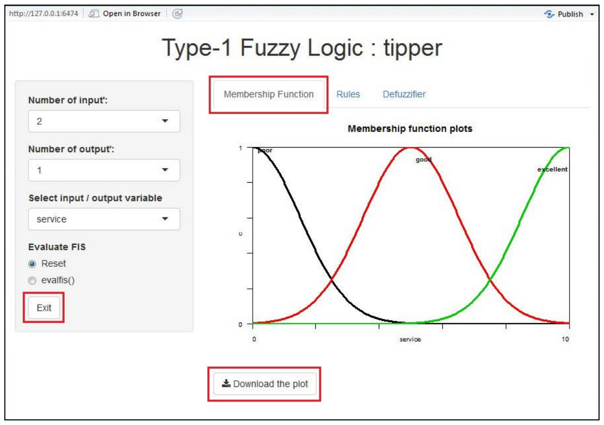
(i) Informative: it can provide an informative result without the need to trim errors.
(ii) Resistant to outliers: it can hardly be dominated by a single forecasting outlier.
(iii) Symmetric: overestimates and underestimates are treated equally.
(iv) Scale-independent: it can be applied to datasets on different scales.
(v) Interpretability: it is easy to understand and can provide intuitive results.

In *FuzzyR*, there is an accuracy function which can provide performance indicators by using eight different accuracy measures including the new measure UMBRAE (see Fig. 5).

```
y <- rnorm(100, sd = 0.1)
x <- rnorm(100, sd = 0.1)
f <- x
f.ref <- 0
fuzzyr.accuracy(f, y, f.ref)

##     MAE    RMSE    MASE    MRAE   GMRAE    MAPE   sMAPE  uMbRAE
##   0.117   0.148      NA   2.389   1.298 238.904 138.207   1.263
```

Fig. 5. An example of the use of accuracy function with the *FuzzyR* toolbox.


http://127.0.0.1:6474

Type-1 Fuzzy Logic : tipper

Number of input': 2

Number of output': 1

Select input / output variable: service

Evaluate FIS: (● Reset) evalfis()

Fig. 6. An example of GUI interface : Main interface

G. Graphical user interface

In the previous toolkit, we use a command-line interface to build the fuzzy logic system which similar to MATLAB®. In order to allow users to construct FLSs easily, a graphic user interface (GUI) is presented, which is also similar to that of MATLAB® Fuzzy Logic Toolbox. Thus, the toolkit will be more user-friendly and more interactive, especially for a new user of *FuzzyR*. This GUI enables the user to view every element of FLS such as membership function (MFs) graph, rules and to evaluate the fuzzy inferences. As part of describing this GUI, we present the Waiter Tipping application, before briefly introducing some of the the visual graphical user interface features.

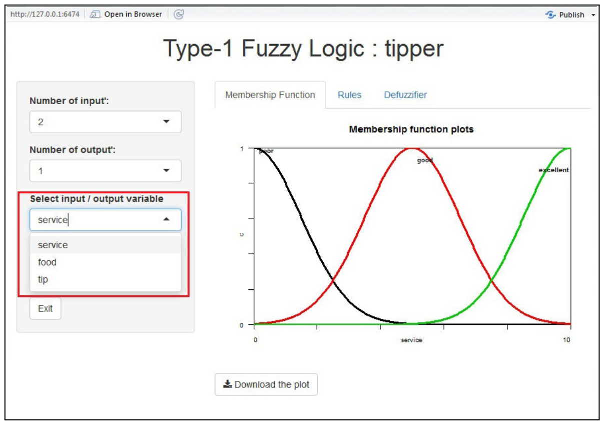
For example, the source code snippet below produces the initial main GUI for tipper example, as shown in Fig. 6. Note that the complete source code of the tipper function can be obtained from the toolbox.

```
fis = tipper();
showGUI(fis);
```

As can be seen in Fig. 6, the main GUI consists of : (i) Menu option namely; *Membership function*, *Rules* and *Defuzzier*. (ii) Selection panes which show the input and output variables and function evaluate fuzzy inference; and (iii) Two buttons for exit and download.

