Package 'CaDENCE'

July 2, 2014

Type Package

Title Conditional Density Estimation Network Construction and Evaluation

Version 1.2.1

Author Alex J. Cannon

Maintainer Alex J. Cannon <acannon@eos.ubc.ca>

Description A probabilistic extension of the standard multi-layer perceptron neural network

License GPL-2

Suggests boot, pso

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2013-06-28 07:11:48

R topics documented:

`CaDENCE-package` *Conditional Density Estimation Network Construction and Evaluation (CaDENCE)*

Description

A conditional density estimation network (CDEN) is a probabilistic extension of the standard multilayer perceptron neural network (MLP) (Neuneier et al., 1994). A CDEN model allows users to estimate parameters of a specified probability density function conditioned upon values of a set of predictors using the MLP architecture. The result is a flexible model for the mean, the variance, exceedance probabilities, prediction intervals, etc. from the specified conditional distribution. Because the CDEN is based on the MLP, nonlinear relationships, including those involving complicated interactions between predictors, can be described by the modelling framework. The CaDENCE (Conditional Density Estimation Network Creation & Evaluation) package provides routines for creating and evaluating CDEN models in the R programming language.

Details

Procedures for fitting CaDENCE models are provided by `cadence.fit`, which relies on the standard `optim` function, the CaDENCE `rprop` function, or, optionally, the `psoptim` function from the `pso` package. Once a model has been developed, `cadence.predict` is used to evaluate the distribution parameters as a function of predictors.

The package also provides a variety of zero-inflated distributions, including the Bernoulli-gamma (`bgamma`), Bernoulli-Weibull (`bweibull`), Bernoulli-Pareto 2 (`bpareto2`), and Bernoulli-lognormal (`blnorm`), for use in the CaDENCE models.

`gam.style`, `dummy.code`, `xval.buffer`, and `rbf` are helper functions that may be useful for data preprocessing, model evaluation, and interpretation of fitted relationships.

Most other functions are used internally and should not normally need to be called directly by the user.

Author(s)

Alex J. Cannon

Maintainer: Alex J. Cannon <acannon@eos.ubc.ca>

References

Cannon, A.J., 2012. Neural networks for probabilistic environmental prediction: Conditional Density Estimation Network Creation & Evaluation (CaDENCE) in R. Computers & Geosciences 41: 126-135. doi:10.1016/j.cageo.2011.08.023

Neuneier, R., F. Hergert, W. Finnoff, and D. Ormoneit, 1994., Estimation of conditional densities: a comparison of neural network approaches. In: M. Marinaro and P. Morasso (eds.), Proceedings of ICANN 94, Berlin, Springer, p. 689-692.

---

bgamma *Bernoulli-gamma distribution*

Description

Functions implementing the Bernoulli-gamma distribution, in which zero values occur with probability `1-prob` and non-zero values follow a gamma distribution with `scale` and `shape` parameters. `dbgamma` gives a probability density function (pdf), `pbgamma` gives the cumulative distribution function (cdf), `qbgamma` gives the quantile function (inverse cdf), and `rbgamma` is used for generating random variates.

Usage

```
dbgamma(x, prob, scale, shape)
pbgamma(q, prob, scale, shape)
qbgamma(p, prob, scale, shape)
rbgamma(n, prob, scale, shape)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of cumulative probabilities.
`n`	number of random samples.
`prob`	probability of a non-zero value.
`scale`	scale parameter of the gamma distribution.
`shape`	shape parameter of the gamma distribution.

Value

`dbgamma` gives the pdf, `pbgamma` gives the cdf, `qbgamma` gives the inverse cdf (or quantile function), and `rbgamma` generates random deviates.

References

Cannon, A.J., 2008. Probabilistic multi-site precipitation downscaling by an expanded Bernoulli-gamma density network. Journal of Hydrometeorology, 9(6): 1284-1300.

See Also

`dgamma`, `bweibull`, `bpareto2`, `blnorm`

Examples

```
plot(rbgamma(365, prob = 0.2, scale = 1, shape = 1), type = "h")
```

blnorm *Bernoulli-lognormal distribution*

Description

Functions implementing the Bernoulli-lognormal distribution, in which zero values occur with probability `1-prob` and non-zero values follow a lognormal distribution with `meanlog` and `sdlog` parameters. `dblnorm` gives a probability density function (pdf), `pblnorm` gives the cumulative distribution function (cdf), `qblnorm` gives the quantile function (inverse cdf), and `rblnorm` is used for generating random variates.

Usage

```
dblnorm(x, prob, meanlog, sdlog)
pblnorm(q, prob, meanlog, sdlog)
qblnorm(p, prob, meanlog, sdlog)
rblnorm(n, prob, meanlog, sdlog)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of cumulative probabilities.
`n`	number of random samples.
`prob`	probability of a non-zero value.
`meanlog`	meanlog parameter of the lognormal distribution.
`sdlog`	sdlog parameter of the lognormal distribution.

