Package ‘varycoef’

September 17, 2022

Type Package

Title Modeling Spatially Varying Coefficients

Version 0.3.4

Description Implements a maximum likelihood estimation (MLE) method for estimation and prediction of Gaussian process-based spatially varying coefficient (SVC) models (Dambon et al. (2021a) <doi:10.1016/j.spasta.2020.100470>). Covariance tapering (Furrer et al. (2006) <doi:10.1198/106186006X132178>) can be applied such that the method scales to large data. Further, it implements a joint variable selection of the fixed and random effects (Dambon et al. (2021b) <doi:10.1080/13658816.2022.2097684>). The package and its capabilities are described in (Dambon et al. (2021c) <arXiv:2106.02364>).

License GPL-2

URL https://github.com/jakobdambon/varycoef

BugReports https://github.com/jakobdambon/varycoef/issues

Depends R (>= 3.5.0)

Imports glmnet, lhs, methods, mlr, mlrMBO, optimParallel (>= 0.8-1), ParamHelpers, pbapply, smoof, spam

Suggests DiceKriging, knitr, lattice, latticeExtra, parallel, rmarkdown, sp, spData, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

VignetteBuilder knitr

NeedsCompilation no

Author Jakob A. Dambon [aut, cre] (<https://orcid.org/0000-0001-5855-2017>), Fabio Sigrist [ctb] (<https://orcid.org/0000-0002-3994-2244>), Reinhard Furrer [ctb] (<https://orcid.org/0000-0002-6319-2332>)

Maintainer Jakob A. Dambon <jakob.dambon@math.uzh.ch>

Repository CRAN

Date/Publication 2022-09-17 19:20:02 UTC

R topics documented:

`check_cov_lower` *Check Lower Bound of Covariance Parameters*

Description

Ensures that the covariance parameters define a positive definite covariance matrix. It takes the vector $(\rho_1, \sigma^2_1, ..., \rho_q, \sigma^2_q, \tau^2)$ and checks if all $\rho_k > 0$, all $\sigma^2_k >= 0$, and $\tau^2 > 0$.

Usage

```
check_cov_lower(cv, q)
```

Arguments

`cv`	`(numeric(2*q+1))` Covariance vector of SVC model.
`q`	`(numeric(1))` Integer indicating the number of SVCs.

Value

`logical(1)` with `TRUE` if all conditions above are fulfilled.

Examples

```
# first one is true, all other are false
check_cov_lower(c(0.1, 0, 0.2,  1, 0.2), q = 2)
check_cov_lower(c(0  , 0, 0.2,  1, 0.2), q = 2)
check_cov_lower(c(0.1, 0, 0.2,  1, 0  ), q = 2)
check_cov_lower(c(0.1, 0, 0.2, -1, 0  ), q = 2)
```

`coef.SVC_mle` *Extact Mean Effects*

Description

Method to extract the mean effects from an `SVC_mle` or `SVC_selection` object.

Usage

```
## S3 method for class 'SVC_mle'
coef(object, ...)

## S3 method for class 'SVC_selection'
coef(object, ...)
```

Arguments

`object`	`SVC_mle` or `SVC_selection` object
`...`	further arguments

Value

named vector with mean effects, i.e. μ from `SVC_mle`

Author(s)

Jakob Dambon

`cov_par` *Extact Covariance Parameters*

Description

Function to extract the covariance parameters from an `SVC_mle` or `SVC_selection`object.

Usage

```
cov_par(...)

## S3 method for class 'SVC_mle'
cov_par(object, ...)

## S3 method for class 'SVC_selection'
cov_par(object, ...)
```

Arguments

`...`	further arguments
`object`	`SVC_mle` or `SVC_selection` object

Value

vector with covariance parameters with the following attributes:

- `"GRF"`, charachter, describing the covariance function used for the GP, see `SVC_mle_control`.
- `"tapering"`, either `NULL` if no tapering is applied of the taper range.