In addition, the details of what can be done using this GUI are described in the following subsystem.

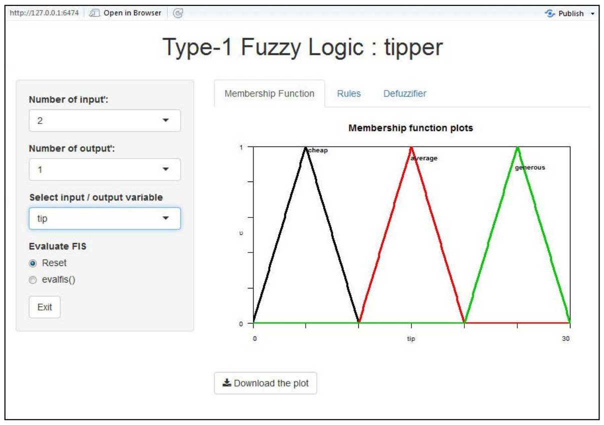
1) Display membership functions plot: This GUI enables the inputs and output of the membership function to be displayed, as shown in Fig. 7. For example, select input/output dropdown menus allow the user to choose which input or output of membership function to be displayed. In the case of Waiter-Tipping example, the user can make a selection based



Type-1 Fuzzy Logic : tipper

Number of input': 2
Number of output': 1
Select input / output variable: service
service
food
tip

Fig. 7. An example of GUI interface : Input and output memberships selection



Type-1 Fuzzy Logic : tipper

Number of input': 2
Number of output': 1
Select input / output variable: tip
Evaluate FIS
[✓ Reset]
evalfis()

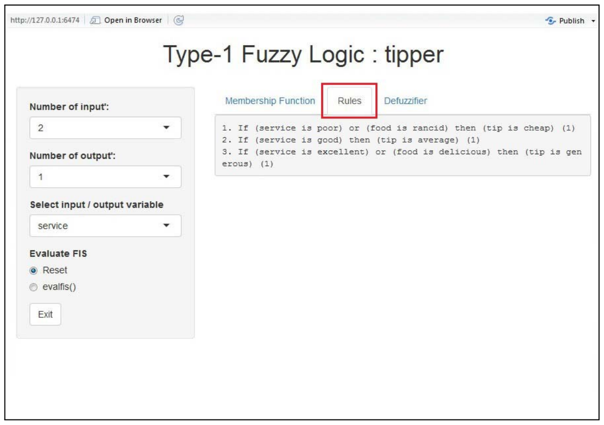
Fig. 8. An example of GUI interface : Output is selected

on service, food and tip, as shown in Fig. 7. Then, Fig. 8 shows the MFs plot that chosen by the user for the Waiter Tipping example. Additionally, download plot button is also provided for the user if they intend to download the plot of that particular MF.

2) View Rule Base of FIS: In the previous toolkit, the user has to write a source code snippet, namely *showrule()*, in order to view all the rules inside the FIS created. Then, as a result, the rules will be shown in the R programming console. However, by using this GUI, it enables the user to graphically view all the rules, by clicking on a button provided, as shown in Fig 9.

Fig 9 shows an example of the rule-based extracted from the FIS of the Waiter-Tipping example.

3) Evaluate Fuzzy Inference System (FIS): The main advantage of this GUI is that it provides the user with an easy to understand visual representation of the fuzzy inference system (FIS). Conversely, in the previous toolkit, users have to write a source code snippet, namely *evalfis()* and also need to provide the values in order to evaluate the FIS object. Then, the toolkit shows the result on the R programming console only. However, in this *FuzzyR*, we have transformed the evaluation FIS function to be more user-friendly and interactive by using the GUI, as shown in Fig. 10.

