

Package ‘evreg’

April 22, 2024

Type Package

Title Evidential Regression

Version 1.0.4

Date 2024-04-22

Description An implementation of the 'Evidential Neural Network for Regression' model recently introduced in Denoeux (2023) <[doi:10.1109/TFUZZ.2023.3268200](https://doi.org/10.1109/TFUZZ.2023.3268200)>. In this model, prediction uncertainty is quantified by Gaussian random fuzzy numbers as introduced in Denoeux (2023) <[doi:10.1016/j.fss.2022.06.004](https://doi.org/10.1016/j.fss.2022.06.004)>. The package contains functions for training the network, tuning hyperparameters by cross-validation or the hold-out method, and making predictions. It also contains utilities for making calculations with Gaussian random fuzzy numbers (such as, e.g., computing the degrees of belief and plausibility of an interval, or combining Gaussian random fuzzy numbers).

License GPL-3

Depends R (>= 3.1.0)

Imports evclust,stats

Encoding UTF-8

VignetteBuilder knitr

Suggests knitr,rmarkdown,nnet,MASS,ggplot2

RoxygenNote 7.2.3

NeedsCompilation no

Author Thierry Denoeux [aut, cre] (<<https://orcid.org/0000-0002-0660-5436>>)

Maintainer Thierry Denoeux <tdenoeux@utc.fr>

Repository CRAN

Date/Publication 2024-04-22 14:12:35 UTC

R topics documented:

Bel	2
Belint	3
combination_GRFN	4

ENNreg	5
ENNreg_cv	7
ENNreg_holdout	9
ENNreg_init	11
evreg	12
intervals	13
Pl	14
pl_contour	15
predict.ENNreg	16

Index	18
--------------	-----------

Bel	<i>Degree of belief of interval for a Gaussian random fuzzy number</i>
-----	--

Description

Bel computes the degree of belief of an interval [x,y] for a given Gaussian random fuzzy number.

Usage

```
Bel(x, y, GRFN)
```

Arguments

x	The lower bound of the interval (may be a vector).
y	The upper bound of the interval (may be a vector).
GRFN	A Gaussian random fuzzy number, encoded as a list with components mu, sig and h.

Value

The degree of belief of the interval.

References

Thierry Denoeux. Reasoning with fuzzy and uncertain evidence using epistemic random fuzzy sets: general framework and practical models. Fuzzy Sets and Systems, Vol. 453, Pages 1-36, 2023.

See Also

[Belint](#), [Pl](#), [pl_contour](#), [combination_GRFN](#)

Examples

```
bel<-Bel(1,2,list(mu=2,sig=1,h=2))
print(bel)
```

Belint	<i>Finds a belief interval centered on mu for a Gaussian random fuzzy number</i>
--------	--

Description

Belint find an interval of the form $[\mu-r, \mu+r]$ with specified degree of belief for a Gaussian random fuzzy number.

Usage

```
Belint(level = 0.9, GRFN)
```

Arguments

level	The specified degree of belief (between 0 and 1).
GRFN	A Gaussian random fuzzy number, encoded as a list with components mu, sig and h.

Value

A vector containing the lower and upper bounds of the interval.

References

Thierry Denoeux. Reasoning with fuzzy and uncertain evidence using epistemic random fuzzy sets: general framework and practical models. Fuzzy Sets and Systems, Vol. 453, Pages 1-36, 2023.

See Also

[Bel](#), [Pl](#), [pl_contour](#)

Examples

```
int<-Belint(0.9,list(mu=2,sig=1,h=2))
print(int)
```

combination_GRFN	<i>Combination of Gaussian random fuzzy numbers</i>
------------------	---

Description

combination_GRFN combines two Gaussian random fuzzy numbers using the generalized product-intersection rule with soft or hard normalization.

Usage

```
combination_GRFN(GRFN1, GRFN2, soft = TRUE)
```

Arguments

GRFN1	A Gaussian random fuzzy number, encoded as a list with components mu, sig and h.
GRFN2	A Gaussian random fuzzy number, encoded as a list with components mu, sig and h.
soft	If TRUE (default), the combination rule with soft normalization is used. Otherwise, hard normalization is employed.

