Package ‘GEVcdn’

July 23, 2015

Type Package
Title GEV Conditional Density Estimation Network
Version 1.1.4
Author Alex J. Cannon
Maintainer Alex J. Cannon <acannon@eos.ubc.ca>
Description Implements a flexible nonlinear modelling framework for nonstationary
generalized extreme value analysis in hydroclimatology.
License GPL-2
Depends VGAM
Suggests boot
LazyLoad yes
Repository CRAN
NeedsCompilation no
Date/Publication 2015-07-23 06:58:56

R topics documented:

GEVcdn-package .................................................. 2
gevcdn.bag ......................................................... 3
gevcdn.bootstrap ............................................... 5
gevcdn.cost ....................................................... 8
gevcdn.evaluate ............................................... 9
gevcdn.fit ......................................................... 10
gevcdn.identity ................................................. 13
agevcdn.initialize .......................................... 14
agevcdn.logistic .............................................. 14
agevcdn.reshape .............................................. 15

Index 16
Description

Parameters in a Generalized Extreme Value (GEV) distribution are specified as a function of covariates using a conditional density estimation network (CDN), which is a probabilistic variant of the multilayer perceptron neural network. If the covariate is time or is dependent on time, then the GEV CDN model can be used to perform nonlinear, nonstationary GEV analysis of hydrological or climatological time series. Owing to the flexibility of the neural network architecture, the model is capable of representing a wide range of nonstationary relationships, including those involving interactions between covariates. Model parameters are estimated by generalized maximum likelihood, an approach that is tailored to the estimation of GEV parameters from geophysical time series.

Details

Procedures for fitting GEV CDN models are provided by the functions `gevcdn.fit` and `gevcdn.bag`. Once a model has been developed, `gevcdn.evaluate` is used to evaluate the GEV distribution parameters as a function of covariates. Confidence intervals for GEV parameters and specified quantiles can be estimated using `gevcdn.bootstrap`. All other functions are used internally and should not normally need to be called directly by the user.

Note: the GEV distribution functions are provided by the `VGAM` package. The convention for the sign of the shape parameter is opposite to that used in hydrology and thus differs from Cannon (2010).

Author(s)

Alex J. Cannon

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

References


See Also

VGAM
Fit an ensemble of GEV CDN models via bagging

**Description**

Used to fit an ensemble of GEV CDN models using bootstrap aggregation (bagging) and, optionally, early stopping.

**Usage**

```r
gevcdn.bag(x, y, iter.max = 1000, iter.step = 10,
            n.bootstrap = 30, n.hidden = 3, Th = gevcdn.logistic,
            fixed = NULL, init.range = c(-0.25, 0.25),
            scale.min = .Machine$double.eps, beta.p = 3.3,
            beta.q = 2, sd.norm = Inf,
            method = c("BFGS", "Nelder-Mead"), max.fails = 100,
            silent = TRUE, ...)
```

**Arguments**

- `x` covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of variables.
- `y` column matrix of target values with number of rows equal to the number of samples.
- `iter.max` maximum number of iterations of optimization algorithm.
- `iter.step` number of iterations between which the value of the cost function is calculated on the out-of-bootstrap samples; used during stopped training.
- `n.bootstrap` number of ensemble members trained during bootstrap aggregation (bagging).
- `n.hidden` number of hidden nodes in each GEV CDN ensemble member.
- `Th` hidden layer transfer function; defaults to `gevcdn.logistic`.
- `fixed` vector indicating GEV parameters to be held constant; elements chosen from `c("location", "scale", "shape")`
- `init.range` range for random weights on `[min(init.range), max(init.range)]`
- `scale.min` minimum allowable value for the GEV scale parameter.
- `beta.p` shape1 parameter for shifted beta distribution prior for GEV shape parameter.
- `beta.q` shape2 parameter for shifted beta distribution prior for GEV shape parameter.
- `sd.norm` sd parameter for normal distribution prior for the magnitude of input-hidden layer weights; equivalent to weight penalty regularization.
- `method` optimization method for `optim` function; must be chosen from `c("BFGS", "Nelder-Mead")`.
- `max.fails` maximum number of repeated exceptions allowed during optimization.
- `silent` logical value indicating whether or not cost function information should be reported every `iter.step` iterations.
- `...` additional arguments passed to the `control` list of `optim` function.
Details

Bootstrap aggregation (bagging) (Breiman, 1996) is used as an alternative to `gevcdn.fit` for fitting an ensemble of nonstationary GEV CDN models (Cannon, 2010). Each ensemble member is trained on bootstrapped x and y sample pairs. As an added check on overfitting, early stopping, whereby training is stopped prior to convergence of the optimization algorithm, can be turned on. In this case, the "best" iteration for stopping optimization is chosen based on model performance on out-of-bag samples. Additional details on the bagging/early stopping procedure can be found in Cannon and Whitfield (2001, 2002). Unlike `gevcdn.fit`, where models of increasing complexity are fitted and the one that satisfies some model selection criterion is chosen for final use, model selection in `gevcdn.bag` is implicit. Ensemble averaging and, optionally, early stopping are used to limit model complexity. One need only set n.hidden to avoid underfitting.