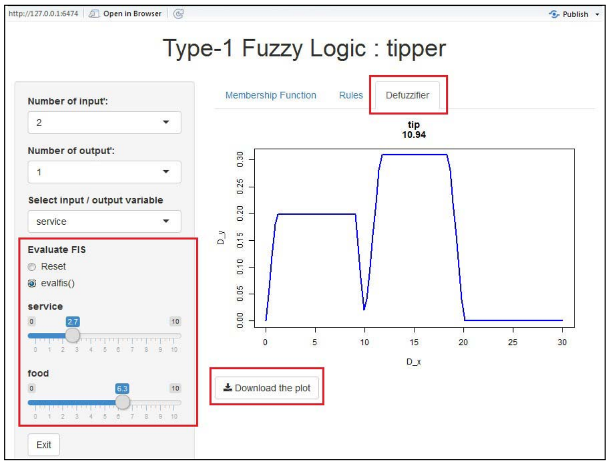


Type-1 Fuzzy Logic : tipper

Number of input': 2
Number of output': 1
Select input / output variable: service
Evaluate FIS
[✓ Reset]
evalfis()

```
1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
2. If (service is good) then (tip is average) (1)
3. If (service is excellent) or (food is delicious) then (tip is gen
erous) (1)
```

Fig. 9. An example of GUI interface : Rules of fis object



Type-1 Fuzzy Logic : tipper

Number of input': 2
Number of output': 1
Select input / output variable: service
Evaluate FIS
Reset
[✓ evalfis()]
service: 2.7
food: 6.3
tip
10.94

Fig. 10. An example of GUI interface : Evaluating FIS

As an example, Fig. 10 shows the FIS of the Waiter Tipping FIS example, which is evaluated using the GUI. In this GUI, the user may insert the values using a slider provided on the left at the menus, as shown in Fig. 10. The number of the slider is generated based on how many input variables that are given into FIS. Then, the results are generated, which are displayed at the centre of this main interface, as shown in Fig. 10. Note that the results and graph automatically change to reflect the values that the user provides using the slider. Likewise, the user also can download the graph by using the download button provided, as shown in Fig. 10.

IV. Code Demo

In this section, we use demo code to illustrate how to use ANFIS with *FuzzyR* for both type-1 and interval type-2 models. The demo models are built based on a commonly used Iris flower data set, which has 150 instances. As shown in Table I, there are five columns in the data set. The first four columns are the features, and the fifth column is the species of Iris (setosa, virginica and versicolor).

Firstly, we need to load the *FuzzyR* package. Then we do the data preprocessing to setup the training, validation and testing

TABLE I
DATA SAMPLES OF THE IRIS FLOWER DATA SET.

Sepal Length	Sepal width	Petal Length	Petal width	Species
5.1	3.5	1.4	0.2	I. setosa
4.9	3.0	1.4	0.2	I. setosa
7.0	3.2	4.7	1.4	I. versicolor
6.3	3.3	6.0	2.5	I. virginica

sets. In this demo, only first two columns are used as inputs and the last column is used as the output.

```
library(FuzzyR)

## First two columns are used
data.all <- iris[, 3:5]
species <- as.factor(data.all[,3])
data.all[,3] <- as.numeric(species)
data.all <- as.matrix(data.all)

## Split the data set into training/validation/
    testing sets.
n.trn <- 120
n.chk <- 15
n.tst <- 15
data.trn <- data.all[1:n.trn,]
data.chk <- data.all[(n.trn+1):(n.trn+n.chk),]
data.tst <- data.all[-(1:(n.trn+n.chk)),]
```

To build the fuzzy inference models, we need to define the number of inputs and membership functions. In this demo, there are two inputs and we define three membership functions for each input.

```
input.num <- 2
input.mf.num <- rep(3, input.num)
```

The rule base can be customised, or selected from the whole rule set. In this case, we use random four rules from the whole rule set.

```
rule.num <- 4
rule.which <- sort(sample(1:prod(input.mf.num),
    rule.num))
```

To use relative accuracy measures such as MRAE, GMRAE and UMBRAE, we need to define the reference benchmark method. In this case, the reference benchmark method is to simply use the values of the second input as the output. A scaling factor needs to be defined for the accuracy measure MASE.

```
naive.trn <- data.trn[, 2]
naive.chk <- data.chk[, 2]
naive.tst <- data.tst[, 2]
scale.mase <- mean(abs(data.trn[, 3] - naive.trn
    ))
```

