Neuro Fuzzy Modelling

Rahul Unnikrshnan Nair

```r
# function to ensure the all packages are loaded
EnsurePackage<-function(x)
{ # EnsurePackage(x) - Installs and loads a package
  # if necessary
  x <- as.character(x)
  if (!require(x, character.only=TRUE))
  {
    install.packages(pkgs=x,
                     repos="http://cran.r-project.org")
  }
  require(x, character.only=TRUE)

}

#Installs and loads all necessary packages
#
Prepare.Packages<-function(){

  EnsurePackage("neuralnet")
  EnsurePackage("Metrics")
  EnsurePackage("mosaic")
  EnsurePackage("tidyr")
  EnsurePackage("dplyr")
  EnsurePackage("nnet")
  EnsurePackage("devtools")
}

Prepare.Packages()
```

```
## Loading required package: neuralnet
## Loading required package: grid
## Loading required package: MASS
## Loading required package: Metrics
## Loading required package: mosaic
## Loading required package: car
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:neuralnet':
##
##     compute
```

```
##
## The following object is masked from 'package:MASS':
##
##     select
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
##
## Loading required package: lattice
## Loading required package: ggplot2
##
## Attaching package: 'mosaic'
##
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
##
## The following object is masked from 'package:car':
##
##     logit
##
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cov, D, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
##
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
##
## Loading required package: tidyr
## Loading required package: nnet
## Loading required package: devtools
```

```
source_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c
1474d0a505ff044412703516c34f1a4684a5/nnet_plot_update.r')
```

```
## SHA-1 hash of file is 74c80bd5ddbc17ab3ae5ece9c0ed9beb612e87ef
```

```
# loading the dataset
concrete <-
read.csv('/home/rahul/programming/independantStudy/datasets/concrete.cs
v')
str(concrete)
```

```
## 'data.frame':    1030 obs. of  9 variables:
##  $ cement      : num  540 540 332 332 199 ...
##  $ slag        : num  0 0 142 142 132 ...
##  $ ash         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ water       : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ superplastic: num  2.5 2.5 0 0 0 0 0 0 0 0 ...
##  $ coarseagg   : num  1040 1055 932 932 978 ...
##  $ fineagg     : num  676 676 594 594 826 ...
##  $ age         : int  28 28 270 365 360 90 365 28 28 28 ...
##  $ strength    : num  80 61.9 40.3 41 44.3 ...
```

```
head(concrete)
```

```
##   cement  slag ash water superplastic coarseagg fineagg age strength
## 1  540.0   0.0   0   162          2.5    1040.0   676.0  28    79.99
## 2  540.0   0.0   0   162          2.5    1055.0   676.0  28    61.89
## 3  332.5 142.5   0   228          0.0     932.0   594.0 270    40.27
## 4  332.5 142.5   0   228          0.0     932.0   594.0 365    41.05
## 5  198.6 132.4   0   192          0.0     978.4   825.5 360    44.30
## 6  266.0 114.0   0   228          0.0     932.0   670.0  90    47.03
```

```
# normalizing function for dataset
normFun <- function(x) ((x - min(x))/(max(x) - min(x)))

# normalized concrete data
concreteNorm <- as.data.frame(lapply(concrete, normFun))
summary(concreteNorm)
```

```
##      cement            slag              ash             water       
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.2063   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.3442  
##  Median :0.3902   Median :0.06121   Median :0.0000   Median :0.5048  
##  Mean   :0.4091   Mean   :0.20561   Mean   :0.2708   Mean   :0.4774  
##  3rd Qu.:0.5662   3rd Qu.:0.39775   3rd Qu.:0.5912   3rd Qu.:0.5607  
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000  
##   superplastic      coarseagg         fineagg            age         
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000  
##  1st Qu.:0.0000   1st Qu.:0.3808   1st Qu.:0.3436   1st Qu.:0.01648  
##  Median :0.1988   Median :0.4855   Median :0.4654   Median :0.07418  
##  Mean   :0.1927   Mean   :0.4998   Mean   :0.4505   Mean   :0.12270  
##  3rd Qu.:0.3168   3rd Qu.:0.6640   3rd Qu.:0.5770   3rd Qu.:0.15110  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000  
##     strength     
##  Min.   :0.0000  
##  1st Qu.:0.2664  
##  Median :0.4001  
##  Mean   :0.4172  
##  3rd Qu.:0.5457  
##  Max.   :1.0000
```

```
# splitting data to trainig and testing
concrete_train <- concreteNorm[1:773,]
```

```
concrete_test <- concreteNorm[774:1030,]

############################################
#######################################

# Trying out a pure neural network approach

# creating the model
formula_1 <- strength ~
cement+slag+ash+water+superplastic+coarseagg+fineagg+age

#model <- neuralnet(formula_1, data = concrete_train, hidden = 4)
model <- nnet(formula_1,concrete_train, size = 4 )
```

```
## # weights:  41
## initial  value 40.865819
## iter  10 value 13.307093
## iter  20 value 5.401238
## iter  30 value 4.857214
## iter  40 value 4.434943
## iter  50 value 4.134689
## iter  60 value 4.058738
## iter  70 value 4.021908
## iter  80 value 3.998910
## iter  90 value 3.989185
## iter 100 value 3.978242
## final  value 3.978242
## stopped after 100 iterations
```

```
plot.nnet(model)
```

