Package ‘anfis’

April 23, 2012

Type Package

Title ANFIS Type 3 Takagi and Sugeno’s fuzzy if-then rule network.

Version 1.01

Date 2012-02-15

Author Cristobal Fresno, Elmer A. Fernandez

Maintainer Cristobal Fresno <`cristobalfresno@gmail.com`>

Description The implementation has the following features (1) Independent number of membership functions(MF) for each input, and also different MF (2) Type 3 Takagi and Sugeno’s fuzzy if-then rule (3) Full Rule combinations, e.g. 2 inputs 2 membership funtions -> 4 fuzzy rules (4) Hibrid learning, i.e. Descent Gradient for precedents and Least Squares Estimation for consequents (5) Multiple outputs.

License GPL (>=2)

Depends R (>= 2.14.1), methods, multicore, membershipfunction, nnet,xtable

Imports methods, multicore, membershipfunction, nnet, xtable

Collate ’Anfis.R’ ’Anfis-initialize.R’ ’Anfis-getters.R’’Anfis-metrics.R’ ’Anfis-printshow.R’ ’Anfis-plotMF.R’’Anfis-plot.R’ ’Anfis-predict.R’ ’Anfis-training.R’’Anfis-trainSet.R’

R topics documented:

`ANFIS-class` *ANFIS S4 class implementation in R*

Description

Features: 1.- Independent number of membership functions(MF) for each input, and also different MF 2.- Type 3 Takagi and Sugeno's fuzzy if-then rule. 3.- Full Rule combinations, e.g. 2 inputs 2 membership funtions -> 4 fuzzy rules. 4.- Hibrid learning, i.e. Descent Gradient for precedents and Least Squares Estimation for consequents. 5.- Multiple outputs.

Details

premises list with the MembershipFunctions for each input

consequents numeric matrix with nrow= #rules, ncol= #outputs

rules matrix with the conectivity of the membership functions to the rules

X input matrix with ncol=#inputs and nrow=#individuals

Y output matrix with ncol=#output and nrow=#individuals

errors numeric vector with training errors

trainingType character describing the training algorithm used (trainHybridJangOffLine, trainHybridOffLine or trainHybridJangOnLine)

fitted.values numeric matrix with predicted values for training data X

residuals numeric matrix with residuals values for training data X

call call class object with training call

Note

Additional functions implemented: (initialize) constructur of ANFIS Architecture to regerate the rule set and consequents; (show/print) generic output of the object; (getRules, getPremises, getConsequents, getErrors, getTrainingType) return the respective ANFIS slots; (plotMF) plot MembershipFunctions domain; (plotMFs) plot all the MembershipFunctions for the input domain; (plot) plot trainnig error acording with training Type; (LSE) auxiliary function for Least Square Estimation to avoid singular matrix system in offline training; (trainHybridJangOffLine) Jang Hybrid offline training; (trainHybridOffLine) Hybrid off-line training with momentum and adaptative learning rate; (trainHybridJangOnLine) Jang Hybrid on-line training; (summary, fitted, fitted.values, coef, coefficients, resid, residuals) wrappers for traditional model functions