Value

`dblnorm` gives the pdf, `pblnorm` gives the cdf, `qblnorm` gives the inverse cdf (or quantile function), and `rblnorm` generates random variates.

See Also

`dlnorm`, `bweibull`, `bpareto2`, `bgamma`

Examples

```
plot(rblnorm(365, prob = 0.2, meanlog = 1, sdlog = 1), type = "h")
```

bweibull *Bernoulli-Weibull distribution*

Description

Functions implementing the Bernoulli-Weibull distribution, in which zero values occur with probability `1-prob` and non-zero values follow a Weibull distribution with `scale` and `shape` parameters. `dbweibull` gives a probability density function (pdf), `pbweibull` gives the cumulative distribution function (cdf), `qbweibull` gives the quantile function (inverse cdf), and `rbweibull` is used for generating random variates.

Usage

```
dbweibull(x, prob, scale, shape)
pbweibull(q, prob, scale, shape)
qbweibull(p, prob, scale, shape)
rbweibull(n, prob, scale, shape)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of cumulative probabilities.
`n`	number of random samples.
`prob`	probability of a non-zero value.
`scale`	scale parameter of the weibull distribution.
`shape`	shape parameter of the weibull distribution.

Value

`dbweibull` gives the pdf, `pbweibull` gives the cdf, `qbweibull` gives the inverse cdf (or quantile function), and `rbweibull` generates random variates.

See Also

`dweibull`, `bgamma`, `bpareto2`, `blnorm`

Examples

```
plot(rbweibull(365, prob = 0.2, scale = 1, shape = 1), type = "h")
```

cadence.cost *Cost function for CDEN model fitting*

Description

The maximum likelihood cost function used for CDEN model fitting. Calculates the negative of the logarithm of the likelihood. A normal distribution prior can be set for the magnitude of the input-hidden layer weights, thus leading to weight penalty regularization.

Usage

```
cadence.cost(weights, x, y, n.hidden, hidden.fcn, distribution, sd.norm,
             valid)
```

Arguments

weights	weight vector of length returned by cadence.initialize.
x	matrix with number of rows equal to the number of samples and number of columns equal to the number of predictor variables.
y	column matrix of predictand values with number of rows equal to the number of samples.
n.hidden	number of hidden nodes in the CDEN model.
hidden.fcn	hidden layer transfer function.
distribution	a list that describes the probability density function associated with the predictand.
sd.norm	sd parameter for normal distribution prior for the magnitude of input-hidden layer weights; equivalent to weight penalty regularization.
valid	valid logical vector indicating which weights are non-zero or fixed at zero, i.e., due to use of parameters.fixed in distribution.

See Also

cadence.fit, optim, rprop

cadence.fit *Fit a CDEN model*

Description

Fit a CDEN model via nonlinear optimization of the maximum likelihood cost function.

Usage