Note: values of x and y need not be standardized or rescaled by the user. All variables are automatically scaled to range between 0 and 1 prior to fitting and parameters are automatically rescaled by `gevcdn.evaluate`.

Value

a list of length n.bootstrap with elements consisting of

- $w_1$ input-hidden layer weights
- $w_2$ hidden-output layer weights

Attributes indicating the minimum/maximum values of x and y; the values of Th, fixed, scale.min; a logical value stopped.training indicating whether or not early stopping was used; and the minimum value of the cost function on the out-of-bag samples cost.valid are also returned.

References


See Also

`gevcdn.cost`, `gevcdn.evaluate`, `gevcdn.fit`, `gevcdn.bootstrap`, `dgev`, `optim`

Examples

```r
## Generate synthetic data, quantiles
x <- as.matrix(seq(0.1, 1, length = 50))
loc <- x^2
```
gevcdn.bootstraps

scl <- x/2
shp <- seq(-0.1, 0.3, length = length(x))

set.seed(100)
y <- as.matrix(rgev(length(x), location = loc, scale = scl,
shape = shp))
q <- sapply(c(0.1, 0.5, 0.9), qgev, location = loc, scale = scl,
shape = shp)

### Not run:
### Fit ensemble of models with early stopping turned on

weights.on <- gevcdn.bag(x = x, y = y, iter.max = 100,
iter.step = 10, n.bootstrap = 10,
n.hidden = 2)

parms.on <- lapply(weights.on, gevcdn.evaluate, x = x)

# 10th, 50th, and 90th percentiles
q.10.on <- q.50.on <- q.90.on <- c()
for(i in seq_along(parms.on)){
  q.10.on <- cbind(q.10.on, qgev(p = 0.1,
  location = parms.on[[i]], "location",
  scale = parms.on[[i]], "scale",
  shape = parms.on[[i]], "shape"))
  q.50.on <- cbind(q.50.on, qgev(p = 0.5,
  location = parms.on[[i]], "location",
  scale = parms.on[[i]], "scale",
  shape = parms.on[[i]], "shape"))
  q.90.on <- cbind(q.90.on, qgev(p = 0.9,
  location = parms.on[[i]], "location",
  scale = parms.on[[i]], "scale",
  shape = parms.on[[i]], "shape"))
}

### Plot data and quantiles

matplot(cbind(y, q, rowMeans(q.10.on), rowMeans(q.50.on),
rowMeans(q.90.on)), type = c("b", rep("l", 6)),
1ty = c(1, rep(c(1, 2, 1, 2), 2)),
1wd = c(1, rep(c(3, 2, 3, 2), 2)),
1col = c("red", rep("orange", 3), rep("blue", 3)),
pch = 19, xlab = "x", ylab = "y",
main = "gevcdn.bag (early stopping on)"
)

### End(Not run)

gevcdn.bootstraps  Bootstrapped confidence intervals for GEV CDN parameters and quantiles
Description

Used to assist in the calculation of bootstrapped confidence intervals for GEV location, scale, and shape parameters, as well as for specified quantiles. Residual and parametric bootstrap estimates are supported.

Usage

gevcdn_bootstrap(n.bootstraps, x, y, iter.max = 1000, n.hidden = 2,
    Th = gevcdn.logistic, fixed = NULL,
    init.range = c(-0.25, 0.25),
    scale.min = .Machine$double.eps,
    beta.p = 3.3, beta.q = 2, sd.norm = Inf,
    n.trials = 5, method = c("BFGS", "Nelder-Mead"),
    boot.method = c("residual", "parametric"),
    init.from.prev = TRUE, max.fails = 100,
    probs = c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9,
              0.95, 0.99), ...)