See Also

`BellMF-class`, `GaussianMF-class` and `NormalizedGaussianMF-class`

Examples

```
##Set 4 cores using global options for multicore
options(cores=4)
#
##Example domain for bidimentional sinc(x,y) function
x <- seq(-10, 10, length= 11)
trainingSet <- trainSet(x,x)
Z <- matrix(trainingSet[,"z"],ncol=length(x),nrow=length(x))
## Not run: persp(x,x,Z,theta = 45, phi = 15, expand = 0.8, col = "lightblue",ticktype="detailed",main="sinc(x)*sin
#
##Training domain patterns
X <- trainingSet[,1:2]
Y <- trainingSet[,3,drop=FALSE]
#
##Defining the required MembershipFunctions for the ANFIS
membershipFunction <- list(x=c(new(Class="NormalizedGaussianMF",parameters=c(mu=-10,sigma=2)),
       new(Class="NormalizedGaussianMF",parameters=c(mu=-5,sigma=2)),
       new(Class="NormalizedGaussianMF",parameters=c(mu=0,sigma=2)),
       new(Class="NormalizedGaussianMF",parameters=c(mu=5,sigma=2)),
       new(Class="NormalizedGaussianMF",parameters=c(mu=10,sigma=2))),
   y=c(new(Class="NormalizedGaussianMF",parameters=c(mu=-10,sigma=2)),
       new(Class="NormalizedGaussianMF",parameters=c(mu=-5,sigma=2)),
       new(Class="NormalizedGaussianMF",parameters=c(mu=0,sigma=2)),
       new(Class="NormalizedGaussianMF",parameters=c(mu=5,sigma=2)),
       new(Class="NormalizedGaussianMF",parameters=c(mu=10,sigma=2))))
#
##Creating the ANFIS network with 2 inputs and 4 MembershipFunctions in each input
anfis3 <- new(Class="ANFIS",X,Y,membershipFunction)
anfis3
#
##Check for epsilon-completeness in each input
## Not run: plotMFs(anfis3)
#
##Training the ANFIS network
trainOutput <- trainHybridJangOffLine(anfis3, epochs=10)
#
##How the training went
## Not run: plot(anfis3)
#
##Test the fit
##MembershipFunctions
## Not run: plotMFs(anfis3)
#
##Just to see if premises, consequents and errors were updated
getPremises(anfis3)[[1]][[1]]
getConsequents(anfis3)[1:2,]
getErrors(anfis3) #Training errors
getTrainingType(anfis3)
names(coef(anfis3))
coef(anfis3)$premises[[input=1]][[mf=1]]
coef(anfis3)$consequents[1:2,]
#
```

```
##First five train pattern associated values for the training process
fitted(anfis3)[1:5,]
resid(anfis3)[1:5,]
summary(anfis3)
#
##Surface comparison between the original training set and the predicted ANFIS network
y <- predict(anfis3,X)
z <- matrix(y[,1],ncol=length(x),nrow=length(x))
## Not run: par(mfrow=c(1,2))
persp(x,x,Z,theta = 45, phi = 15, expand = 0.8, col = "lightblue",ticktype="detailed",main="Goal",xlim=c(-10,10),y
persp(x,x,z,theta = 45, phi = 15, expand = 0.8, col = "lightblue",ticktype="detailed",main="Fitted training Pattern
## End(Not run)
```

fitted *ANFIS trainnig results*

Description

Obtain ANFIS slot information, acording to training output

Usage

```
  ## S4 method for signature ’ANFIS’
fitted.values(object, ...)

  ## S4 method for signature ’ANFIS’
coef(object, ...)

  ## S4 method for signature ’ANFIS’
coefficients(object, ...)

  ## S4 method for signature ’ANFIS’
resid(object, ...)

  ## S4 method for signature ’ANFIS’
residuals(object, ...)

  ## S4 method for signature ’ANFIS’
summary(object, ...)
```

Arguments

`object`	ANFIS class object
`...`	required by resid, residuals, coef and coefficients

Value

according to the call one of the following objects can be returned

`list`	list with premises and consequents
`numeric`	numeric vector with trainnig errors, fitted training values and residuals
`printed`	statistics of the training process

Note

see full example in `ANFIS-class`

`getRules` *Getters for ANFIS object*

Description

Obtain ANFIS slot information, acording to the given function call.

Usage

```
  ## S4 method for signature 'ANFIS'
getRules(object)

  ## S4 method for signature 'ANFIS'
getPremises(object)

  ## S4 method for signature 'ANFIS'
getConsequents(object)

  ## S4 method for signature 'ANFIS'
getErrors(object)

  ## S4 method for signature 'ANFIS'
getTrainingType(object)
```

Arguments

`object`	ANFIS class object

Value

according to the call one of the following objects can be returned

`matrix`	numeric matrix with rules or consequents
`list`	list with MembershipFunctions or premises and consequents
`character`	name of the trainingType
`numeric`	numeric vector with trainnig errors, fitted training values and residuals

Note

see full example in ANFIS-class

`initialize` *`initialize` ANFIS object constructor*

Description

Create the ANFIS object arquitecture for the trainingSet (X,Y) with full rules

Arguments

`.Object`	ANFIS class
`X`	input matrix with ncol=#inputs and nrow=#individuals
`Y`	output matrix with ncol=#output and nrow=#individuals
`membershipFunction`	list with the MembershipFunction for each input

Value

ANFIS object

Note

see full example in ANFIS-class

See Also

ANFIS-class

`LSE` *Train ANFIS network*

Description

ANFIS on-line or off-line hybrid Jang dinamic learning training process. In addition for off-line learning there is also adaptative learning coeficient and momentum term.