```
cadence.fit(x, y, iter.max = 500, n.hidden = 2, hidden.fcn = tanh,
            distribution = NULL, sd.norm = Inf, init.range = c(-0.5, 0.5),
            method = c("optim", "psoptim", "Rprop"), n.trials = 1,
            trace = 0, maxit.Nelder = 2000, trace.Nelder = 0,
            swarm.size = NULL, vectorize = TRUE,
            delta.0 = 0.1, delta.min = 1e-06, delta.max = 50, epsilon = 1e-08,
            range.mult = 2, step.tol = 1e-08, f.target = -Inf,
            max.exceptions = 500)
```

Arguments

x	matrix with number of rows equal to the number of samples and number of columns equal to the number of predictor variables.
y	column matrix of predictand values with number of rows equal to the number of samples.
iter.max	maximum number of iterations of the optimization function.
n.hidden	number of hidden nodes in the CDEN model; can be a vector indicating a range of values to fit.
hidden.fcn	hidden layer transfer function.
distribution	a list that describes the probability density function associated with the predictand.
sd.norm	sd parameter for normal distribution prior for the magnitude of input-hidden layer weights; equivalent to weight penalty regularization.
init.range	range for random weights on [min(init.range), max(init.range)]
method	specifies the optimization method used to minimize cadence.cost; must be chosen from c("optim", "psoptim", "Rprop").
n.trials	number of repeated trials used to avoid shallow local minima during optimization.
trace	the level of printing which is done during optimization. A value of 0 suppresses any progress reporting.
maxit.Nelder	maximum number of iterations of the Nelder-Mead optimization function prior to main calling method.
trace.Nelder	the level of printing which is done during Nelder-Mead optimization. A value of 0 suppresses any progress reporting.
swarm.size	swarm.size if psoptim is used for optimization.
vectorize	vectorize if psoptim is used for optimization.
delta.0	size of the initial update-value if rprop is used for optimization.
delta.min	minimum value for the adaptive update-value if rprop is used for optimization.
delta.max	maximum value for the adaptive update-value if rprop is used for optimization.
epsilon	step-size used in the finite difference calculation of the gradient if rprop is used for optimization.

range.mult	if psoptim is used for optimization, sets the search space boundaries to range.mult times the range of weights found by the Nelder-Mead algorithm.
step.tol	convergence criterion if rprop is used for optimization. Optimization will stop if the change in f over the previous three iterations falls below this value.
f.target	target value of f if rprop is used for optimization. Optimization will stop if f falls below this value.
max.exceptions	maximum number of repeated exceptions allowed during optimization.

Details

Fit a CDEN model by optimizing the maximum likelihood cost function. Optimization relies on the standard optim function, the built-in rprop function, or, optionally, the psoptim function from the pso package.

The hidden layer transfer function hidden.fcn should be set to tanh for a nonlinear model and to identity for a linear model. In the nonlinear case, the number of hidden nodes n.hidden controls the overall complexity of the model. The predictand distribution is set by the distribution argument. Parameters of the specified distribution can be held constant via the parameters.fixed element distribution. Weight penalty regularization for the magnitude of the input-hidden layer weights can be applied by setting sd.norm to a value less than Inf.

The distribution argument in cadence.fit is the most important part of the CaDENCE modelling framework and has been designed to be as flexible as possible. To this end, distribution is a list with three mandatory elements: density.fcn, which specifies the R density function for the predictand distribution; parameters, which specifies the names of the parameters used as arguments in density.fcn; and output.fcns, which specifies the functions used to constrain the density function parameters to their allowable ranges (i.e., inverse link functions). If not specified, distribution defaults to a normal distribution. Note: the order of parameters and output.fcns must match the order of arguments in the specified density.fcn.

A fourth element of distribution, parameters.fixed, is optional. Setting parameters.fixed="sd" for the normal distribution would, for example, force the sd parameter to take a constant value.

Samples of distribution lists for a variety of probability distributions are given below for reference:

```
# normal distribution
norm.distribution <- list(density.fcn = dnorm,
                          parameters = c("mean", "sd"),
                          parameters.fixed = NULL,
                          output.fcns = c(identity, exp))

# lognormal distribution
lnorm.distribution <- list(density.fcn = dlnorm,
                           parameters = c("meanlog", "sdlog"),
                           parameters.fixed = NULL,
                           output.fcns = c(identity, exp))

# exponential distribution
exp.distribution <- list(density.fcn = dexp,
                         parameters = c("rate"),
```

```
                          parameters.fixed = NULL,
                          output.fcns = c(exp))

# Poisson distribution
poisson.distribution <- list(density.fcn = dpois,
                             parameters = c("lambda"),
                             parameters.fixed = NULL,
                             output.fcns = c(exp))

# Bernoulli-gamma distribution
bgamma.distribution <- list(density.fcn = dbgamma,
                            parameters = c("prob", "scale", "shape"),
                            parameters.fixed = NULL,
                            output.fcns = c(logistic, exp, exp))

# Bernoulli-Weibull distribution
bweibull.distribution <- list(density.fcn = dbweibull,
                              parameters = c("prob", "scale", "shape"),
                              parameters.fixed = NULL,
                              output.fcns = c(logistic, exp, exp))

# Bernoulli-lognormal distribution
blnorm.distribution <- list(density.fcn = dblnorm,
                            parameters = c("prob", "meanlog", "sdlog"),
                            parameters.fixed = NULL,
                            output.fcns = c(logistic, identity, exp))