Value

A list with two components:

GRFN The combined Gaussian random fuzzy number, encoded as a list with components mu, sig and h.

conflict The degree of conflict (equal to 0 if soft==FALSE).

References

Thierry Denoeux. Reasoning with fuzzy and uncertain evidence using epistemic random fuzzy sets: general framework and practical models. Fuzzy Sets and Systems, Vol. 453, Pages 1-36, 2023.

See Also

[Bel](#), [Pl](#), [pl_contour](#)

Examples

```
GRFN1<-list(mu=1,sig=1,h=2)
GRFN2<-list(mu=2,sig=2,h=3)
GRFN12s<-combination_GRFN(GRFN1,GRFN2) # soft normalization
GRFN12h<-combination_GRFN(GRFN1,GRFN2,soft=FALSE) # hard normalization
print(GRFN12s)
print(GRFN12h)
```

Description

ENNreg trains the ENNreg model using batch or minibatch learning procedures.

Usage

```
ENNreg(
  X,
  y,
  init = NULL,
  K = NULL,
  batch = TRUE,
  nstart = 100,
  c = 1,
  lambda = 0.9,
  xi = 0,
  rho = 0,
  eps = NULL,
  nu = 1e-16,
  optimProto = TRUE,
  verbose = TRUE,
  options = list(maxiter = 1000, rel.error = 1e-04, print = 10),
  opt.rmsprop = list(batch_size = 100, epsi = 0.001, rho = 0.9, delta = 1e-08, Dtmax =
    100)
)
```

Arguments

X	Input matrix of size $n \times p$, where n is the number of objects and p the number of attributes.
y	Vector of length n containing observations of the response variable.
init	Initial model generated by ENNreg_init (default=NULL).
K	Number of prototypes (default=NULL; must be supplied if initial model is not supplied).
batch	If TRUE (default), batch learning is used; otherwise, online learning is used.
nstart	Number of random starts of the k-means algorithm (default: 100, used only if initial model is not supplied).
c	Multiplicative coefficient applied to scale parameter gamma (default: 1, used only if initial model is not supplied)
lambda	Parameter of the loss function (default=0.9)
xi	Regularization coefficient penalizing precision (default=0).

<code>rho</code>	Regularization coefficient shrinking the solution towards a linear model (default=0).
<code>eps</code>	Parameter of the loss function (if NULL, set to 0.01 times the standard deviation of y).
<code>nu</code>	Parameter of the loss function to avoid a division par zero (default=1e-16).
<code>optimProto</code>	If TRUE (default), the initial prototypes are optimized.
<code>verbose</code>	If TRUE (default) intermediate results are displayed.
<code>options</code>	Parameters of the optimization procedure (see details).
<code>opt.rmsprop</code>	Parameters of the RMSprop algorithm (see details).

Details

If `batch=TRUE`, function `harris` from package `evclust` is used for optimization. Otherwise, the RMSprop minibatch learning algorithm is used. The three parameters in list `options` are:

maxiter Maximum number of iterations (default: 100).

rel.error Relative error for stopping criterion (default: 1e-4).

print Number of iterations between two displays (default: 10).

Additional parameters for the RMSprop, used only if `batch=FALSE`, are contained in list `opt.rmsprop`. They are: '

batch_size Minibatch size.

epsi Global learning rate.

rho Decay rate.

delta Small constant to stabilize division by small numbers.

Dtmax The algorithm stops when the loss has not decreased in the last `Dtmax` iterations.

Value

An object of class "ENNreg" with the following components:

loss Value of the loss function.

param Parameter values.

K Number of prototypes.

pred Predictions on the training set (a list containing the prototype unit activations, the output means, variances and precisions, as well as the lower and upper expectations).

References

Thierry Denoeux. An evidential neural network model for regression based on random fuzzy numbers. In "Belief functions: Theory and applications (proc. of BELIEF 2022)", pages 57-66, Springer, 2022.

