Package 'esaddle'

October 13, 2022

Type Package

Title Extended Empirical Saddlepoint Density Approximations

Version 0.0.7

Date 2021-04-25

Author Matteo Fasiolo and Simon N. Wood

Maintainer Matteo Fasiolo <matteo.fasiolo@gmail.com>

Description Tools for fitting the Extended Empirical Saddlepoint (EES) density of Fasiolo et al. (2018) <doi:10.1214/18-EJS1433>.

License GPL (>= 2)

URL https://github.com/mfasiolo/esaddle

Imports compiler, stats, graphics, parallel, plyr, doParallel, mvnfast

Suggests knitr, markdown, testthat

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

RoxygenNote 7.0.2

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-04-26 14:50:02 UTC

R topics documented:

demvn *Evaluate the density of a multivariate Gaussian fit*

Description

Given a sample X, it gives a pointwise evaluation of the multivariate normal (MVN) density fit at position y.

Usage

```
demvn(y, X, log = FALSE, verbose = TRUE, alpha = 2, beta = 1.25)
```

Arguments

y	points at which the MVN is evaluated. It can be either a d-dimensional vector or an n by d matrix, each row indicating a different position.
X	an n by d matrix containing the data.
log	if TRUE the log-density is returned.
verbose	currently not used.
alpha	tuning parameter of robCov, see ?robCov for details.
beta	tuning parameter of robCov, see ?robCov for details.

Details

The covariance matrix is estimated robustly, using the robCov function.

Value

A vector where the i-th entry is the density corresponding to the i-th row of y.

Author(s)

Matteo Fasiolo <matteo.fasiolo@gmail.com> and Simon N. Wood.

Examples

```
library(esaddle)
X <- matrix(rnorm(2 * 1e3), 1e3, 2) # Sample used to fit a multivariate Gaussian
demvn(rnorm(2), X, log = TRUE)     # Evaluate the fitted log-density at a random location
```

dsaddle *Evaluating the Extended Empirical Saddlepoint (EES) density*

Description

Gives a pointwise evaluation of the EES density (and optionally of its gradient) at one or more locations.

Usage

```
dsaddle(
  y,
  X,
  decay,
  deriv = FALSE,
  log = FALSE,
  normalize = FALSE,
  control = list(),
  multicore = !is.null(cluster),
  ncores = detectCores() - 1,
  cluster = NULL
)
```

Arguments

y	points at which the EES is evaluated (d dimensional vector) or an n by d matrix, each row indicating a different position.
X	n by d matrix containing the data.
decay	rate at which the EES falls back on a normal density approximation, fitted to X. It must be a positive number, and it is inversely proportional to the complexity of the fit. Setting it to Inf leads to a Gaussian fit.
deriv	If TRUE also the gradient of the log-saddlepoint density is returned.
log	If TRUE the log of the saddlepoint density is returned.
normalize	If TRUE the normalizing constant of the EES density will be computed. FALSE by default.
control	A list of control parameters with entries: • method the method used to calculate the normalizing constant. Either "LAP" (laplace approximation) or "IS" (importance sampling). • nNorm if control$method == "IS", this is the number of importance samples used. • tol the tolerance used to assess the convergence of the solution to the saddlepoint equation. The default is 1e-6. • maxit maximal number of iterations used to solve the saddlepoint equation. The default is 100;

- `ml` Relevant only if `control$method=="IS"`. n random variables are generated from a Gaussian importance density with covariance matrix `ml*cov(X)`. By default the inflation factor is `ml=2`.

`multicore` if TRUE the empirical saddlepoint density at each row of y will be evaluated in parallel.

`ncores` number of cores to be used.

`cluster` an object of class `c("SOCKcluster", "cluster")`. This allowes the user to pass her own cluster, which will be used if `multicore == TRUE`. The user has to remember to stop the cluster.

Value

A list with entries:

- `llk` the value of the EES log-density at each location y;
- `mix` for each location y, the fraction of saddlepoint used: 1 means that only ESS is used and 0 means that only a Gaussian fit is used;
- `iter` for each location y, the number of iteration needed to solve the saddlepoint equation;
- `lambda` an n by d matrix, where the i-th row is the solution of the saddlepoint equation corresponding to the i-th row of y;
- `grad` the gradient of the log-density at y (optional);
- `logNorm` the estimated log normalizing constant (optional);

Author(s)

Matteo Fasiolo <matteo.fasiolo@gmail.com> and Simon N. Wood.