Note that *FuzzyR* provides a function (*fis.builder*) which can automatically build a fuzzy inference system based on the settings provided (e.g. the domain of inputs, the number of inputs). Below is the code for generating a type-1 fuzzy inference system.

```
x.range <- matrix(0, ncol=2, nrow=input.num)
for(i in 1:input.num) {
  x.range[i,] <- range(data.trn[,1:input.num][,i
    ])
}
fis.t1 <- fis.builder(x.range, input.num, input.
    mf.num, input.mf.type="T1", rule.num=rule.
    num, rule.which=rule.which, defuzzMethod="KM
    ")
```

Before optimising the model by ANFIS, some setttings need to be defined.

```
## optimisation method: alg1 -- the hybrid
    algorithm; alg2 -- pure back propagation
alg1 <- c("gradient", "lse")
alg2 <- c("gradient", "gradient")
algorithm <- alg1

## epochs: the number of training epochs
## stepsize: the length of each gradient
    transition in the parameter space
epochs <- 100
stepsize <- 0.01
## online training flag: 0 -- batch; 1 -- online
    ; 2 -- semi-online
online.flag <- 0
## err.log: T or F, the flag indicate whether to
    save the error log.
err.log <- T
## opt.by: the optimum model will be chosen
    based on smallest error obtained. "err.trn"
    -- trainining error; "err.chk" -- validation
     error.
opt.by <- "err.chk"
```

After building the ANFIS architecture, the fuzzy inference model can be optimised with the function *anfis.optimise*.

```
anfis.t1 <- anfis.builder(fis.t1)
anfis.t1 <- anfis.optimise(anfis.t1, data.trn,
    data.chk, epoch.total=epochs, stepsize=
    stepsize, rate.inc=1.1, rate.dec=0.9, method
    =algorithm, err.log=T, online=online.flag,
    lambda=1, opt.by)
```

The training/validation errors during training epochs are saved in `err.trn` and `err.chk`, which can be used for further analysis such as plotting the error curves (example code provided below at the end of this section).

```
if(err.log) {
  err.trn.log.t1 <- err.trn
  err.chk.log.t1 <- err.chk
}
```

Finally, the optimised model can be used for further evaluations (e.g. to obtain the testing errors).

```
f.tst.t1 <- anfis.eval(anfis.t1, data.tst[,1:
    input.num])
fuzzyr.accuracy(f.tst.t1, data.tst[,-(1:input.
    num)], naive.tst, scale.mase)
```

Code for the interval type-2 model is listed below.

```
fis.it2 <- fis.builder(x.range, input.num, input
    .mf.num, input.mf.type="IT2", rule.num=rule.
    num, rule.which=rule.which, defuzzMethod="KM
    ")

anfis.it2 <- anfis.builder(fis.it2)
anfis.it2 <- anfis.optimise(anfis.it2, data.trn,
     data.chk, epoch.total=epochs, stepsize=
    stepsize, rate.inc=1.1, rate.dec=0.9, method
    =algorithm, err.log=T, online=online.flag,
    lambda=1, opt.by)

if(err.log) {
  err.trn.log.it2 <- err.trn
  err.chk.log.it2 <- err.chk
}

f.tst.it2 <- anfis.eval(anfis.it2, data.tst[,1:
    input.num])
fuzzyr.accuracy(f.tst.it2, data.tst[,-(1:input.
    num)], naive.tst, scale.mase)
```