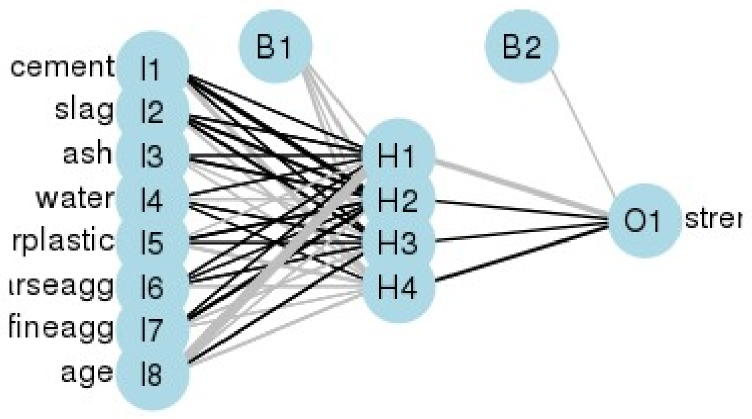
```
## Loading required package: scales
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:mosaic':
##
##     rescale
##
## Loading required package: reshape
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```

```
## Loading required package: reshape
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```

```
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```



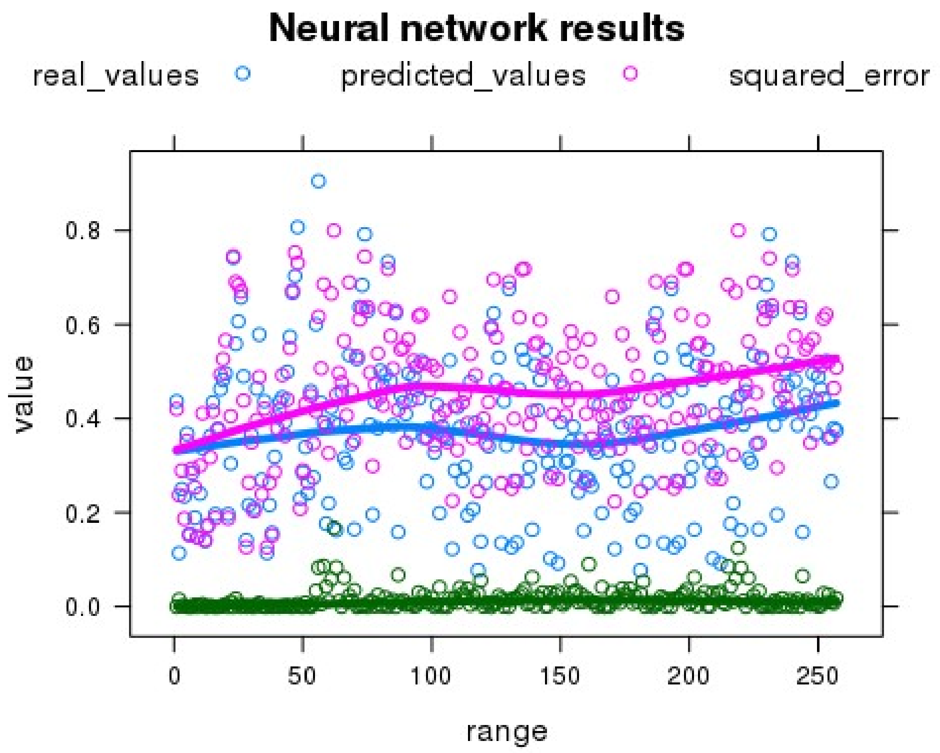
```
# prediction
#model_results <- compute(model, concrete_test[1:8])
nrow(concrete_train)