References

Fasiolo, M., Wood, S. N., Hartig, F. and Bravington, M. V. (2016). An Extended Empirical Saddlepoint Approximation for Intractable Likelihoods. ArXiv http://arxiv.org/abs/1601.01849.

Examples

```
library(esaddle)

### Simple univariate example
set.seed(4141)
x <- rgamma(1000, 2, 1)

# Evaluating EES at several point
xSeq <- seq(-2, 8, length.out = 200)
tmp <- dsaddle(y = xSeq, X = x, decay = 0.05, log = TRUE)  # Un-normalized EES
tmp2 <- dsaddle(y = xSeq, X = x, decay = 0.05,         # EES normalized by importance sampling
              normalize = TRUE, control = list("method" = "IS", nNorm = 500), log = TRUE)

# Plotting true density, EES and normal approximation
plot(xSeq, exp(tmp$llk), type = 'l', ylab = "Density", xlab = "x")
```

```
lines(xSeq, dgamma(xSeq, 2, 1), col = 3)
lines(xSeq, dnorm(xSeq, mean(x), sd(x)), col = 2)
lines(xSeq, exp(tmp2$llk), col = 4)
suppressWarnings( rug(x) )
legend("topright", c("EES un-norm", "EES normalized", "Truth", "Gaussian"),
        col = c(1, 4, 3, 2), lty = 1)
```

---

ecgf *Cumulant generating function estimation*

Description

Calculates the empirical cumulant generating function (CGF) and its derivatives given a sample of n d-dimentional vectors.

Usage

```
ecgf(lambda, X, mix, grad = 0)
```

Arguments

`lambda`	point at which the empirical CGF is evaluated (d-dimensional vector).
`X`	an n by d matrix containing the data.
`mix`	fraction of empirical and normal CGF to use. If `mix==1` only the empirical CGF is used, if `mix==0` only the normal CGF is used.
`grad`	if `grad==0` only the value of the CGF at `lambda` is returned, if `grad==1` also its first derivative wrt `lambda` and `if grad==2` also the second derivarive wrt `lambda`.

Details

For details on the CGF estimator being used here, see Fasiolo et al. (2016).

Value

A list with entries:

- `K` the value of the empirical CGF at `lambda`;
- `dK` the value of the gradient empirical CGF wrt lambda at `lambda`;
- `d2K` the value of the hessian of the empirical CGF wrt lambda at `lambda`.

Author(s)

Matteo Fasiolo <matteo.fasiolo@gmail.com> and Simon N. Wood.

References

Fasiolo, M., Wood, S. N., Hartig, F. and Bravington, M. V. (2016). An Extended Empirical Saddlepoint Approximation for Intractable Likelihoods. ArXiv http://arxiv.org/abs/1601.01849.

Examples

```
X <- matrix(rnorm(2 * 1e3), 1e3, 2)
K <- ecgf(lambda = c(0, 0), X = X, mix = 0.5, grad = 2)
K$K # CGF
K$dK # CGF' (gradient)
K$d2K # CGF'' (Hessian)
```

`findMode` *Finding the mode of the empirical saddlepoint density*

Description

Given a sample from a d-dimensional distribution, the routine finds the mode of the corresponding Extended Empirical Saddlepoint (EES) density.

Usage

```
findMode(
  X,
  decay,
  init = NULL,
  method = "BFGS",
  hess = FALSE,
  sadControl = list(),
  ...
)
```

Arguments

`X`	an n by d matrix containing the data.
`decay`	rate at which the SPA falls back on a normal density. Should be a positive number. See Fasiolo et al. (2016) for details.
`init`	d-dimensional vector containing the starting point for the optimization. By default it is equal to `colMeans(X)`.
`method`	optimization method used by `stats::optim()`, see ?optim for details. By default it is "BFGS".
`hess`	if TRUE also an estimate of the Hessian at the mode will be returned.
`sadControl`	list corresponding to the `control` argument in the `dsaddle` function.
`...`	Extra arguments to be passed to the optimization routine `stats::optim`.

Value

A list where mode is the location of mode of the empirical saddlepoint density, logDens is the log-density at the mode and hess (present only if argument hess==TRUE) is the approximate Hessian at convergence. The other entries are the same as for stats::optim.