# Bernoulli-Pareto 2 distribution
bpareto2.distribution <- list(density.fcn = dbpareto2,
                          parameters = c("prob", "scale", "shape"),
                          parameters.fixed = NULL,
                          output.fcns = c(logistic, exp, exp))

# beta distribution
beta.distribution <- list(density.fcn=dbeta,
                          parameters=c("shape1", "shape2"),
                          parameters.fixed=NULL,
                          output.fcns=c(exp, exp))

# truncated normal distribution with lower = 0
library(msm)
dtnormal <- function(x, mean, sd) dtnorm(x, mean, sd, lower = 0)
dtnorm.distribution <- list(density.fcn = dtnormal,
                            parameters = c("mean", "sd"),
                            parameters.fixed = NULL,
                            output.fcns = c(identity, exp))

# mixture of two normal distributions
```

```
library(nor1mix)
dnormix <-
function(x, mu1, mu2, sigs1, sigs2, w1){
    if(length(x) > 1){
        dens <- mapply(dnormix, x, mu1 = mu1, mu2 = mu2,
                       sigs1 = sigs1, sigs2 = sigs2, w1 = w1)
    } else{
        mix <- norMix(mu = c(mu1, mu2), sig2 = c(sigs1, sigs2),
                      w = c(w1, 1-w1))
        dens <- dnorMix(x, mix)
    }
    dens
}
normix.distribution <- list(density.fcn = dnormix,
                            parameters = c("mu1", "mu2", "sigs1",
                                           "sigs2", "w1"),
                            parameters.fixed = NULL,
                            output.fcns = c(identity, identity,
                                            exp, exp, logistic))
```

Values of the Akaike information criterion with small sample size correction (AICc), and Bayesian information criterion (BIC) are calculated to assist in model selection. It is possible for such criteria to fail in the face of overfitting, for example with a nonlinear model and `n.hidden` set too high, as the distribution may converge on one or more samples. This can usually be diagnosed by inspecting the scale parameter of the distribution for near zero values. In this case, one can apply a weight penalty (via `sd.norm`), although this rules out the straightforward use of AICc/BIC for model selection as the effective number of model parameters will no longer equal the number of weights in the CDEN model.

Note: values of `x` need not be standardized or rescaled by the user. Predictors are automatically scaled to zero mean and unit standard deviation and are rescaled by `cadence.predict`.

Value

a list of with number of elements equal to the length of `n.hidden`; each list consists of:

`W1`	input-hidden layer weights
`W2`	hidden-output layer weights. Attributes indicating the mean and standard deviation of columns of `x`; the value of `hidden.fcn`; the valud of `hidden.fcn`; the negative log-likelihood `NLL`; the number of model parameters `k`; the value of the weight penalty `penalty` (if `sd.norm` is less than infinity); the value of the `BIC`, `AIC`, and `AICc` cost-complexity criteria; and the predictand `distribution` list are also returned.

References

Cannon, A.J., 2012. Neural networks for probabilistic environmental prediction: Conditional Density Estimation Network Creation & Evaluation (CaDENCE) in R. Computers & Geosciences 41: 126-135. doi:10.1016/j.cageo.2011.08.023

Neuneier, R., F. Hergert, W. Finnoff, and D. Ormoneit, 1994., Estimation of conditional densities: a comparison of neural network approaches. In: M. Marinaro and P. Morasso (eds.), Proceedings of ICANN 94, Berlin, Springer, p. 689-692.

See Also

cadence.predict, optim, rprop, xval.buffer, logistic

Examples

```
data(FraserSediment)
set.seed(1)
lnorm.distribution <- list(density.fcn = dlnorm,
                           parameters = c("meanlog", "sdlog"),
                           parameters.fixed = NULL,
                           output.fcns = c(identity, exp))
fit <- cadence.fit(x = FraserSediment$x.1970.1976[c(TRUE, rep(FALSE, 19)),],
                   y = FraserSediment$y.1970.1976[c(TRUE, rep(FALSE, 19)),,
                   drop=FALSE], n.hidden = 3, n.trials = 1,
                   maxit.Nelder = 100, trace.Nelder = 1, hidden.fcn = tanh,
                   distribution = lnorm.distribution, trace = 1)
pred <- cadence.predict(x = FraserSediment$x.1977.1979, fit = fit)
matplot(pred, type = "l")
```

---

cadence.initialize *Initialize a CDEN weight vector*

Description

Random initialization of the weight vector used during fitting of the CDEN model.

Usage