Author(s)

Jakob Dambon

`fitted.SVC_mle` *Extact Model Fitted Values*

Description

Method to extract the fitted values from an `SVC_mle` object. This is only possible if `save.fitted` was set to `TRUE` in the control of the function call

Usage

```
## S3 method for class 'SVC_mle'
fitted(object, ...)
```

Arguments

`object`	`SVC_mle` object
`...`	further arguments

Value

Data frame, fitted values to given data, i.e., the SVC as well as the response and their locations

Author(s)

Jakob Dambon

`GLS_chol` *GLS Estimate using Cholesky Factor*

Description

Computes the GLS estimate using the formula:

$$\mu_{GLS} = (X^\top \Sigma^{-1} X)^{-1} X^\top \Sigma^{-1} y.$$

The computation is done depending on the input class of the Cholesky factor `R`. It relies on the classical `solve` or on using `forwardsolve` and `backsolve` functions of package `spam`, see `solve`. This is much faster than computing the inverse of Σ, especially since we have to compute the Cholesky decomposition of Σ either way.

Usage

```
GLS_chol(R, X, y)

## S3 method for class 'spam.chol.NgPeyton'
GLS_chol(R, X, y)

## S3 method for class 'matrix'
GLS_chol(R, X, y)
```

Arguments

`R`	(`spam.chol.NgPeyton` or `matrix(n, n)`) Cholesky factor of the covariance matrix Σ. If covariance tapering and sparse matrices are used, then the input is of class `spam.chol.NgPeyton`. Otherwise, `R` is the output of a standard `chol`, i.e., a simple `matrix`
`X`	(`matrix(n, p)`) Data / design matrix.
`y`	(`numeric(n)`) Response vector

Value

A `numeric(p)` vector, i.e., the mean effects.

Author(s)

Jakob Dambon

Examples

```
# generate data
n <- 10
X <- cbind(1, 20+1:n)
y <- rnorm(n)
A <- matrix(runif(n^2)*2-1, ncol=n)
Sigma <- t(A) %*% A
# two possibilities
## using standard Cholesky decomposition
R_mat <- chol(Sigma); str(R_mat)
mu_mat <- GLS_chol(R_mat, X, y)
## using spam
R_spam <- chol(spam::as.spam(Sigma)); str(R_spam)
mu_spam <- GLS_chol(R_spam, X, y)
# should be identical to the following
mu <- solve(crossprod(X, solve(Sigma, X))) %*%
      crossprod(X, solve(Sigma, y))
## check
abs(mu - mu_mat)
abs(mu - mu_spam)
```

`house` *Lucas County House Price Data*

Description

A dataset containing the prices and other attributes of 25,357 houses in Lucas County, Ohio. The selling dates span years 1993 to 1998. Data taken from `house` (`spData` package) and slightly modified to a `data.frame`.

Usage

```
house
```

Format

A data frame with 25357 rows and 25 variables:

price (`integer`) selling price, in US dollars

yrbuilt (`integer`) year the house was built

stories (factor) levels are "one", "bilevel", "multilvl", "one+half", "two", "two+half", "three"

TLA (integer) total living area, in square feet.

wall (factor) levels are "stucdrvt", "ccbtile", "metlvnyl", "brick", "stone", "wood", "partbrk"

beds, baths, halfbaths (integer) number of corresponding rooms / facilities.

frontage, depth dimensions of the lot. Unit is feet.

garage (factor) levels are "no garage", "basement", "attached", "detached", "carport"

garagesqft (integer) garage area, in square feet. If garage == "no garage", then garagesqft == 0.

rooms (integer) number of rooms

lotsize (integer) area of lot, in square feet

sdate (Date) selling date, in format yyyy-mm-dd

avalue (int) appraised value

s1993, s1994, s1995, s1996, s1997, s1998 (int) dummies for selling year.

syear (factor) levels are selling years "1993", "1994", "1995", "1996", "1997", "1998"

long, lat (numeric) location of houses. Longitude and Latitude are given in CRS(+init=epsg:2834), the Ohio North State Plane. Units are meters.