Author(s)

Matteo Fasiolo <matteo.fasiolo@gmail.com>.

References

Fasiolo, M., Wood, S. N., Hartig, F. and Bravington, M. V. (2016). An Extended Empirical Saddlepoint Approximation for Intractable Likelihoods. ArXiv http://arxiv.org/abs/1601.01849.

Examples

```
# library(esaddle)
set.seed(4141)
x <- rgamma(1000, 2, 1)

# Fixing tuning parameter of EES
decay <-  0.05

# Evaluating EES at several point
xSeq <- seq(-2, 8, length.out = 200)
tmp <- dsaddle(y = xSeq, X = x, decay = decay, log = TRUE)  # Un-normalized EES

# Plotting true density, EES and normal approximation
plot(xSeq, exp(tmp$llk), type = 'l', ylab = "Density", xlab = "x")
lines(xSeq, dgamma(xSeq, 2, 1), col = 3)
suppressWarnings( rug(x) )
legend("topright", c("EES", "Truth"), col = c(1, 3), lty = 1)

# Find mode and plot it
res <- findMode(x, init = mean(x), decay = decay)$mode
abline(v = res, lty = 2, lwd = 1.5)
```

---

robCov *Robust covariance matrix estimation*

Description

Obtains a robust estimate of the covariance matrix of a sample of multivariate data, using Campbell's (1980) method as described on p231-235 of Krzanowski (1988).

Usage

```
robCov(sY, alpha = 2, beta = 1.25)
```

Arguments

sY	A matrix, where each column is a replicate observation on a multivariate r.v.
alpha	tuning parameter, see details.
beta	tuning parameter, see details.

Details

Campbell (1980) suggests an estimator of the covariance matrix which downweights observations at more than some Mahalanobis distance `d.0` from the mean. `d.0` is `sqrt(nrow(sY))+alpha/sqrt(2)`. Weights are one for observations with Mahalanobis distance, `d`, less than `d.0`. Otherwise weights are `d.0*exp(-.5*(d-d.0)^2/beta^2)/d`. The defaults are as recommended by Campbell. This routine also uses pre-conditioning to ensure good scaling and stable numerical calculations. If some of the columns of `sY` has zero variance, these are removed.

Value

A list where:

- `COV` The estimated covariance matrix.
- `E` A square root of the inverse covariance matrix. i.e. the inverse cov matrix is `t(E)%*%E`;
- `half.ldet.V` Half the log of the determinant of the covariance matrix;
- `mY` The estimated mean;
- `sd` The estimated standard deviations of each variable.
- `weights` This is `w1/sum(w1)*ncol(sY)`, where `w1` are the weights of Campbell (1980).
- `lowVar` The indexes of the columns of `sY` whose variance is zero (if any). These variable were removed and excluded from the covariance matrix.

Author(s)

Simon N. Wood, maintained by Matteo Fasiolo <matteo.fasiolo@gmail.com>.

References

Krzanowski, W.J. (1988) Principles of Multivariate Analysis. Oxford. Campbell, N.A. (1980) Robust procedures in multivariate analysis I: robust covariance estimation. JRSSC 29, 231-237.

Examples

```
p <- 5;n <- 100
Y <- matrix(runif(p*n),p,n)
robCov(Y)
```

`rsaddle` *Simulate random variables from the Extended Empirical Saddlepoint density (ESS)*

Description

Simulate random variables from the Extended Empirical Saddlepoint density (ESS), using importance sampling and then resampling according to the importance weights.

Usage

```
rsaddle(
  n,
  X,
  decay,
  ml = 2,
  multicore = !is.null(cluster),
  cluster = NULL,
  ncores = detectCores() - 1,
  ...
)
```

Arguments

`n`	number of simulated vectors.
`X`	an m by d matrix containing the data.
`decay`	rate at which the ESS falls back on a normal density. Should be a positive number. See Fasiolo et al. (2016) for details.
`ml`	n random variables are generated from a Gaussian importance density with covariance matrix `ml*cov(X)`. By default the inflation factor is `ml=2`.
`multicore`	if TRUE the ESS densities corresponding the samples will be evaluated in parallel.
`cluster`	an object of class `c("SOCKcluster", "cluster")`. This allowes the user to pass her own cluster, which will be used if `multicore == TRUE`. The user has to remember to stop the cluster.
`ncores`	number of cores to be used.
`...`	additional arguments to be passed to `dsaddle`.