```
cadence.initialize(x, n.hidden, init.range, distribution)
```

Arguments

x	matrix with number of rows equal to the number of samples and number of columns equal to the number of predictors.
n.hidden	number of hidden nodes in the CDEN model.
init.range	range for random weights on [min(init.range), max(init.range)]
distribution	list used to specify the predictand distribution

cadence.predict *Predict conditional distribution parameters from a fitted CDEN model*

Description

Predict conditional distribution parameters from a fitted CDEN model. The returned value is a matrix with columns corresponding to the parameters of the probability distribution specified in the `distribution` argument passed to `cadence.fit`.

Usage

```
cadence.predict(x, fit)
```

Arguments

`x`	matrix with number of rows equal to the number of samples and number of columns equal to the number of predictor variables.
`fit`	list returned by `cadence.fit`.

Value

a matrix with number of rows equal to that of `x` and columns corresponding to the parameters of the `distribution` argument passed to `cadence.fit`.

See Also

`cadence.fit`, `optim`, `rprop`

Examples

```
data(FraserSediment)
lnorm.distribution.fixed <- list(density.fcn = dlnorm,
                                 parameters = c("meanlog", "sdlog"),
                                 parameters.fixed = "sdlog",
                                 output.fcns = c(identity, exp))
fit <- cadence.fit(x = FraserSediment$x.1970.1976,
                   y = FraserSediment$y.1970.1976,
                   hidden.fcn = identity, maxit.Nelder = 100,
                   trace.Nelder = 1, trace = 1,
                   distribution = lnorm.distribution.fixed)
pred <- cadence.predict(x = FraserSediment$x.1977.1979, fit = fit)
matplot(pred, type = "l")
```

dummy.code *Convert a factor to a matrix of dummy codes*

Description

Converts a factor (categorical) variable to a matrix of dummy codes using a 1 of C-1 binary coding scheme.

Usage

```
dummy.code(x)
```

Arguments

x	a factor variable.

Value

a matrix with the number of rows equal to the number of cases in x and the number of columns equal to one minus the number of factors in x. The last factor serves as the reference group.

Examples

```
print(dummy.code(iris$Species))
```

FraserSediment *Sediment and stream discharge data for Fraser River at Hope*

Description

A dataset consisting of daily observations of suspended sediment concentration (SSC) (mg/L) and stream discharge (Q) (cu. m/s) for the years 1970-1979 at the Fraser River at Hope station in British Columbia, Canada (Water Survey of Canada station 08MF005). Samples are split into a seven year training period (1970-1976) and a three year testing period (1977-1979).

In terms of structure, FraserSediment is a list with four elements: x.1970.1976, y.1970.1976, x.1977.1979, and y.1977.1979. x.1970.1976 and x.1977.1979 are matrices with predictor variables: logQ, log-transformed Q; and dQ5, dQ30, and dQ90, 5-, 30-, and 90-day moving averages of daily changes in Q. y.1970.1976 and y.1977.1979 are matrices with the predictand variable SSC.

Examples

```
data(FraserSediment)
pairs(cbind(FraserSediment$x.1970.1976, FraserSediment$y.1970.1976))
```

gam.style *GAM-style effects plots for interpreting CDEN models*

Description

GAM-style effects plots provide a graphical means of interpreting relationships between predictors and conditional pdf parameter values predicted by a CDEN. From Plate et al. (2000): The effect of the ith input variable at a particular input point Delta.i.x is the change in f resulting from changing X1 to x1 from b1 (the baseline value [...]) while keeping the other inputs constant. The effects are plotted as short line segments, centered at (x.i, Delta.i.x), where the slope of the segment is given by the partial derivative. Variables that strongly influence the function value have a large total vertical range of effects. Functions without interactions appear as possibly broken straight lines (linear functions) or curves (nonlinear functions). Interactions show up as vertical spread at a particular horizontal location, that is, a vertical scattering of segments. Interactions are present when the effect of a variable depends on the values of other variables.

Usage