Source

http://www.spatial-econometrics.com/html/jplv6.zip

IC.SVC_mle *Conditional Akaike's and Bayesian Information Criteria*

Description

Methods to calculate information criteria for SVC_mle objects. Currently, two are supported: the conditional Akaike's Information Criteria $cAIC = -2 * log - likelihood + 2 * (edof + df)$ and the Bayesian Information Criteria $BIC = -2 * log - likelihood + log(n) * npar$. Note that the Akaike's Information Criteria is of the corrected form, that is: $edof$ is the effective degrees of freedom which is derived as the trace of the hat matrices and df is the degree of freedoms with respect to mean parameters.

Usage

```
## S3 method for class 'SVC_mle'
BIC(object, ...)

## S3 method for class 'SVC_mle'
AIC(object, conditional = "BW", ...)
```

Arguments

`object`	`SVC_mle` object
`...`	further arguments
`conditional`	string. If `conditional = "BW"`, the conditional AIC is calculated.

Value

numeric, value of information criteria

Author(s)

Jakob Dambon

`init_bounds_optim` *Setting of Optimization Bounds and Initial Values*

Description

Sets bounds and initial values for `optim` by extracting potentially given values from `SVC_mle_control` and checking them, or calculating them from given data. See Details.

Usage

```
init_bounds_optim(control, p, q, id_obj, med_dist, y_var, OLS_mu)
```

Arguments

`control`	(`SVC_mle_control` output, i.e. `list`)
`p`	(`numeric(1)`) Number of fixed effects
`q`	(`numeric(1)`) Number of SVCs
`id_obj`	(`numeric(2*q+1+q)`) Index vector to identify the arguments of objective function.
`med_dist`	(`numeric(1)`) Median distance between observations
`y_var`	(`numeric(1)`) Variance of response `y`
`OLS_mu`	(`numeric(p)`) Coefficient estimates of ordinary least squares (OLS).

Details

If values are not provided, then they are set in the following way. Let d be the median distance `med_dist`, let s_y^2 be the variance of the response `y_var`, and let b_j be the OLS coefficients of the linear model. The computed values are given in the table below.

Parameter	Lower bound	Initial Value	Upper Bound
Range	$d/1000$	$d/4$	$10d$
Variance	0	$s_y^2/(q+1)$	$10s_y^2$
Nugget	10^{-6}	$s_y^2/(q+1)$	$10s_y^2$
Mean j	`-Inf`	b_j	`Inf`

Value

A `list` with three entries: `lower`, `init`, and `upper`.

Author(s)

Jakob Dambon

`logLik.SVC_mle` *Extact the Likelihood*

Description

Method to extract the computed (penalized) log (profile) Likelihood from an `SVC_mle` object.

Usage

```
## S3 method for class 'SVC_mle'
logLik(object, ...)
```

Arguments

`object`	`SVC_mle` object
`...`	further arguments

Value

an object of class `logLik` with attributes

- `"penalized"`, logical, if the likelihood (`FALSE`) or some penalized likelihood (`TRUE`) was optimized.
- `"profileLik"`, logical, if the optimization was done using the profile likelihood (`TRUE`) or not.
- `"nobs"`, integer of number of observations
- `"df"`, integer of how many parameters were estimated. **Note**: This includes only the covariance parameters if the profile likelihood was used.

Author(s)

Jakob Dambon

nlocs *Extract Number of Unique Locations*

Description

Function to extract the number of unique locations in the data set used in an MLE of the SVC_mle object.

Usage

```
nlocs(object)
```

Arguments

object	SVC_mle object

Value

integer with the number of unique locations

Author(s)

Jakob Dambon

nobs.SVC_mle *Extract Number of Observations*

Description

Method to extract the number of observations used in MLE for an SVC_mle object.