## [1] 773

model_results <- predict(model,concrete_test[1:8] )
strength_predicted <-  model_results
real_pred <- cbind(concrete_test[9],strength_predicted )
names(real_pred)

## [1] "strength"          "strength_predicted"

# squared error
squared_error <- se(real_pred$strength, real_pred$strength_predicted)
squared_error

##   [1] 2.778463e-04 1.530104e-02 1.500323e-03 2.240207e-03
2.095412e-04
##   [6] 1.099247e-05 8.035328e-05 4.173528e-03 1.445661e-07
3.484908e-03
##  [11] 5.466925e-03 6.450503e-06 4.656107e-06 2.028970e-03
3.610594e-04
##  [16] 6.329952e-05 8.975433e-06 9.496655e-04 1.973124e-03
4.969725e-03
##  [21] 7.527926e-05 1.023240e-02 9.240593e-06 1.729729e-02
5.923127e-03
##  [26] 1.658700e-04 2.580365e-03 2.306561e-04 2.251801e-03
```

```
2.454568e-04
##  [31] 3.512637e-05 4.497062e-03 8.224468e-03 9.290063e-04
2.469877e-04
##  [36] 1.504334e-04 2.251801e-03 3.608346e-05 1.032056e-03
5.428632e-04
##  [41] 2.814912e-04 2.171029e-03 8.136105e-04 3.425287e-03
5.695119e-04
##  [46] 2.137377e-05 2.482048e-03 5.763632e-03 4.670992e-04
2.243314e-05
##  [51] 8.388899e-03 5.204597e-04 7.138604e-05 5.224833e-03
3.405204e-02
##  [56] 8.313559e-02 5.297106e-03 8.630039e-02 4.292990e-02
1.130899e-02
##  [61] 5.876404e-02 1.675504e-01 8.341475e-02 1.194287e-02
3.740151e-04
##  [66] 6.225039e-02 1.025684e-02 2.378857e-02 1.267420e-02
3.542458e-02
##  [71] 4.025916e-05 7.021538e-04 2.356203e-03 2.238303e-03
7.603759e-05
##  [76] 1.214378e-02 1.062519e-02 1.313316e-03 2.520638e-02
3.122000e-03
##  [81] 1.697049e-03 1.980203e-02 2.433204e-04 9.175304e-03
1.442261e-02
##  [86] 3.123072e-05 6.770345e-02 9.459224e-03 1.375009e-02
3.615794e-03
##  [91] 5.339297e-03 5.010397e-03 1.227586e-03 2.260469e-03
3.048277e-02
##  [96] 9.833977e-03 2.482792e-02 8.842938e-03 6.665898e-03
1.898687e-02
## [101] 2.602103e-04 5.247012e-03 4.184273e-02 3.473862e-04
2.920505e-03
## [106] 5.407148e-05 1.825384e-02 1.056336e-02 1.908060e-02
8.427339e-03
## [111] 4.084088e-02 2.777902e-02 2.415327e-02 3.810677e-02
3.541844e-03
## [116] 2.303509e-02 1.025804e-02 2.838058e-02 5.630765e-02
7.454707e-04
## [121] 6.964736e-03 8.837817e-03 2.029381e-05 5.075711e-03
8.493282e-03
## [126] 2.358174e-02 1.617521e-02 7.814652e-04 1.423706e-02
2.002605e-04
## [131] 1.570177e-02 1.742897e-02 1.694198e-02 2.338155e-02
2.892533e-02
## [136] 3.725306e-02 4.781517e-03 8.140207e-03 6.295407e-02
1.964883e-03
## [141] 2.067411e-02 1.622117e-02 3.053468e-02 9.646288e-04
1.236820e-03
## [146] 2.975357e-02 1.588979e-02 2.602557e-02 3.233084e-02
7.552721e-03
## [151] 3.947672e-03 9.986141e-03 1.651950e-02 5.423959e-02
```