Arguments

n.bootstraps number of bootstrap trials used to calculate confidence intervals.
x covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of variables.
y column matrix of target values with number of rows equal to the number of samples.
iter.max maximum number of iterations of optimization algorithm.
n.hidden number of hidden nodes in each GEV CDN ensemble member.
Th hidden layer transfer function; defaults to gevcdn.logistic.
fixed vector indicating GEV parameters to be held constant; elements chosen from c("location", "scale", "shape")
init.range range for random weights on [min(init.range), max(init.range)]
scale.min minimum allowable value for the GEV scale parameter.
beta.p shape1 parameter for shifted beta distribution prior for GEV shape parameter
beta.q shape2 parameter for shifted beta distribution prior for GEV shape parameter
sd.norm sd parameter for normal distribution prior for the magnitude of input-hidden layer weights; equivalent to weight penalty regularization.
n.trials number of repeated trials used to avoid shallow local minima during optimization
method optimization method for optim function; must be chosen from c("BFGS", "Nelder-Mead").
boot.method bootstrap method; must be chosen from c("residual", "parametric").
init.from.prev logical value indicating whether or not optimization runs should be initialized from the final weights of the previous run.
max.fails maximum number of repeated exceptions allowed during optimization
gevcdn.bootstrap

probs vector of cumulative probabilities for which quantile confidence intervals are desired.

... additional arguments passed to the control list of optim function

Details

Note: the boot package provides a more comprehensive suit of methods for estimating bootstrap-based confidence intervals.

Value

a list consisting of
weights.bootstrap a list of length n.bootstrap consisting of weights following the format returned by gevcdn.fit

cparms.bootstrap a list of length n.bootstrap consisting of GEV parameters following the format returned by gevcdn.evaluate

location.bootstrap a matrix of GEV location parameters with number of rows equal to that of x and number of columns equal to n.bootstrap

scale.bootstrap a matrix of GEV scale parameters with number of rows equal to that of x and number of columns equal to n.bootstrap

shape.bootstrap a matrix of GEV shape parameters with number of rows equal to that of x and number of columns equal to n.bootstrap

quantiles.bootstrap a list of length n.bootstrap with each consisting of a matrix with number of rows equal to that of x and columns corresponding to probs

References


See Also

gevcdn.cost, gevcdn.evaluate, gevcdn.fit, gevcdn.bag, dgev, optim

Examples

## Generate synthetic data

```
x <- as.matrix(seq(0.1, 1, length = 50))

loc <- x^2
scl <- x/2
```
#### gevcdn.cost

Cost function for GEV CDN model fitting

**Description**

The generalized maximum likelihood (GML) cost function used for GEV CDN model fitting (Martin and Stedinger, 2000). Calculates the negative of the logarithm of the GML, which includes a shifted beta distribution prior for the GEV shape parameter. A normal distribution prior can also be set for the magnitude of the input-hidden layer weights, thus leading to weight penalty regularization.

**Usage**

```
gevcdn.cost(weights, x, y, n.hidden, Th, fixed, scale.min, beta.p, beta.q, sd.norm)
```
Arguments

weights
x
covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of variables.

y
column matrix of target values with number of rows equal to the number of samples.

n.hidden
number of hidden nodes in the GEV CDN model.

Th
hidden layer transfer function; defaults to gevcdn.logistic.

fixed
vector indicating GEV parameters to be held constant; elements chosen from c("location", "scale", "shape")

scale.min
minimum allowable value for the GEV scale parameter.

beta.p
shape1 parameter for shifted beta distribution prior for GEV shape parameter.

beta.q
shape2 parameter for shifted beta distribution prior for GEV shape parameter.

sd.norm
sd parameter for normal distribution prior for the magnitude of input-hidden layer weights; equivalent to weight penalty regularization.

References


See Also

gevcdn.fit, gevcdn.bag, dgev, optim

description

evaluate parameters from trained GEV CDN model

Usage

gevcdn.evaluate(x, weights)

Arguments

x
covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of variables.

weights
list containing GEV CDN input-hidden and hidden-output layer weight matrices from gevcdn.fit or gevcdn.bag.
Value

A matrix with number of rows equal to that of \( x \) and columns corresponding to the GEV location, scale, and shape parameters.

References


See Also

g.evcdn.fit, gevcdn.bag, d.gev

Examples

# Generate synthetic data
x <- as.matrix(1:50)
y <- as.matrix(rgev(length(x), location = 0, scale = 1, shape = 0.2))

## Fit stationary model
weights <- gevcdn.fit(x = x, y = y, Th = gevcdn.identity, fixed = c("location", "scale", "shape"))

## Evaluate GEV parameters
parms <- gevcdn.evaluate(x, weights)
cat("GEV parameters", parms[1], "\n")

description

Fit a GEV CDN model via nonlinear optimization of the generalized maximum likelihood cost function.