Usage

```
## S3 method for class 'SVC_mle'
nobs(object, ...)
```

Arguments

object	SVC_mle object
...	further arguments

Value

an integer of number of observations

Author(s)

Jakob Dambon

plot.SVC_mle *Plotting Residuals of* SVC_mle *model*

Description

Method to plot the residuals from an SVC_mle object. For this, save.fitted has to be TRUE in SVC_mle_control.

Usage

```
## S3 method for class 'SVC_mle'
plot(x, which = 1:2, ...)
```

Arguments

x	(SVC_mle)
which	(numeric) A numeric vector and subset of 1:2 indicating which of the 2 plots should be plotted.
...	further arguments

Value

a maximum 2 plots

- Tukey-Anscombe plot, i.e. residuals vs. fitted
- QQ-plot

Author(s)

Jakob Dambon

See Also

legend SVC_mle

Examples

```
#' ## ---- toy example ----
## sample data
# setting seed for reproducibility
set.seed(123)
m <- 7
# number of observations
n <- m*m
# number of SVC
p <- 3
# sample data
```

```
y <- rnorm(n)
X <- matrix(rnorm(n*p), ncol = p)
# locations on a regular m-by-m-grid
locs <- expand.grid(seq(0, 1, length.out = m),
                    seq(0, 1, length.out = m))

## preparing for maximum likelihood estimation (MLE)
# controls specific to MLE
control <- SVC_mle_control(
  # initial values of optimization
  init = rep(0.1, 2*p+1),
  # using profile likelihood
  profileLik = TRUE
)

# controls specific to optimization procedure, see help(optim)
opt.control <- list(
  # number of iterations (set to one for demonstration sake)
  maxit = 1,
  # tracing information
  trace = 6
)

## starting MLE
fit <- SVC_mle(y = y, X = X, locs = locs,
               control = control,
               optim.control = opt.control)

## output: convergence code equal to 1, since maxit was only 1
summary(fit)

## plot residuals
# only QQ-plot
plot(fit, which = 2)

# two plots next to each other
oldpar <- par(mfrow = c(1, 2))
plot(fit)
par(oldpar)
```

`predict.SVC_mle` *Prediction of SVCs (and response variable)*

Description

Prediction of SVCs (and response variable)

Usage

```
## S3 method for class 'SVC_mle'
predict(
  object,
  newlocs = NULL,
  newX = NULL,
  newW = NULL,
  newdata = NULL,
  compute.y.var = FALSE,
  ...
)
```

Arguments

object	(SVC_mle) Model obtained from SVC_mle function call.
newlocs	(NULL or matrix(n.new, 2)) If NULL, then function uses observed locations of model to estimate SVCs. Otherwise, these are the new locations the SVCs are predicted for.
newX	(NULL or matrix(n.new, q)) If provided (together with newW), the function also returns the predicted response variable.
newW	(NULL or matrix(n.new, p)) If provided (together with newX), the function also returns the predicted response variable.
newdata	(NULL or data.frame(n.new, p)) This argument can be used, when the SVC_mle function has been called with an formula, see examples.
compute.y.var	(logical(1)) If TRUE and the response is being estimated, the predictive variance of each estimate will be computed.
...	further arguments

Value

The function returns a data frame of n.new rows and with columns

- SVC_1, ..., SVC_p: the predicted SVC at locations newlocs.
- y.pred, if newX and newW are provided
- y.var, if newX and newW are provided and compute.y.var is set to TRUE.
- loc_x, loc_y, the locations of the predictions

Author(s)

Jakob Dambon

References

Dambon, J. A., Sigrist, F., Furrer, R. (2021) *Maximum likelihood estimation of spatially varying coefficient models for large data with an application to real estate price prediction*, Spatial Statistics doi: 10.1016/j.spasta.2020.100470