```
gam.style(x, fit, column, baseline = mean(x[,column]),
          additive.scale = FALSE, epsilon = 1e-5,
          seg.len = 0.02, seg.cols = "black", plot = TRUE,
          return.results = FALSE, ...)
```

Arguments

x	matrix with number of rows equal to the number of samples and number of columns equal to the number of predictor variables.
fit	element from list returned by cadence.fit.
column	column of x for which effects plots should be returned.
baseline	value of x[,column] to be used as the baseline for calculation of predictor effects; defaults to mean(x[,column]).
additive.scale	if TRUE then predictor effects and partial derivatives are calculated before the inverse link functions for the distribution parameters are applied; if FALSE (the default) then values are calculated after the inverse link functions are applied.
epsilon	step-size used in the finite difference calculation of the partial derivatives.
seg.len	length of effects line segments expressed as a fraction of the range of x[,column].
seg.cols	colors of effects line segments.
plot	if TRUE (the default) then an effects plots for each distribution parameter is produced.
return.results	if TRUE then values of effects and partial derivatives for each distribution parameter are returned.
...	further arguments to be passed to plot.

Value

A list with elements:

effects	a matrix of predictor effects.
partials	a matrix of predictor partial derivatives.

References

Cannon, A.J. and I.G. McKendry, 2002. A graphical sensitivity analysis for interpreting statistical climate models: Application to Indian monsoon rainfall prediction by artificial neural networks and multiple linear regression models. International Journal of Climatology, 22:1687-1708.

Plate, T., J. Bert, J. Grace, and P. Band, 2000. Visualizing the function computed by a feedforward neural network. Neural Computation, 12(6): 1337-1354.

See Also

cadence.fit, cadence.predict

Examples

```
data(FraserSediment)
set.seed(1)
lnorm.distribution <- list(density.fcn = dlnorm,
                           parameters = c("meanlog", "sdlog"),
                           parameters.fixed = NULL,
                           output.fcns = c(identity, exp))
x <- FraserSediment$x.1970.1976[c(TRUE, rep(FALSE, 19)),]
y <- FraserSediment$y.1970.1976[c(TRUE, rep(FALSE, 19)),,drop=FALSE]
fit.nlin <- cadence.fit(x, y, n.hidden = 3, n.trials = 1,
                        hidden.fcn = tanh, distribution =
                        lnorm.distribution, maxit.Nelder = 100,
                        trace.Nelder = 1, trace = 1)
fit.lin <- cadence.fit(x, y, hidden.fcn = identity, n.trials = 1,
                       distribution = lnorm.distribution,
                       maxit.Nelder = 100, trace.Nelder = 1,
                       trace = 1)
gam.style(x, fit = fit.nlin[[1]], column = 1,
          main = "Nonlinear")
gam.style(x, fit = fit.lin[[1]], column = 1,
          additive.scale = TRUE,
          main = "Linear (additive.scale = TRUE)")
```

---

logistic *Logistic sigmoid function*

Description

logistic computes a logistic sigmoid (S-shaped) function bounded between 0 and 1.

Usage

```
logistic(x)
```

Arguments

x	a numeric vector

---

pareto2 *Pareto 2 (Lomax) and Bernoulli-Pareto 2 distributions*

Description

Functions implementing the Pareto 2 (Lomax) and Bernoulli-Pareto 2 distributions. In the latter case, zero values occur with probability 1-prob and non-zero values follow the Pareto 2 distribution with scale and shape parameters. dpareto2 and dbpareto2 give the probability density functions (pdf); ppareto2 and pbpareto2 give the cumulative distribution functions (cdf); qpareto2 and qbpareto2 give the quantile functions (inverse cdfs), and rpareto2 and rbpareto2 are used for generating random variates.

Usage

```
dpareto2(x, scale, shape)
ppareto2(q, scale, shape)
qpareto2(p, scale, shape)
rpareto2(n, scale, shape)
dbpareto2(x, prob, scale, shape)
pbpareto2(q, prob, scale, shape)
qbpareto2(p, prob, scale, shape)
rbpareto2(n, prob, scale, shape)
```

Arguments

x, q	vector of quantiles.
p	vector of cumulative probabilities.
n	number of random samples.
prob	probability of a non-zero value.
scale	scale parameter of the pareto2 distribution.
shape	shape parameter of the pareto2 distribution.

Value

dpareto2 and dbpareto2 gives the pdfs; ppareto2 and pbpareto2 gives the cdfs; qpareto2 and qbpareto2 gives the inverse cdfs (or quantile functions); and rpareto2 and rbpareto2 generate random variates.

References

Arnold, B.C., 1983. The Pareto Distributions, International Co-operative Publishing House, Fairland, MD.

Lomax, K.S., 1954. Business failures: another example of the analysis of failure data. Journal of the American Statistical Association, 49(268): 847-852.

See Also

bgamma, bweibull, blnorm

Examples