```
3.140274e-02
## [156] 3.494898e-02 9.156996e-03 2.715519e-02 1.035036e-02
1.863938e-02
## [161] 9.027163e-02 1.026117e-02 9.089877e-03 2.464724e-04
5.236393e-03
## [166] 4.194418e-02 9.193274e-05 2.108141e-03 3.899349e-04
3.183749e-02
## [171] 1.021981e-02 1.892502e-02 7.790128e-03 3.902013e-02
2.711200e-02
## [176] 2.405501e-02 3.837580e-02 7.057287e-03 2.271477e-02
9.868505e-03
## [181] 2.856343e-02 5.373560e-02 4.057627e-04 7.057240e-03
9.928421e-03
## [186] 8.954591e-05 4.410557e-03 8.923979e-03 2.156174e-02
1.625051e-02
## [191] 7.386394e-04 1.609722e-02 1.916490e-04 1.583221e-02
1.778327e-02
## [196] 1.661499e-02 2.538847e-02 2.910590e-02 3.697127e-02
5.121087e-03
## [201] 9.616744e-03 6.285389e-02 1.961013e-03 2.029556e-02
1.608074e-02
## [206] 3.178388e-02 1.512930e-03 1.353561e-03 2.924631e-02
1.567069e-02
## [211] 2.825322e-02 3.226565e-02 9.199737e-03 5.240818e-03
8.548360e-02
## [216] 4.319812e-02 1.055499e-02 6.036743e-02 1.246602e-01
8.227218e-02
## [221] 1.255291e-02 2.430031e-04 6.154730e-02 1.048932e-02
2.374854e-02
## [226] 1.179066e-02 3.354007e-02 1.595507e-05 7.207027e-04
2.897074e-03
## [231] 2.568774e-03 1.437981e-04 1.326548e-02 9.952232e-03
1.064096e-03
## [236] 1.773511e-02 2.990534e-03 2.060850e-03 1.448268e-02
2.794802e-04
## [241] 9.140519e-03 1.599491e-02 1.848299e-04 6.567322e-02
9.562741e-03
## [246] 1.486985e-02 3.777995e-03 5.429449e-03 5.333338e-03
1.059713e-03
## [251] 2.368616e-03 2.863243e-02 9.871432e-03 2.610900e-02
9.016577e-03
## [256] 7.318474e-03 1.793304e-02
```

```r
# combined real, predicted and squared error into one data frame
range <- seq(1, length(squared_error), 1)
real_pred <- as.data.frame(cbind(concrete_test$strength,
strength_predicted,squared_error, range))
colnames(real_pred) <- c("real_values", "predicted_values","squared_error",
"range")
real_pred_gather <- real_pred %>%  gather("attr", "value", 1:3)
```

```
# results

# finding the correlation between predicted and real value
correlation <- cor(strength_predicted,concrete_test$strength)
correlation

##          [,1]
## [1,] 0.8057775

# mean squared error between observed and predicted values
MSE <- mse(strength_predicted,concrete_test$strength)

# ploting the results
xyplot (value ~ range, group=attr, data =real_pred_gather,type = c("p",
"smooth"), lwd = 4, auto.key = list(columns =
nlevels(real_pred_gather$attr)), main = "Neural network results")
```



```
########################################
##################################

# Trying out a fuzzy inference system for the same

library('frbs')
method.type <- "WM"
range.data<-matrix(apply(concrete_train, 2, range), nrow = 2)
range.data
```