Thierry Denoeux. Quantifying prediction uncertainty in regression using random fuzzy sets: the ENNreg model. IEEE Transactions on Fuzzy Systems, Vol. 31, Issue 10, pages 3690-3699, 2023.

See Also

[predict.ENNreg](#), [ENNreg_init](#), [ENNreg_cv](#), [ENNreg_holdout](#)

Examples

```
# Boston dataset

library(MASS)
X<-as.matrix(scale(Boston[,1:13]))
y<-Boston[,14]
set.seed(220322)
n<-nrow(Boston)
ntrain<-round(0.7*n)
train <-sample(n,ntrain)
fit <- ENNreg(X[train,],y[train],K=30)
plot(y[train],fit$pred$mu,xlab="observed response",ylab="predicted response")
```

ENNreg_cv

Hyperparameter tuning for the ENNreg model using cross-validation

Description

ENNreg_cv tunes parameters ξ and ρ of the ENNreg model using cross-validation.

Usage

```
ENNreg_cv(
  X,
  y,
  K,
  batch = TRUE,
  folds = NULL,
  Kfold = 5,
  XI,
  RHO,
  nstart = 100,
  c = 1,
  lambda = 0.9,
  eps = NULL,
  nu = 1e-16,
  optimProto = TRUE,
  verbose = TRUE,
  options = list(maxiter = 1000, rel.error = 1e-04, print = 10),
  opt.rmsprop = list(batch_size = 100, epsi = 0.001, rho = 0.9, delta = 1e-08, Dtmax =
    100)
)
```

Arguments

<code>X</code>	Input matrix of size $n \times p$, where n is the number of objects and p the number of attributes.
<code>y</code>	Vector of length n containing observations of the response variable.
<code>K</code>	Number of prototypes.
<code>batch</code>	If TRUE (default), batch learning is used; otherwise, online learning is used.
<code>folds</code>	Vector of length n containing the folds (integers between 1 and <code>Kfold</code>).
<code>Kfold</code>	Number of folds (default=5, used only if <code>folds</code> is not provided).
<code>XI</code>	Vector of candidate values for hyperparameter x_i .
<code>RHO</code>	Vector of candidate values for hyperparameter ρ .
<code>nstart</code>	Number of random starts of the k-means algorithm (default: 100).
<code>c</code>	Multiplicative coefficient applied to scale parameter γ (default: 1).
<code>lambda</code>	Parameter of the loss function (default=0.9).
<code>eps</code>	Parameter of the loss function (if NULL, fixed to 0.01 times the standard deviation of y).
<code>nu</code>	Parameter of the loss function to avoid a division par zero (default=1e-16).
<code>optimProto</code>	If TRUE (default), the initial prototypes are optimized.
<code>verbose</code>	If TRUE (default) intermediate results are displayed.
<code>options</code>	Parameters of the optimization algorithm (see ENNreg).
<code>opt.rmsprop</code>	Parameters of the RMSprop algorithm (see ENNreg).

Details

Either the `folds` (a vector of the same length as `y`, such that `folds[i]` equals the fold, between 1 and `Kfold`, containing observation i), or the number of folds must be provided. Arguments `options` and `opt.rmsprop` are passed to function [ENNreg](#).

Value

A list with three components:

xi Optimal value of x_i .

rho Optimal value of ρ .

RMS Matrix of root mean squared error values.

References

Thierry Denoeux. An evidential neural network model for regression based on random fuzzy numbers. In "Belief functions: Theory and applications (proc. of BELIEF 2022)", pages 57-66, Springer, 2022.

Thierry Denoeux. Quantifying prediction uncertainty in regression using random fuzzy sets: the ENNreg model. IEEE Transactions on Fuzzy Systems, Vol. 31, Issue 10, pages 3690-3699, 2023.