Below is a code example for plotting the training and validation RMSE error curves, shown in Fig. 11, during the training epochs of type-1 and interval type-2 ANFIS models.

```
require(ggplot2)
require(reshape2)

rmse.trn.t1 <- err.trn.log.t1[,2]
rmse.chk.t1 <- err.chk.log.t1[,2]
rmse.trn.it2 <- err.trn.log.it2[,2]
rmse.chk.it2 <- err.chk.log.it2[,2]

errors <- cbind(rmse.trn.t1, rmse.chk.t1, rmse.
    trn.it2, rmse.chk.it2)

colnames(errors) <- c("training.t1", "validation
    .t1", "training.it2", "validation.it2")
df <- data.frame(errors)
df$epochs <- factor(1:epochs)
df <- melt(df,id.vars="epochs")

cbbPalette <- c(rainbow(15), "#000000")

g <- ggplot(df, aes(x=epochs, value, group=
    variable, shape=variable, fill=variable,
    colour=variable, linetype=variable)) +
    scale_x_discrete(breaks=seq(epochs/10,
        epochs, epochs/10)) +
    geom_line(size=0.8) + geom_point(size=2) +
    scale_shape_manual(values=c(2,6,1,4,3)) +
    labs(x="Epochs", y="RMSE") + ggtitle("") +
    theme_bw() + theme(plot.title = element_text
        (lineheight=.8, face="bold"))
print(g)
```

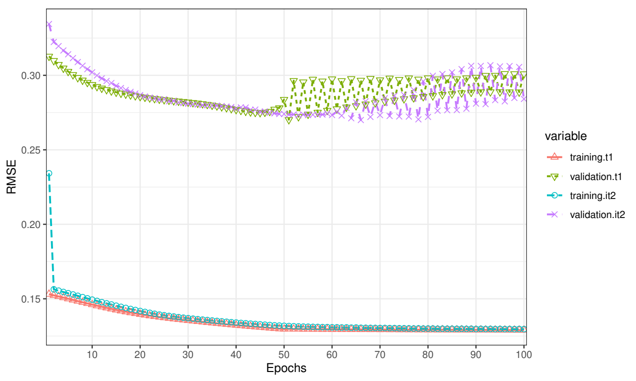


Fig. 11. The training and validation RMSE error curves during the training epochs of type-1 and interval type-2 ANFIS models.

With the function *anfis.plotmf* which can be used to plot the antecedent membership functions of an ANFIS object, the user can visually check the changes of membership functions made by the optimisation. The below code is an example to show the membership functions (see Fig. 12) of the second input used in the interval type-2 ANFIS model demonstrated above.

```
par(mfrow=c(1,2))
anfis.plotmf(anfis.builder(fis.it2), 'input', 2)
anfis.plotmf(anfis.it2, 'input', 2)
```

V. Conclusion

In this paper, we have presented a newly released fuzzy logic toolbox, called *FuzzyR*, for the R programming language. The toolbox is available as a standard R package on CRAN [17]. *FuzzyR* keeps the main functionalities of the previous R toolbox, as summarised in this paper, but many new features have been added in this updated toolbox. Specifically, we have redesigned membership functions and fuzzy operators. TSK type models have been supported in *FuzzyR* and ANFIS functions can be used to optimise the performance of type-1 and interval type-2 TSK models. Moreover, we have implemented the state-of-the-art type-reduction algorithms (e.g. EIASC, DAND and SC) presented in recent studies [30, 31]. In *FuzzyR*, there is an accuracy function which uses eight different accuracy measures including a new accuracy measure UMBRAE proposed in [33]. Additionally, graphical user interfaces have been provided, which can be very useful for the purposes of teaching and learning.

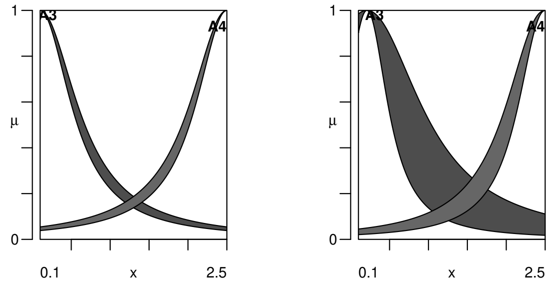


Fig. 12. An illustration of the antecedent membership functions for an interval type-2 ANFIS model before (left) and after (right) optimisation.