Usage

g.evcdn.fit(x, y, iter.max = 1000, n.hidden = 2,
Th = gevcdn.logistic, fixed = NULL,
init.range = c(-0.25, 0.25),
scale.min = .Machine$double.eps, beta.p = 3.3,
beta.q = 2, sd.norm = Inf, n.trials = 5,
method = c("BFGS", "Nelder-Mead"), max.fails = 100, ...)
Arguments

\( x \)  

covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of variables.

\( y \)  

column matrix of target values with number of rows equal to the number of samples.

\( \text{iter.max} \)  

maximum number of iterations of optimization algorithm.

\( \text{n.hidden} \)  

number of hidden nodes in the GEV CDN model.

\( \text{th} \)  

hidden layer transfer function; defaults to \texttt{gevcdn.logistic}.

\( \text{fixed} \)  

vector indicating GEV parameters to be held constant; elements chosen from \texttt{c("location", "scale", "shape")}

\( \text{init.range} \)  

range for random weights on \([\min(\text{init.range}), \max(\text{init.range})]\)

\( \text{scale.min} \)  

minimum allowable value for the GEV scale parameter.

\( \text{beta.p} \)  

shape1 parameter for shifted beta distribution prior for GEV shape parameter.

\( \text{beta.q} \)  

shape2 parameter for shifted beta distribution prior for GEV shape parameter.

\( \text{sd.norm} \)  

sd parameter for normal distribution prior for the magnitude of input-hidden layer weights; equivalent to weight penalty regularization.

\( \text{n.trials} \)  

number of repeated trials used to avoid shallow local minima during optimization.

\( \text{method} \)  

optimization method for \texttt{optim} function; must be chosen from \texttt{c("BFGS", "Nelder-Mead")}.

\( \text{max.fails} \)  

maximum number of repeated exceptions allowed during optimization.

\( ... \)  

additional arguments passed to the control list of \texttt{optim} function

Details

Fit a nonstationary GEV CDN model (Cannon, 2010) by minimizing a cost function based on the generalized maximum likelihood of Martins and Stedinger (2000). The hidden layer transfer function \( \text{th} \) should be set to \texttt{gevcdn.logistic} for a nonlinear model and to \texttt{gevcdn.identity} for a linear model. In the nonlinear case, the number of hidden nodes \( \text{n.hidden} \) controls the overall complexity of the model. GEV parameters can be held constant (i.e., stationary) via the \texttt{fixed} argument. The form of the shifted beta distribution prior for the GEV shape parameter is controlled by the \texttt{beta.p} and \texttt{beta.q} arguments. By default, these are set to values used in Cannon (2010). Other alternatives include values recommended by Martins and Stedinger (2000) (\( \text{beta.p} = 9 \) and \( \text{beta.q} = 6 \)) or values following from the normal distribution reported by Papalexiou and Koutsoyiannis (2013) (\( \text{beta.p} = 71.1 \) and \( \text{beta.q} = 44.7 \)). Weight penalty regularization for the magnitude of the input-hidden layer weights can be applied by setting \texttt{sd.norm} to a value less than \texttt{Inf}.

Values of the Akaike information criterion (AIC), 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 \( \text{n.hidden} \) set too high, as the GEV distribution may converge on one or more samples. This can usually be diagnosed by inspecting the scale parameter for near zero values. In this case, one can apply a weight penalty (via \texttt{sd.norm}), although this rules out the use of AIC/AICc/BIC for model selection as the effective number of model parameters will no longer equal the number of
weights in the GEV CDN model. Alternatively, a lower threshold (as a proportion of the range of y) for the scale parameter can be set via `scale.min`. Finally, bootstrap aggregation is available via `gevcdn_bag` as a second method for fitting GEV CDN models.

Note: values of x and y need not be standardized or rescaled by the user. All variables are automatically scaled to range between 0 and 1 prior to fitting and parameters are automatically rescaled by `gevcdn.evaluate`.

Value

A list consisting of

- `w1`: input-hidden layer weights
- `w2`: hidden-output layer weights

Attributes indicating the minimum/maximum values of x and y; the values of `Th, fixed, scale.min`; the negative of the logarithm of the generalized maximum likelihood `gml`, the negative of the logarithm of the likelihood `nll`, the value of the penalty term `penalty`, the Bayesian information criterion `bic`, the Akaike information criterion with (AICc) and without (AIC) small sample size correction; and the number of model parameters k are also returned.