```
##      [,1] [,2]      [,3]      [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    0    0 0.0000000 0.0000000    0    0    0    0    0
## [2,]    1    1 0.8730635 0.8482428    1    1    1    1    1
```

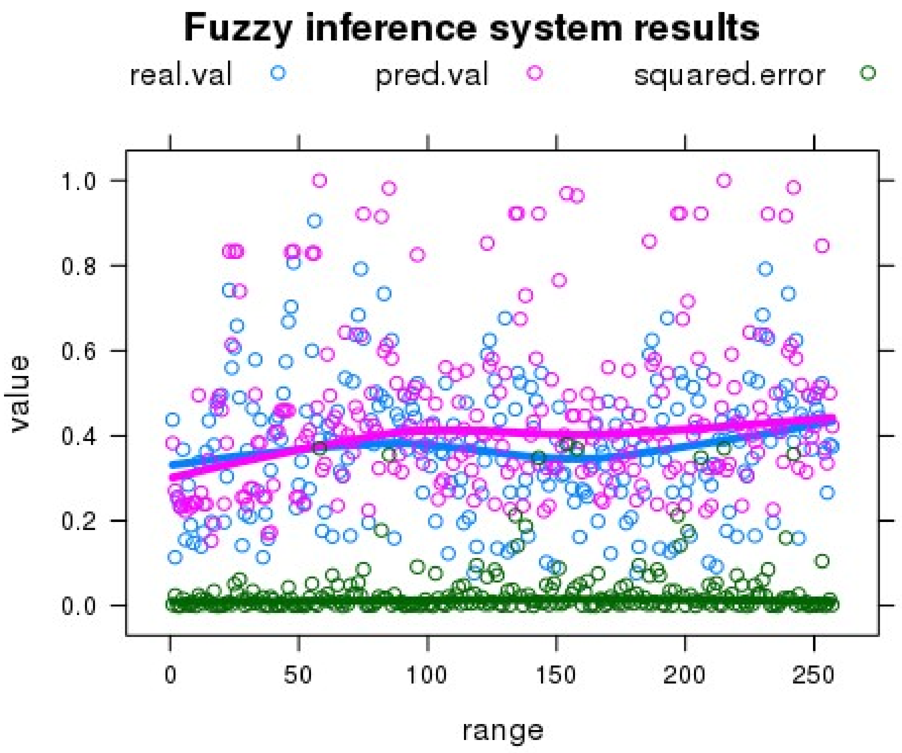
```
control <- list(num.labels = 7, max.iter = 10, step.size = 0.01, type.tnorm =
"MIN", type.snorm = "MAX",type.mf = "GAUSSIAN", type.defuz = "WAM",
type.implication.func = "ZADEH", name = "concrete")
# generating the fuzzy inference rules from the data set
object.WM <- frbs.learn(concrete_train,range.data, method.type, control)
str(object.WM)
```

```
## List of 17
##  $ num.labels           : num [1, 1:9] 7 7 7 7 7 7 7 7 7
##  $ varout.mf            : num [1:5, 1:7] 5 0 0.0583 NA NA ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:7] "vv.small" "v.small" "small" "medium" ...
##  $ rule                 : chr [1:514, 1:36] "IF" "IF" "IF" "IF" ...
##  $ varinp.mf            : num [1:5, 1:56] 5 0 0.0583 NA NA ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:56] "vv.small" "v.small" "small" "medium" ...
##  $ degree.ante          : num [1:514, 1] 0.78 0.777 0.777 0.777 0.741 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr ""
##  $ rule.data.num        : num [1:514, 1:9] 2 4 4 4 4 6 5 4 3 2 ...
##  $ degree.rule          : num [1:514, 1] 0.78 0.777 0.777 0.747 0.741 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr ""
##  $ range.data.ori       : num [1:2, 1:9] 0 1 0 1 0 ...
##  $ type.mf              : chr "GAUSSIAN"
##  $ type.tnorm           : chr "MIN"
##  $ type.implication.func: chr "ZADEH"
##  $ type.model           : chr "MAMDANI"
##  $ type.defuz           : chr "WAM"
##  $ type.snorm           : chr "MAX"
##  $ method.type          : chr "WM"
##  $ name                 : chr "concrete"
##  $ colnames.var         : chr [1:9] "cement" "slag" "ash" "water" ...
##  - attr(*, "class")= chr "frbs"
```