```
plot(rbpareto2(365, prob = 0.2, scale = 1, shape = 1), type = "h")
```

---

rbf *Radial basis function kernel*

Description

Evaluate a kernel matrix based on the radial basis function kernel. Can be used in conjunction with cadence.fit with hidden.fcn set to identity and sd.norm set to a value less than infinity to implement a kernel CDEN model.

Usage

```
rbf(x, x.basis, sigma)
```

Arguments

x	matrix with number of rows equal to the number of samples and number of columns equal to the number of predictors.
x.basis	matrix with number of rows equal to the number of basis functions and number of columns equal to the number of predictors.
sigma	kernel width

Value

kernel matrix with number of rows equal to the number of samples and number of columns equal to the number of basis functions.

See Also

cadence.fit

rprop *Resilient backpropagation (Rprop) optimization algorithm*

Description

From Riedmiller (1994): Rprop stands for 'Resilient backpropagation' and is a local adaptive learning scheme. The basic principle of Rprop is to eliminate the harmful influence of the size of the partial derivative on the weight step. As a consequence, only the sign of the derivative is considered to indicate the direction of the weight update. The size of the weight change is exclusively determined by a weight-specific, so called 'update-value'.

This function implements the iRprop+ algorithm from Igel and Huesken (2003).

Usage

```
rprop(w, f, iterlim = 100, print.level = 1, delta.0 = 0.1,
      delta.min = 1e-06, delta.max = 50, epsilon = 1e-08,
      step.tol = 1e-06, f.target = -Inf, ...)
```

Arguments

w	the starting parameters for the minimization.
f	the function to be minimized. If the function value has an attribute called gradient, this will be used in the calculation of updated parameter values. Otherwise, numerical derivatives will be used.
iterlim	the maximum number of iterations before the optimization is stopped.
print.level	the level of printing which is done during optimization. A value of 0 suppresses any progress reporting, whereas positive values report the value of f and the mean change in f over the previous three iterations.
delta.0	size of the initial Rprop update-value.
delta.min	minimum value for the adaptive Rprop update-value.
delta.max	maximum value for the adaptive Rprop update-value.
epsilon	step-size used in the finite difference calculation of the gradient.
step.tol	convergence criterion. Optimization will stop if the change in f over the previous three iterations falls below this value.
f.target	target value of f. Optimization will stop if f falls below this value.
...	further arguments to be passed to f.

Value

A list with elements:

par	The best set of parameters found.
value	The value of f corresponding to par.
gradient	An estimate of the gradient at the solution found.

References

Igel, C. and M. Huesken, 2003. Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing 50: 105-123.

Riedmiller, M., 1994. Advanced supervised learning in multilayer perceptrons - from backpropagation to adaptive learning techniques. Computer Standards and Interfaces 16(3): 265-278.

`xval.buffer` *Cross-validation indices with a buffer between training/validation datasets*

Description

Caculates training/validation indices for N-fold cross-validation of a dataset. Cross-validation folds are taken as contiguous blocks of cases with an optional buffer to prevent leakage of information between training/validation subsets due to the presence of autocorrelation.

Usage

```
xval.buffer(n.cases, n.xval=5, buffer.length=0)
```

Arguments

`n.cases`	an integer specifying the length of the dataset.
`n.xval`	an integer specifying the desired number of cross-validation folds.
`buffer.length`	an integer specifying the number of cases to be left out as a buffer between the training/validation subsets.

Value

a list with `n.xval` elements, each containing:

`train`	indices of training cases
`valid`	indices of validation cases

References

Shabbar, A. and V. Kharin. 2007. An assessment of cross-validation for estimating skill of empirical seasonal forecasts using a global coupled model simulation. CLIVAR Exchanges. 12(4): 10-12.

Zeng, Z., W.W. Hsieh, A. Shabbar, and W.W. Burrows, 2011. Seasonal prediction of winter extreme precipitation over Canada by support vector regression, Hydrology and Earth System Sciences, 15: 65-74.

Examples

```
print(xval.buffer(100, n.xval = 3, buffer.length = 10))
```

Index