Though many new features have been included in *FuzzyR* compared to the previous R toolbox, there are still a lot of things that can be done in the future. For example, currently, the ANFIS models in *FuzzyR* can only be based on generalised bell-shaped membership functions. This could be expanded with the support of more types of membership functions. Also, other optimisation methods (derivative-free algorithms such as simulated annealing and particle swarm optimisation) could also be considered. Furthermore, since there have been more and more research interest in general type-2 fuzzy inference systems, the support of general type-2 functionalities in *FuzzyR* is also in the plan. Also, since the Computational Intelligence Society has endorsed a standard for Fuzzy Markup Language (FML) which enables the interoperability of fuzzy inference systems irrespective of the environment they are executed [34, 35, 36, 37], we are aiming to extend FuzzyR in a future version that complies with the FML.

References

[1] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[2] O. Castillo and P. Melin, "A review on the design and optimization of interval type-2 fuzzy controllers," *Applied Soft Computing*, vol. 12, no. 4, pp. 1267–1278, 2012.

[3] D. Soria, J. M. Garibaldi, A. R. Green, D. G. Powe, C. C. Nolan, C. Lemetre, G. R. Ball, and I. O. Ellis, "A quantifier-based fuzzy classification system for breast

cancer patients." *Artificial Intelligence in Medicine*, vol. 58, no. 3, pp. 175–84, 2013.

[4] C. Wagner, S. Miller, and J. M. Garibaldi, "A fuzzy toolbox for the R programming language," *Proceedings IEEE International Conference on Fuzzy Systems*, pp. 1185–1192, 2011.

[5] J. Alcalá-Fdez and J. M. Alonso, "A Survey of Fuzzy Systems Software: Taxonomy, Current Research Trends, and Prospects," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 1, pp. 40–56, feb 2016.

[6] J. McCulloch, "Fuzzycreator: A python-based toolkit for automatically generating and analysing data-driven fuzzy sets," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2017, pp. 1–6.

[7] R. Babuska, "Fuzzy Modelling and Identification," *Control Engineering Laboratory, Faculty of Information Technology and Systems, Delft University of Technology, Delft, The Netherlands, version*, vol. 3, 2000.

[8] J. R. Castro, O. Castillo, and P. Melin, "An Interval Type-2 Fuzzy Logic Toolbox for Control Applications," pp. 1–6, 2007.

[9] The Mathworks, "Fuzzy Logic Toolbox - Design and simulate fuzzy logic systems," https://www.mathworks.com/products/fuzzy-logic.html, 2019.

[10] J. M. Alonso and L. Magdalena, "Generating Understandable and Accurate Fuzzy Rule-Based Systems in a Java Environment," in *Fuzzy Logic and Applications*, A. M. Fanelli, W. Pedrycz, and A. Petrosino, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 212–219.

[11] S. Guillaume and B. Charnomordic, "Fuzzy Inference Systems: an integrated modelling environment for collaboration between expert knowledge and data using FisPro," *Expert Systems with Applications*, vol. 39, no. 10, pp. 8744–8755, aug 2012.

[12] C. Wagner, "Juzzy - A Java based toolkit for Type-2 Fuzzy Logic," in *Proceedings EEE Symposium on Advances in Type-2 Fuzzy Logic Systems*, 2013, pp. 45–52.

[13] C. Wagner, M. Pierfitt, and J. McCulloch, "Juzzy online: An online toolkit for the design, implementation, execution and sharing of Type-1 and Type-2 fuzzy logic systems," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2014, pp. 2321–2328.

[14] C. Knott, L. Hovell, N. Karimian, and J. M. Garibaldi, "FuzzyToolkitUoN: Type 1 Fuzzy Logic Toolkit," https://CRAN.R-project.org/package=FuzzyToolkitUoN, 2013.

[15] J.-S. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[16] C. Chen, R. John, J. Twycross, and J. M. Garibaldi, "An extended ANFIS architecture and its learning properties for type-1 and interval type-2 models," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2016, pp. 602–609.

[17] C. Chen, J. M. Garibaldi, and T. Razak, "FuzzyR: Fuzzy Logic Toolkit for R," https://CRAN.R-project.org/package=FuzzyR, 2019.