```
# predicting based on the generated fuzzy inference system
pred.WM <- predict(object.WM, concrete_test[1:8])
```

```
## Warning in validate.params(object, newdata): There are your newdata
which
## are out of the specified range
## Warning in validate.params(object, newdata): There are your newdata
which
## are out of the specified range
```

```
real_pred_fuzzy <- as.data.frame(cbind(concrete_test[,9], pred.WM))

# calculating the squared errorof real and predicted values
f_se <- se( real_pred_fuzzy$V1,  real_pred_fuzzy$V2)

f_range <- range <- seq(1, length(f_se), 1)


# combining everything into a dataframe
real_pred_fuzzy <- as.data.frame( cbind(real_pred_fuzzy, f_se, range))
colnames(real_pred_fuzzy) <- c("real.val", "pred.val", "squared.error",
"range")


# merging columns real.val, pred.val and squared.error into 2 columns
real_pred_fuzzy_gather <- real_pred_fuzzy %>% gather("attr", "value", 1:3)

# ploting the results
xyplot (value ~ range, group=attr, data =real_pred_fuzzy_gather,type =
c("p", "smooth"), lwd = 4, auto.key = list(columns =
nlevels(real_pred_fuzzy_gather$attr)), main = "Fuzzy inference system
results")
```



```
# mean squared error of the fuzzy logic value
f_mse <- mse( real_pred_fuzzy$real.val,  real_pred_fuzzy$pred.val)

# correlation between real and predicted strength of concrete using fuzzy
inference
f_correlation <-  cor(real_pred_fuzzy$real.val,real_pred_fuzzy$pred.val )

# MSE matrix for neuro and fuzzy systems
MSE_for_both <- cbind("MSE.nnet" = MSE, "MSE.fuzzy" = f_mse )
correlation_for_both <- cbind("corr.nnet" = correlation, "corr.fuzzy" =
f_correlation)

#############################################
#########################################

#connecting fuzzy inference with a neural net

# inputs: concrete[1:8] + trained fuzzy inference system output
# setting seed for reproducing the shuffle
set.seed(123)
ind <- sample(2, nrow(concrete), replace = TRUE, prob = c(0.7, 0.3))
concrete.train_2 <- concrete[ind ==1,]
concrete.test_2 <- concrete[ind ==2,]

#concreteShuffled <- concrete[sample(nrow(concrete)),]
# normalizing function for dataset
normFun <- function(x) ((x - min(x))/(max(x) - min(x)))
# normalized concrete data
#concreteShuffledNorm <- as.data.frame(lapply(concreteShuffled, normFun))
#summary(concreteShuffledNorm)

concrete.train_2 <- as.data.frame(lapply(concrete.train_2, normFun))
concrete.test_2 <- as.data.frame(lapply(concrete.test_2, normFun))
concrete.train_2.shuffled <-
concrete.train_2[sample(nrow(concrete.train_2)),]

# creating the model
control <- list(num.labels = 7, max.iter = 10, step.size = 0.01, type.tnorm =
"MIN", type.snorm = "MAX",type.mf = "GAUSSIAN", type.defuz = "WAM",
type.implication.func = "ZADEH", name = "concrete_2")

range.data<-matrix(apply(concrete.train_2, 2, range), nrow = 2)
range.data

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    0    0    0    0    0    0    0    0    0
## [2,]    1    1    1    1    1    1    1    1    1

# generating the fuzzy inference rules from the data set
object.WM <- frbs.learn(concrete.train_2,range.data, method.type, control)
```

```
pred.WM_in <- predict(object.WM, concrete.train_2[1:8])
pred.WM.in.test <- predict (object.WM, concrete.test_2[1:8])

pred.WM_in <- as.numeric(pred.WM_in)
pred.WM.in.test  <- as.numeric(pred.WM.in.test)

cor(pred.WM_in, concrete.train_2[9]) # fuzzy predicted and observed value -
training data