Usage

```
  ## S4 method for signature 'ANFIS'
LSE(object, A, B, initialGamma = 1000)

  ## S4 method for signature 'ANFIS'
trainHybridJangOffLine(object,
    epochs = 5, tolerance = 1e-05, initialGamma = 1000,
    k = 0.01)

  ## S4 method for signature 'ANFIS'
trainHybridOffLine(object, epochs = 5,
    tolerance = 1e-05, initialGamma = 1000, eta = 0.05,
    phi = 0.2, a = 0.01, b = 0.1, delta_alpha_t_1 = list())

  ## S4 method for signature 'ANFIS'
trainHybridJangOnLine(object,
    epochs = 5, tolerance = 1e-15, initialGamma = 1000,
    k = 0.01, lamda = 0.9, S = matrix(nrow = 0, ncol = 0))
```

Arguments

`object`	ANFIS' class object
`A`	internal matrix for Iterative Least Squares Estimation of the system AX=B
`B`	internal matrix for Iterative Least Squares Estimation of the system AX=B
`initialGamma`	numeric large number » 0. Default 1000
`epochs`	the max number of training epochs. Default 5
`tolerance`	convergence error to stop training. Default 1e-5
`k`	numeric with the initial step size for the learning rule. Default 0.01
`eta`	numeric learning rule coefficient. Default 0.05
`phi`	numeric momentum rule coefficient. Default 0.2
`a`	numeric step to increase eta if delta_e is < 0, i.e. descending. Default 0.01
`b`	numeric fraction to decrease eta if delta_e is > 0, i.e. ascending. Default 0.1
`delta_alpha_t_1`	list with numeric matrix with last time step. Default list()
`lamda`	0 < numeric < 1 forgetting factor. Default 0.9
`S`	covariance matrix for on-line LSE. Default matrix(nrow=0,ncol=0)

Value

`matrix`	with the system solution for LSE output
`error`	numeric vector with training asociated errors (pattern or epoch) according to trainingType
`convergence`	TRUE/FALSE if it reached convergence or not
`updated`	trainingType, premises, consequents, error, residuals, fitted.values and coefficient

Note

see full example in ANFIS-class

See Also

ANFIS-class

plot *Plot ANFIS training errors*

Description

Plot the trainning error of the network. If trainingType is "on-line" then full pattern errors along the patterns of the whole training process; for a specific epoch or the epoch summary error

Arguments

`x` ANFIS class object

`y` not used but necesary for redefining the generic function

`epoch` for on-line only: epoch == Inf the whole training error; epoch == integer > 0 the give epoch trainings errors, epoch == 0 the abs epoch trainnig sum of errors.

`...` plot aditional parameters

Value

output graphics

Note

see full example in ANFIS-class

See Also

ANFIS-class

plotMF *PlotMF/s ANFIS' MembershipFunction domain/s*

Description

Plot the corresponding MembershipFunctions for each/all input/s domain

Usage

```
  ## S4 method for signature 'ANFIS'
plotMF(object, x, input, ...)

  ## S4 method for signature 'ANFIS'
plotMFs(object, ...)
```

Arguments

`object`	ANFIS class object
`x`	numeric sequence to evaluate each MembershipFunction
`input`	integer with the input MembershipFunctions to plot
`...`	plot aditional parameters

Value

output graphics

Note

see full example in `ANFIS-class`

See Also

`ANFIS-class`

predict `Predict` *ANFIS' network output*

Description

Foward Pass to predict the ANFIS' output

Arguments

`object`	ANFIS class object
`x`	numeric matrix [patterns x inputs] of input patterns

Value

matrix with the output values

Note

see full example in `ANFIS-class`

See Also

`ANFIS-class`

`print` `Print` *and* `Show` *an ANFIS object*

Description

Generic Print/Show Method for ANFIS class output visualization. Usage: print(x, ...), show(object,...)

Arguments

`x`	ANFIS class object
`object`	ANFIS class object
`...`	not used but included for generic print comparitibility

Value

console output of the object

Note

see full example in `ANFIS-class`

See Also

`ANFIS-class`

trainSet *Bidimentional Sinc train set example*

Description

Generates the training set of sinc(x)*sinc(y) for the (x,y) regular grid

Usage

```
trainSet(x, y)
```

Arguments

x	numeric vector with the x-th grid coordenates
y	numeric vector with the x-th grid coordenates

Value

matrix	numeric matrix with the columns x, y and z=sync(x,y)

Examples

```
##Domain definition for a regular (x,y) grid with 11 points for each coordenates
x <- seq(-10, 10, length= 11)
trainingSet <- trainSet(x,x)
Z <- matrix(trainingSet[,"z"],ncol=length(x),nrow=length(x))

##Ploting the domain
persp(x,x,Z,theta = 45, phi = 15, expand = 0.8, col = "lightblue",ticktype="detailed",main="sinc(x)*sinc(y)")
```

Index