References


See Also

- `gevcdn.cost`, `gevcdn.evaluate`, `gevcdn_bag`, `gevcdn.bootstrap`, `dgev`, `optim`

Examples

```r
## Generate synthetic data, quantiles
x <- as.matrix(seq(0.1, 1, length = 50))
loc <- x^2
scl <- x/2
shp <- seq(-0.1, 0.3, length = length(x))
set.seed(100)
y <- as.matrix(rgev(length(x), location = loc, scale = scl, shape = shp))
q <- sapply(c(0.1, 0.5, 0.9), qgev, location = loc, scale = scl, shape = shp)

## Define a hierarchy of models of increasing complexity
```
models <- list()
# Stationary model
models[[1]] <- list(Th = gevcdn.identity,
    fixed = c("location", "scale", "shape"))
# Linear model
models[[2]] <- list(Th = gevcdn.identity)
# Nonlinear, 1 hidden node
models[[3]] <- list(n.hidden = 1, Th = gevcdn.logistic)
# Nonlinear, 2 hidden nodes
models[[4]] <- list(n.hidden = 2, Th = gevcdn.logistic)

## Fit models
weights.models <- list()
for(i in seq_along(models)){
    weights.models[[i]] <- gevcdn.fit(x = x, y = y, n.trials = 1,
        n.hidden = models[[i]]$n.hidden,
        Th = models[[i]]$Th,
        fixed = models[[i]]$fixed)
}

## Select model with minimum AICC
models.AICC <- sapply(weights.models, attr, which = "AICC")
weights.best <- weights.models[[which.min(models.AICC)]]
parms.best <- gevcdn.evaluate(x, weights.best)

## 10th, 50th, and 90th percentiles
q.best <- sapply(c(0.1, 0.5, 0.9), qgev,
    location = parms.best[, "location"],
    scale = parms.best[, "scale"],
    shape = parms.best[, "shape"])

## Plot data and quantiles
matplot(x, cbind(y, q, q.best), type = c("b", rep("1", 6)),
    lty = c(1, rep(c(1, 2, 1), 2)),
    lwd = c(1, rep(c(3, 2, 3), 2)),
    col = c("red", rep("orange", 3), rep("blue", 3)),
    pch = 19, xlab = "x", ylab = "y", main = "gevcdn.fit")

---

**Description**

`gevcdn.identity` computes a trivial identity function. Used as the hidden layer transfer function for linear GEV CDN models.
Usage

gevcdn.identity(x)

Arguments

x a numeric vector

See Also

gevcdn.identity

gevcdn.initialize

Initialize GEV CDN weight vector

Description

Random initialization of the weight vector used during fitting of the GEV CDN model.

Usage

gevcdn.initialize(x, n.hidden, init.range)

Arguments

x covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of variables.

n.hidden number of hidden nodes in the GEV CDN model.

init.range range for random weights on [min(init.range), max(init.range)]

See Also

gevcdn.reshape

gevcdn.logistic

Logistic sigmoid function

Description

gevcdn.logistic computes the logistic sigmoid (S-shaped) function. Used as the hidden layer transfer function for nonlinear GEV CDN models.

Usage

gevcdn.logistic(x)
gevcdn.reshape

Arguments

x a numeric vector

See Also

gevcdn.identity

devcdn.reshape  Reshape a GEV CDN weight vector

Description

Reshapes a weight vector used during fitting of the GEV CDN model into input-hidden and hidden-output layer weight matrices.

Usage

devcdn.reshape(x, weights, n.hidden)

Arguments

x covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of variables.

weights weight vector of length returned by gevcdn.initialize.

n.hidden number of hidden nodes in the GEV CDN model.

See Also

gevcdn.initialize
Index

+Topic package
  GEVcdn-package, 2

boot, 7

d gev, 4, 7, 9, 10, 12

GEVcdn (GEVcdn-package), 2
GEVcdn-package, 2
gevcdn.bag, 2, 3, 7, 9, 10, 12
gevcdn.bootstrap, 2, 4, 5, 12
gevcdn.cost, 4, 7, 8, 12
gevcdn.evaluate, 2, 4, 7, 9, 12
gevcdn.fit, 2, 4, 7, 9, 10, 10
gevcdn.identity, 11, 13, 15
gevcdn.initialize, 9, 14, 15
gevcdn.logistic, 3, 6, 9, 11, 14, 14
gevcdn.reshape, 14, 15

optim, 3, 4, 6, 7, 9, 11, 12

VGAM, 2