See Also

SVC_mle

Examples

```
## ---- toy example ----
## We use the sampled, i.e., one dimensional SVCs
str(SVCdata)
# sub-sample data to have feasible run time for example
set.seed(123)
id <- sample(length(SVCdata$locs), 50)

## SVC_mle call with matrix arguments
fit_mat <- with(SVCdata, SVC_mle(
  y[id], X[id, ], locs[id],
  control = SVC_mle_control(profileLik = TRUE, cov.name = "mat32")))

## SVC_mle call with formula
df <- with(SVCdata, data.frame(y = y[id], X = X[id, -1]))
fit_form <- SVC_mle(
  y ~ X, data = df, locs = SVCdata$locs[id],
  control = SVC_mle_control(profileLik = TRUE, cov.name = "mat32")
)

## prediction

# predicting SVCs
predict(fit_mat, newlocs = 1:2)
predict(fit_form, newlocs = 1:2)

# predicting SVCs and response providing new covariates
predict(
  fit_mat,
  newX = matrix(c(1, 1, 3, 4), ncol = 2),
  newW = matrix(c(1, 1, 3, 4), ncol = 2),
  newlocs = 1:2
)
predict(fit_form, newdata = data.frame(X = 3:4), newlocs = 1:2)
```

print.summary.SVC_mle *Printing Method for* summary.SVC_mle

Description

Printing Method for summary.SVC_mle

Usage

```
## S3 method for class 'summary.SVC_mle'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	summary.SVC_mle
digits	the number of significant digits to use when printing.
...	further arguments

Value

The printed output of the summary in the console.

See Also

summary.SVC_mle SVC_mle

---

print.SVC_mle *Print Method for* SVC_mle

Description

Method to print an SVC_mle object.

Usage

```
## S3 method for class 'SVC_mle'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	SVC_mle object
digits	(numeric) Number of digits to be plotted.
...	further arguments

Author(s)

Jakob Dambon

`residuals.SVC_mle` *Extact Model Residuals*

Description

Method to extract the residuals from an `SVC_mle` object. This is only possible if `save.fitted` was set to `TRUE`.

Usage

```
## S3 method for class 'SVC_mle'
residuals(object, ...)
```

Arguments

`object`	`SVC_mle` object
`...`	further arguments

Value

(`numeric(n)`) Residuals of model

Author(s)

Jakob Dambon

`sample_SVCdata` *Sample Function for GP-based SVC Model for Given Locations*

Description

Samples SVC data at given locations. The SVCs parameters and the covariance function have to be provided. The sampled model matrix can be provided or it is sampled. The SVCs are sampled according to their given parametrization and at respective observation locations. The error vector is sampled from a nugget effect. Finally, the response vector is computed. Please note that the function is not optimized for sampling large data sets.

Usage

```
sample_SVCdata(
  df.pars,
  nugget.sd,
  locs,
  cov.name = c("exp", "sph", "mat32", "mat52", "wend1", "wend2"),
  X = NULL
)
```

Arguments

- `df.pars` — (`data.frame(p, 3)`)
  Contains the mean and covariance parameters of SVCs. The three columns must have the names `"mean"`, `"var"`, and `"scale"`.
- `nugget.sd` — (`numeric(1)`)
  Standard deviation of the nugget / error term.
- `locs` — (`numeric(n)` or `matrix(n, d)`)
  The numeric vector or matrix contains the observation locations and therefore defines the number of observations to be `n`. For a vector, we assume locations on the real line, i.e., $d = 1$.
- `cov.name` — (`character(1)`)
  Character defining the covariance function, c.f. `SVC_mle_control`.
- `X` — (`NULL` or `matrix(n, p)`)
  If `NULL`, the covariates are sampled, where the first column contains only ones to model an intercept and further columns are sampled from a standard normal. If it is provided as a `matrix`, then the dimensions must match the number of locations in `locs` (`n`) and the number of SVCs defined by the number of rows in `df.pars` (`p`).