##       strength
## [1,] 0.9149172

concrete.fuzzy.train <- as.data.frame(cbind (concrete.train_2, "fuzzy" =
pred.WM_in ))
names(concrete.fuzzy.train[10])

## [1] "fuzzy"

ncol(concrete.fuzzy.train)

## [1] 10

nrow(concrete.fuzzy.train)

## [1] 729

concrete_fuzzy_test <- as.data.frame(cbind (concrete.test_2, "fuzzy" =
pred.WM.in.test))

names(concrete_fuzzy_test)

##  [1] "cement"       "slag"         "ash"          "water"
##  [5] "superplastic" "coarseagg"    "fineagg"      "age"
##  [9] "strength"     "fuzzy"

ncol (concrete_fuzzy_test)

## [1] 10

nrow (concrete_fuzzy_test)

## [1] 301

str(concrete_fuzzy_test)

## 'data.frame':    301 obs. of  10 variables:
##  $ cement      : num  1 0.526 0.221 0.635 0.221 ...
##  $ slag        : num  0 0.396 0.368 0.264 0.368 ...
##  $ ash         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ water       : num  0.349 1 0.645 1 0.645 ...
##  $ superplastic: num  0.0887 0 0 0 0 ...
##  $ coarseagg   : num  0.762 0.393 0.532 0.393 0.532 ...
##  $ fineagg     : num  0.206 0 0.581 0 0.581 ...
##  $ age         : num  0.0742 1 0.9863 0.0742 0.2445 ...
##  $ strength    : num  0.742 0.47 0.513 0.411 0.432 ...
##  $ fuzzy       : num  0.999 0.504 0.424 0.509 0.349 ...
```

```
# creating a nnet model - something wrong here with the setup
formula_2 <- strength ~
cement+slag+ash+water+superplastic+coarseagg+fineagg+age+fuzzy
model2 <- nnet(formula_2,concrete.fuzzy.train, size = 4 )
```

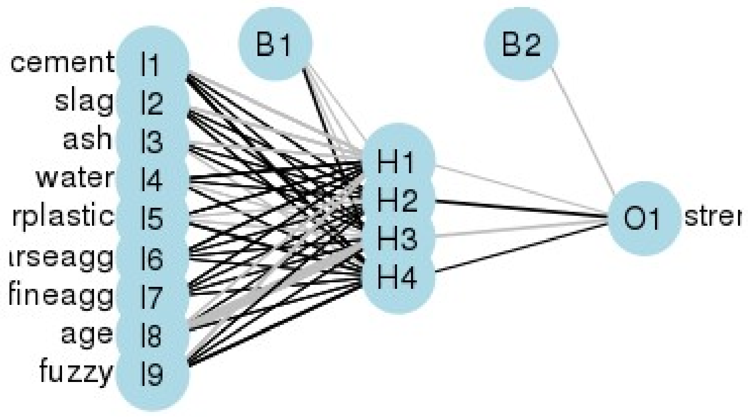
```
## # weights:  45
## initial  value 32.975809
## iter  10 value 6.149119
## iter  20 value 4.329640
## iter  30 value 2.872440
## iter  40 value 2.537507
## iter  50 value 2.301020
## iter  60 value 2.197529
## iter  70 value 2.122916
## iter  80 value 2.077253
## iter  90 value 2.050253
## iter 100 value 2.042970
## final  value 2.042970
## stopped after 100 iterations
```