[18] Q. Liang, N. N. Karnik, and J. M. Mendel, "Connection admission control in ATM networks using survey-based type-2 fuzzy logic systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 3, pp. 329–339, 2000.

[19] D. Wu and W. W. Tan, "Genetic learning and performance evaluation of interval type-2 fuzzy logic controllers," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 8, pp. 829–841, 2006.

[20] D. Wu and W. W. Tan, "A simplified type-2 fuzzy logic controller for real-time control," *ISA Transactions*, vol. 45, no. 4, pp. 503–516, 2006.

[21] Z. Liu, Y. Zhang, and Y. Wang, "A Type-2 Fuzzy Switching Control System for Biped Robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1202–1213, 2007.

[22] H. Hagras, "Type-2 FLCs: A New Generation of Fuzzy Controllers," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 30–43, 2007.

[23] O. Castillo and P. Melin, *Type-2 Fuzzy Logic: Theory and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[24] E. A. Jammeh, M. Fleury, C. Wagner, H. Hagras, and M. Ghanbari, "Interval Type-2 Fuzzy Logic Congestion Control for Video Streaming Across IP Networks," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 5, pp. 1123–1142, 2009.

[25] D. Wu and J. M. Mendel, "Enhanced Karnik–Mendel algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 4, pp. 923–934, 2009.

[26] D. Wu and M. Nie, "Comparison and practical implementation of type-reduction algorithms for type-2 fuzzy sets and systems," in *Proceedings IEEE International Conference on Fuzzy Systems*, 2011, pp. 2131–2138.

[27] C. Chen, R. John, J. Twycross, and J. M. Garibaldi, "A Direct Approach for Determining the Switch Points in the Karnik–Mendel Algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 2, pp. 1079–1085, 2018.

[28] M. A. Khanesar, A. J. Khakshour, O. Kaynak, and H. Gao, "Improving the Speed of Center of Sets Type Reduction in Interval Type-2 Fuzzy Systems by Eliminating the Need for Sorting," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 5, pp. 1193–1206, 2017.

[29] C. Chen, D. Wu, J. M. Garibaldi, R. John, J. Twycross, and J. M. Mendel, "A Comment on "A Direct Approach for Determining the Switch Points in the Karnik-Mendel Algorithm"," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 6, pp. 3905 – 3907, 2018.

[30] C. Chen and D. Wu, "Source Code for 'A Comprehensive Study of the Efficiency of Type-Reduction Algorithms'," https://www.codeocean.com/, 2019.

[31] C. Chen, D. Wu, J. M. Garibaldi, R. I. John, J. Twycross, and J. M. Mendel, "A Comprehensive Study of the Efficiency of Type-Reduction Algorithms," *IEEE Transactions on Fuzzy Systems*, p. 1, 2020.

[32] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy,” *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[33] C. Chen, J. Twycross, and J. M. Garibaldi, “A new accuracy measure based on bounded relative error for time series forecasting,” *PLOS ONE*, vol. 12, no. 3, pp. 1–23, 2017.

[34] G. Acampora, *Fuzzy Markup Language: A XML Based Language for Enabling Full Interoperability in Fuzzy Systems Design*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 17–31.

[35] “IEEE Standard for Fuzzy Markup Language,” *IEEE Std 1855-2016*, pp. 1–89, 2016.

[36] J. M. Soto-Hidalgo, J. M. Alonso, G. Acampora, and J. Alcala-Fdez, “JFML: A Java Library to Design Fuzzy Logic Systems According to the IEEE Std 1855-2016,” *IEEE Access*, vol. 6, pp. 54 952–54 964, 2018.

[37] J. M. Soto-Hidalgo, A. Vitiello, J. M. Alonso, G. Acampora, and J. Alcala-Fdez, “Design of Fuzzy Controllers for Embedded Systems With JFML,” *International Journal of Computational Intelligence Systems*, vol. 12, no. 1, pp. 204–214, 2019.