See Also

[ENNreg](#), [ENNreg_holdout](#)

Examples

```
# Boston dataset

library(MASS)
X<-as.matrix(scale(Boston[,1:13]))
y<-Boston[,14]
set.seed(220322)
n<-nrow(Boston)
ntrain<-round(0.7*n)
train <-sample(n,ntrain)
cv<-ENNreg_cv(X=X[train,],y=y[train],K=30,XI=c(0.1,1,10),RHO=c(0.1,1,10))
cv$RMS
fit <- ENNreg(X[train,],y[train],K=30,xi=cv$xi,rho=cv$rho)
pred<-predict(fit,newdata=X[-train,],yt=y[-train])
print(pred$RMS)
```

ENNreg_holdout

Hyperparameter tuning for the ENNreg model using the hold-out method

Description

ENNreg_holdout tunes parameters xi and rho of the ENNreg model using the hold-out method.

Usage

```
ENNreg_holdout(
  X,
  y,
  K,
  batch = TRUE,
  val = NULL,
  nval = NULL,
  XI,
  RHO,
  nstart = 100,
  c = 1,
  lambda = 0.9,
  eps = NULL,
  nu = 1e-16,
  optimProto = TRUE,
  verbose = TRUE,
  options = list(maxiter = 1000, rel.error = 1e-04, print = 10),
```

```

opt.rmsprop = list(batch_size = 100, epsi = 0.001, rho = 0.9, delta = 1e-08, Dtmax =
  100)
)

```

Arguments

<code>X</code>	Input matrix of size $n \times p$, where n is the number of objects and p the number of attributes.
<code>y</code>	Vector of length n containing observations of the response variable.
<code>K</code>	Number of prototypes.
<code>batch</code>	If TRUE (default), batch learning is used; otherwise, online learning is used.
<code>val</code>	Vector of indices of the validation instances ($nval$ integers between 1 and n). Needed only if <code>nval</code> is not provided.
<code>nval</code>	Number of validation instances (needed only if <code>val</code> is not provided).
<code>XI</code>	Vector of candidate values for hyperparameter ξ .
<code>RHO</code>	Vector of candidate values for hyperparameter ρ .
<code>nstart</code>	Number of random starts of the k-means algorithm (default: 100).
<code>c</code>	Multiplicative coefficient applied to scale parameter γ (default: 1).
<code>lambda</code>	Parameter of the loss function (default=0.9).
<code>eps</code>	Parameter of the loss function (if NULL, fixed to 0.01 times the standard deviation of y).
<code>nu</code>	Parameter of the loss function to avoid a division par zero (default=1e-16).
<code>optimProto</code>	If TRUE (default), the initial prototypes are optimized.
<code>verbose</code>	If TRUE (default) intermediate results are displayed.
<code>options</code>	Parameters of the optimization algorithm (see ENNreg).
<code>opt.rmsprop</code>	Parameters of the RMSprop algorithm (see ENNreg).

Details

Either the validation set (a vector of indices), or the number `nval` of validation instances must be provided. Arguments `options` and `opt.rmsprop` are passed to function [ENNreg](#).

Value

A list with three components:

`xi` Optimal value of ξ .

`rho` Optimal value of ρ .

`RMS` Matrix of root mean squared error values.

References

Thierry Denoeux. An evidential neural network model for regression based on random fuzzy numbers. In "Belief functions: Theory and applications (proc. of BELIEF 2022)", pages 57-66, Springer, 2022.

Thierry Denoeux. Quantifying prediction uncertainty in regression using random fuzzy sets: the ENNreg model. IEEE Transactions on Fuzzy Systems, Vol. 31, Issue 10, pages 3690-3699, 2023.

See Also

[ENNreg](#), [ENNreg_cv](#)

Examples

```
# Boston dataset

library(MASS)
X<-as.matrix(scale(Boston[,1:13]))
y<-Boston[,14]
set.seed(220322)
n<-nrow(Boston)
hold.out<-ENNreg_holdout(X,y,K=30,nval=round(n/3),XI=c(0.1,1,10),RHO=c(0.1,1,10))
hold.out$RMS
```

ENNreg_init

Parameter initialization for the ENNreg model

Description

ENNreg_init returns initial parameter values for the ENNreg model.