Details

The parameters of the model can be chosen such that we obtain data from a not full model, i.e., not all covariates are associated with a fixed and a random effect. Using `var = 0` for instance yields a constant beta coefficient for respective covariate. Note that in that case the `scale` value is neglected.

Value

`list`
Returns a list with the response `y`, model matrix `X`, a matrix `beta` containing the sampled SVC at given locations, a vector `eps` containing the error, and a matrix `locs` containing the original locations. The `true_pars` contains the data frame of covariance parameters that were used to sample the GP-based SVCs. The nugget variance has been added to the original argument of the function with its respective variance, but `NA` for `"mean"` and `"scale"`.

Examples

```
set.seed(123)
# SVC parameters
(df.pars <- data.frame(
   var = c(2, 1),
   scale = c(3, 1),
   mean = c(1, 2)))
# nugget standard deviation
tau <- 0.5

# sample locations
s <- sort(runif(500, min = 0, max = 10))
SVCdata <- sample_SVCdata(
```

```
  df.pars = df.pars, nugget.sd = tau, locs = s, cov.name = "mat32"
)
```

---

summary.SVC_mle *Summary Method for* SVC_mle

Description

Method to construct a summary.SVC_mle object out of a SVC_mle object.

Usage

```
## S3 method for class 'SVC_mle'
summary(object, ...)
```

Arguments

object	SVC_mle object
...	further arguments

Value

object of class summary.SVC_mle with summarized values of the MLE.

Author(s)

Jakob Dambon

See Also

SVC_mle

SVCdata *Sampled SVC Data*

Description

A list object that contains sampled data of 500 observations. The data has been sampled using the RandomFields package (Schlather et al., 2015). It is given in the list object SVCdata which contains the following.

Usage

```
SVCdata
```

Format

A `list` with the following entries:

y (`numeric`) Response

X (`numeric`) Covariates; first columns contains ones to model an intercept, the second column contains standard-normal sampled data.

beta (`numeric`) The sampled Gaussian processes, which are usually unobserved. It uses a Matern covariance function and the true parameters are given in the entry 'true_pars'.

eps (`numeric`) Error (or Nugget effect), i.e., drawn from a zero-mean normal distribution with 0.5 standard deviation.

locs (`numeric`) Locations sampled from a uniform distribution on the interval 0 to 10.

true_pars (`data.frame`) True parameters of the GP-based SVC model with Gaussian process mean, variance, and range. Additionally, the smoothness (nu) is given.

References

Schlather, M., Malinowski, A., Menck, P. J., Oesting, M., Strokorb, K. (2015) *Analysis, simulation and prediction of multivariate random fields with package RandomFields*, Journal of Statistical Software, doi: 10.18637/jss.v063.i08

`SVC_mle` *MLE of SVC model*

Description

Conducts a maximum likelihood estimation (MLE) for a Gaussian process-based spatially varying coefficient model as described in Dambon et al. (2021) doi: 10.1016/j.spasta.2020.100470.

Usage

```
SVC_mle(...)

## Default S3 method:
SVC_mle(y, X, locs, W = NULL, control = NULL, optim.control = list(), ...)

## S3 method for class 'formula'
SVC_mle(
  formula,
  data,
  RE_formula = NULL,
  locs,
  control = NULL,
  optim.control = list(),
  ...
)
```

Arguments

`...`	further arguments
`y`	`(numeric(n))` Response vector.
`X`	`(matrix(n, p))` Design matrix. Intercept has to be added manually.
`locs`	`(matrix(n, d))` Locations in a d-dimensional space. May contain multiple observations at single location.
`W`	`(NULL` or `matrix(n, q))` If `NULL`, the same matrix as provided in `X` is used. This fits a full SVC model, i.e., each covariate effect is modeled with a mean and an SVC. In this case we have $p = q$. If optional matrix `W` is provided, SVCs are only modeled for covariates within matrix `W`.
`control`	`(list)` Control paramaters given by `SVC_mle_control`.
`optim.control`	`(list)` Control arguments for optimization function, see Details in `optim`.
`formula`	Formula describing the fixed effects in SVC model. The response, i.e. LHS of the formula, is not allowed to have functions such as `sqrt()` or `log()`.
`data`	data frame containing the observations
`RE_formula`	Formula describing the random effects in SVC model. Only RHS is considered. If `NULL`, the same RHS of argument `formula` for fixed effects is used.