Details

Notice that, while importance sampling is used, the output is a matrix of unweighted samples, obtained by resampling with probabilities proportional to the importance weights.

Value

An n by d matrix containing the simulated vectors.

Author(s)

Matteo Fasiolo <matteo.fasiolo@gmail.com>.

References

Fasiolo, M., Wood, S. N., Hartig, F. and Bravington, M. V. (2016). An Extended Empirical Saddlepoint Approximation for Intractable Likelihoods. ArXiv http://arxiv.org/abs/1601.01849.

Examples

```
# Simulate bivariate data, where each marginal distribution is Exp(2)
X <- matrix(rexp(2 * 1e3), 1e3, 2)

# Simulate bivariate data from a saddlepoint fitted to X
Z <- rsaddle(1000, X, decay = 0.5)

# Look at first marginal distribution
hist( Z[ , 1] )
```

`selectDecay` *Tuning the Extended Empirical Saddlepoint (EES) density by cross-validation*

Description

Performs k-fold cross-validation to choose the EES's tuning parameter, which determines the mixture between a consistent and a Gaussian estimator of the Cumulant Generating Function (CGF).

Usage

```
selectDecay(
  decay,
  simulator,
  K,
  nrep = 1,
  normalize = FALSE,
  draw = TRUE,
  multicore = !is.null(cluster),
  cluster = NULL,
  ncores = detectCores() - 1,
  control = list(),
  ...
)
```

Arguments

decay	Numeric vector containing the possible values of the tuning parameter.
simulator	Function with prototype function(...) that will be called nrep times to simulate d-dimensional random variables. Each time simulator is called, it will return a n by d matrix.
K	the number of folds to be used in cross-validation.
nrep	Number of times the whole cross-validation procedure will be repeated, by calling simulator to generate random variable and computing the cross-validation score for every element of the decay vector.
normalize	if TRUE the normalizing constant of EES is normalized at each value of decay. FALSE by default.
draw	if TRUE the results of cross-validation will be plotted. TRUE by default.
multicore	if TRUE each fold will run on a different core.
cluster	an object of class c("SOCKcluster", "cluster"). This allowes the user to pass her own cluster, which will be used if multicore == TRUE. The user has to remember to stop the cluster.
ncores	number of cores to be used.
control	a list of control parameters, with entries: • method The method used to calculate the normalizing constant. Either "LAP" (laplace approximation) or "IS" (importance sampling). • tol The tolerance used to assess the convergence of the solution to the saddlepoint equation. The default is 1e-6. • nNorm Number of simulations to be used in order to estimate the normalizing constant of the saddlepoint density. By default equal to 1e3. • ml if method=="IS" nNorm, random variables are generated from a Gaussian importance density with covariance matrix ml*cov(X). By default the inflation factor is ml=2.
...	extra arguments to be passed to simulator.

Value

A list with entries:

- negLogLik A matrix length{decay} by K*nrep where the i-th row represent the negative loglikelihood estimated for the i-th value of decay, while each column represents a different fold and repetition.
- summary A matrix of summary results from the cross-validation procedure.
- normConst A matrix length{decay} by nrep where the i-th row contains the estimates of the normalizing constant.

The list is returned invisibly. If control$draw == TRUE the function will also plot the cross-validation curve.

Author(s)

Matteo Fasiolo <matteo.fasiolo@gmail.com>.

References

Fasiolo, M., Wood, S. N., Hartig, F. and Bravington, M. V. (2016). An Extended Empirical Saddlepoint Approximation for Intractable Likelihoods. ArXiv http://arxiv.org/abs/1601.01849.

Examples

```
library(esaddle)
# The data is far from normal: saddlepoint is needed and we expect
# cross validation to be minimized at low "decay"
set.seed(4124)
selectDecay(decay = c(0.001, 0.01, 0.05, 0.1, 0.5, 1),
            simulator = function(...) rgamma(500, 2, 1),
            K = 5)

# The data is far from normal: saddlepoint is not needed and we expect
#  the curve to be fairly flat for high "decay"
selectDecay(decay = c(0.001, 0.01, 0.05, 0.1, 0.5, 1),
            simulator = function(...) rnorm(500, 0, 1),
            K = 5)
```

Index