```
names(concrete.fuzzy.train)
```

```
##  [1] "cement"       "slag"         "ash"          "water"
##  [5] "superplastic" "coarseagg"    "fineagg"      "age"
##  [9] "strength"     "fuzzy"
```

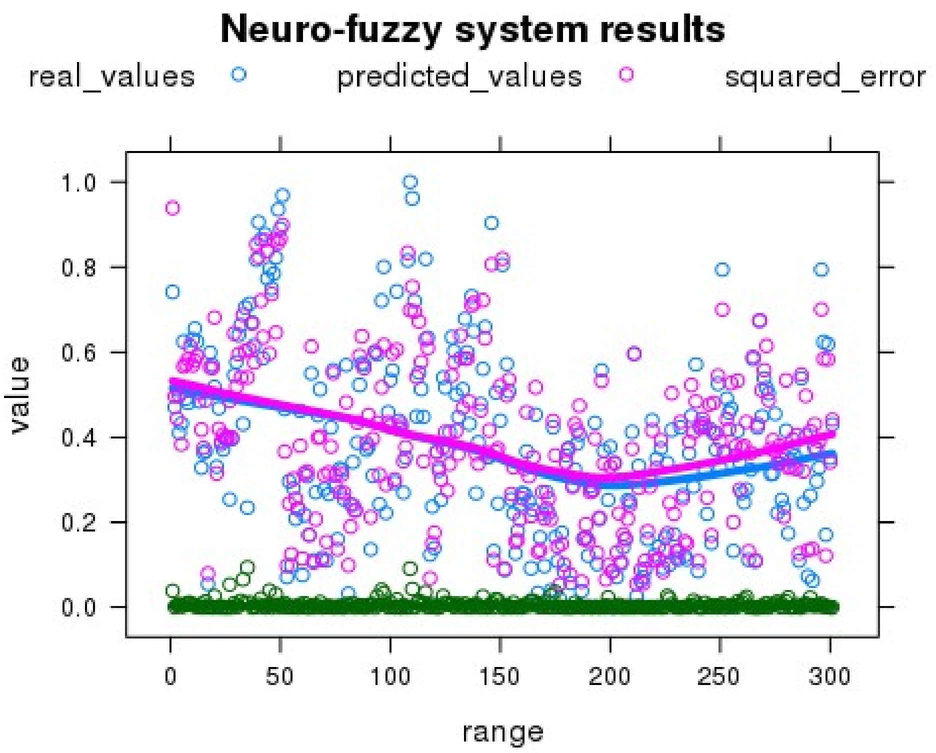
```
plot.nnet(model2)
```

```
## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'

## Loading required package: reshape

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```



```
model_results_final <- predict(model2,concrete_fuzzy_test)

# squared error
squared_error <- se(concrete_fuzzy_test[,9], model_results_final)

# combined real, predicted and squared error into one data frame
range <- seq(1, length(squared_error), 1)
real_pred <- as.data.frame(cbind(concrete_fuzzy_test$strength,
model_results_final, squared_error, range))
colnames(real_pred) <- c("real_values", "predicted_values","squared_error",
"range")
real_pred_gather <- real_pred %>%  gather("attr", "value", 1:3)
head(real_pred_gather)

##   range        attr     value
## 1     1 real_values 0.7415745
## 2     2 real_values 0.4703969
## 3     3 real_values 0.5126871
## 4     4 real_values 0.4105400
## 5     5 real_values 0.4316200
## 6     6 real_values 0.6247235

# results

# finding the correlation between predicted and real value
correlation <- cor(model_results_final,concrete_fuzzy_test$strength)
correlation
```

```
##          [,1]
## [1,] 0.9314786

# mean squared error between observed and predicted values
MSE <- mse(model_results_final,concrete_fuzzy_test$strength)
MSE

## [1] 0.006202421

# MSE  and correlation matrix for neuro, fuzzy  and fuzzy-neuro systems

MSE_for_all <- cbind(MSE_for_both, "mse.neuro_fuzzy" = MSE)
MSE_for_all <- MSE_for_all[,1:3]
correlation_for_all <- cbind(correlation_for_both, "corr_neuro_fuzzy" =
correlation)
colnames(correlation_for_all) <- c("corr.neuro", "corr.fuzzy",
"corr.neuro.fuzzy")
# output
MSE_for_all

##        MSE.nnet       MSE.fuzzy mse.neuro_fuzzy
##     0.015829179     0.034375753     0.006202421

correlation_for_all

##      corr.neuro corr.fuzzy corr.neuro.fuzzy
## [1,]  0.8057775  0.5480245        0.9314786

# ploting the results
xyplot (value ~ range, group=attr, data =real_pred_gather,type = c("p",
"smooth"), lwd = 4, auto.key = list(columns =
nlevels(real_pred_gather$attr)), main = "Neuro-fuzzy system results")
```



```
###################################################
##################################
            #### deep learning using h2o library####
```