Usage

```
ENNreg_init(X, y, K, nstart = 100, c = 1)
```

Arguments

X	Input matrix of size $n \times p$, where n is the number of objects and p the number of attributes.
y	Vector of length n containing observations of the response variable.
K	Number of prototypes.
nstart	Number of random starts of the k-means algorithm (default: 100)
c	Multiplicative coefficient applied to scale parameter gamma (default: 1)

Details

Prototypes are initialized by the k-means algorithm.

Value

An object of class "ENNreg", which can be passed to function [ENNreg](#).

Author(s)

Thierry Denoeux.

See Also[ENNreg](#)**Examples**

```
## Boston dataset
library(MASS)
attach(Boston)
X <- as.matrix(scale(Boston[,1:13]))
y <- Boston[,14]
psi <- ENNreg_init(X,y,K=30)
```

evreg

*evreg: A package for evidential regression***Description**

The evreg package implements ENNreg (Denoeux 2022, 2023b), a neural network model for regression in which prediction uncertainty is quantified by Gaussian random fuzzy numbers (GRFNs), a newly introduced family of random fuzzy subsets of the real line that generalizes both Gaussian random variables and Gaussian possibility distributions (Denoeux, 2023a). The evreg package contains functions for training the ENNreg model, tuning hyperparameters by cross-validation or the hold-out method, and making predictions. It also contains utilities for making calculations with GRFNs (such as, e.g., computing the degrees of belief and plausibility of an interval, or combining GRFNs). The package consists in the following main functions:

ENNreg Training of the ENNreg model**ENNreg_init** Initializing a ENNreg model**ENNreg_cv** Hyperparameter tuning using cross-validation**ENNreg_holdout** Hyperparameter tuning using cross-validation using hold-out**predict.ENNreg** Prediction with a trained ENNreg model**intervals** Computation of probabilistic and belief prediction intervals**Bel** Degree of belief of a real interval**Pl** Degree of plausibility of a real interval**Belint** Computation of an interval with given degree of belief**pl_contour** Plausibility contour function of a GRFN**combination_GRFN** Combination of two GRFNs**References**

Thierry Denoeux. An evidential neural network model for regression based on random fuzzy numbers. In "Belief functions: Theory and applications (proc. of BELIEF 2022)", pages 57-66, Springer, 2022.

Thierry Denoeux. Quantifying prediction uncertainty in regression using random fuzzy sets: the ENNreg model. IEEE Transactions on Fuzzy Systems, Vol. 31, Issue 10, pages 3690-3699, 2023.

Thierry Denoeux. Reasoning with fuzzy and uncertain evidence using epistemic random fuzzy sets: general framework and practical models. Fuzzy Sets and Systems, Vol. 453, Pages 1-36, 2023.

See Also

[ENNreg](#), [ENNreg_init](#), [ENNreg_cv](#), [ENNreg_holdout](#), [predict.ENNreg](#), [intervals](#), [intervals](#), [Bel](#), [Pl](#), [Belint](#), [pl_contour](#), [combination_GRFN](#).

intervals

Computation of prediction intervals from a trained ENNreg model

Description

`intervals` computes probabilistic and belief prediction intervals from a prediction object returned by function [predict.ENNreg](#).

Usage

```
intervals(pred, level = 0.9, yt = NULL)
```

Arguments

<code>pred</code>	Prediction object returned by function predict.ENNreg .
<code>level</code>	Level of the prediction interval (between 0 and 1).
<code>yt</code>	Optional vector of test response values.

Value

A list with four elements:

INTP Matrix (n,2) of probabilistic prediction intervals.

INTBel Matrix (n,2) of belief prediction intervals.

coverage.P Estimated coverage rate of the probabilistic intervals (if `yt` is provided).

coverage.Bel Estimated coverage rate of the belief intervals (if `yt` is provided).

See Also

[predict.ENNreg](#), [ENNreg](#)

Examples

```
library(MASS)

X<-as.matrix(scale(Boston[,1:13]))
y<-Boston[,14]
set.seed(220322)
n<-nrow(Boston)
ntrain<-round(0.7*n)
train <-sample(n,ntrain)
fit <- ENNreg(X[train,],y[train],K=30)
pred<-predict(fit,newdata=X[-train,],yt=y[-train])
```

```
int<- intervals(pred,level=0.95,y[-train])
print(c(int$coverage.P,int$coverage.Bel))
```

PI

Degree of plausibility of interval for a Gaussian random fuzzy number

Description

PI computes the degree of plausibility of an interval $[x,y]$ for a given Gaussian random fuzzy number.

Usage

```
PI(x, y, GRFN)
```

Arguments

x	The lower bound of the interval (may be a vector).
y	The upper bound of the interval (may be a vector).
GRFN	A Gaussian random fuzzy number, encoded as a list with components mu, sig and h.

Value

The degree of plausibility of the interval.

References

Thierry Denoeux. Reasoning with fuzzy and uncertain evidence using epistemic random fuzzy sets: general framework and practical models. Fuzzy Sets and Systems, Vol. 453, Pages 1-36, 2023.

See Also

[Belint](#), [Bel](#), [pl_contour](#), [combination_GRFN](#)

Examples

```
pl<-PI(1,2,list(mu=2,sig=1,h=2))
print(pl)
```

`pl_contour`*Contour function of a Gaussian random fuzzy number*

Description

`pl_contour` computes the degree of plausibility of any number x for a given Gaussian random fuzzy number.

Usage

```
pl_contour(x, GRFN)
```

Arguments

<code>x</code>	The input value (can be a vector).
<code>GRFN</code>	A Gaussian random fuzzy number, encoded as a list with components <code>mu</code> , <code>sig</code> and <code>h</code> .

Details

`pl_contour(x,GRFN)` returns the same value as `Pl(x,x,GRFN)`, but is more efficient.

Value

The degree of plausibility of x .

References

Thierry Denoeux. Reasoning with fuzzy and uncertain evidence using epistemic random fuzzy sets: general framework and practical models. *Fuzzy Sets and Systems*, Vol. 453, Pages 1-36, 2023.

See Also

[Pl](#), [Bel](#), [Belint](#)

Examples

```
pl<-pl_contour(1,list(mu=2,sig=1,h=2))
print(pl)
```

predict.ENNreg *Prediction method for the ENNreg model*

Description

Predicted values based on a trained ENNreg model (object of class "ENNreg").

Usage

```
## S3 method for class 'ENNreg'
predict(object, newdata, yt = NULL, ...)
```

Arguments

object	An object of type "ENNreg"
newdata	Input matrix of attributes for test data
yt	Optional test response vector
...	Further arguments passed to or from other methods

Value

Predictions for the new data, coded as a list with the following components:

mux Predicted means

sigx Predicted standard deviations.

hx Prediction precisions.

Einf Lower expectation.

Esup Upper expectations

NLL Negative log likelihood (computed only if yt is provided).

RMS Root mean squared error (computed only if yt is provided).

See Also

[ENNreg](#), [ENNreg_init](#)

Examples

```
# Boston dataset

library(MASS)
X<-as.matrix(scale(Boston[,1:13]))
y<-Boston[,14]
set.seed(220322)
n<-nrow(Boston)
ntrain<-round(0.7*n)
train <-sample(n,ntrain)
```



```
fit <- ENNreg(X[train,],y[train],K=30)
pred<-predict(fit,newdata=X[-train,],yt=y[-train])
plot(y[-train],pred$mu,xlab="observed response",ylab="predicted response")
```

Index

Bel, [2](#), [3](#), [4](#), [13–15](#)

Belint, [2](#), [3](#), [13–15](#)

combination_GRFN, [2](#), [4](#), [13](#), [14](#)

ENNreg, [5](#), [8–13](#), [16](#)

ENNreg_cv, [7](#), [7](#), [11](#), [13](#)

ENNreg_holdout, [7](#), [9](#), [9](#), [13](#)

ENNreg_init, [5](#), [7](#), [11](#), [13](#), [16](#)

evreg, [12](#)

evreg-package (evreg), [12](#)

intervals, [13](#), [13](#)

pl, [2–4](#), [13](#), [14](#), [15](#)

pl_contour, [2–4](#), [13](#), [14](#), [15](#)

predict.ENNreg, [7](#), [13](#), [16](#)