Details

The GP-based SVC model is defined with some abuse of notation as:

$$y(s) = X\mu + W\eta(s) + \epsilon(s)$$

where:

- y is the response (vector of length n)
- X is the data matrix for the fixed effects covariates. The dimensions are n times p. This leads to p fixed effects.
- μ is the vector containing the fixed effects
- W is the data matrix for the SVCs modeled by GPs. The dimensions are n times q. This lead to q SVCs in the model.
- η are the SVCs represented by a GP.
- ϵ is the nugget effect

The MLE is an numeric optimization that runs `optim` or (if parallelized) `optimParallel`.

You can call the function in two ways. Either, you define the model matrices yourself and provide them using the arguments `X` and `W`. As usual, the individual columns correspond to the fixed and random effects, i.e., the Gaussian processes, respectively. The second way is to call the function with formulas, like you would in lm. From the data.frame provided in argument data, the respective model matrices as described above are implicitly built. Using simple arguments formula and RE_formula with data column names, we can decide which covariate is modeled with a fixed or random effect (SVC).

Note that similar to model matrix call from above, if the RE_formula is not provided, we use the one as in argument formula. Further, note that the intercept is implicitly constructed in the model matrix if not prohibited.

Value

Object of class SVC_mle if control$extract_fun = FALSE, meaning that a MLE has been conducted. Otherwise, if control$extract_fun = TRUE, the function returns a list with two entries:

- obj_fun: the objective function used in the optimization
- args: the arguments to evaluate the objective function.

For further details, see description of SVC_mle_control.

Author(s)

Jakob Dambon

References

Dambon, J. A., Sigrist, F., Furrer, R. (2021) *Maximum likelihood estimation of spatially varying coefficient models for large data with an application to real estate price prediction*, Spatial Statistics doi: 10.1016/j.spasta.2020.100470

See Also

predict.SVC_mle

Examples

```
## ---- toy example ----
## We use the sampled, i.e., one dimensional SVCs
str(SVCdata)
# sub-sample data to have feasible run time for example
set.seed(123)
id <- sample(length(SVCdata$locs), 50)

## SVC_mle call with matrix arguments
fit <- with(SVCdata, SVC_mle(
  y[id], X[id, ], locs[id],
  control = SVC_mle_control(profileLik = TRUE, cov.name = "mat32")))

## SVC_mle call with formula
df <- with(SVCdata, data.frame(y = y[id], X = X[id, -1]))
fit <- SVC_mle(
  y ~ X, data = df, locs = SVCdata$locs[id],
```

```
  control = SVC_mle_control(profileLik = TRUE, cov.name = "mat32")
)
class(fit)

summary(fit)


## ---- real data example ----
require(sp)
## get data set
data("meuse", package = "sp")

# construct data matrix and response, scale locations
y <- log(meuse$cadmium)
X <- model.matrix(~1+dist+lime+elev, data = meuse)
locs <- as.matrix(meuse[, 1:2])/1000


## starting MLE
# the next call takes a couple of seconds
fit <- SVC_mle(
  y = y, X = X, locs = locs,
  # has 4 fixed effects, but only 3 random effects (SVC)
  # elev is missing in SVC
  W = X[, 1:3],
  control = SVC_mle_control(
    # inital values for 3 SVC
    # 7 = (3 * 2 covariance parameters + nugget)
    init = c(rep(c(0.4, 0.2), 3), 0.2),
    profileLik = TRUE
  )
)

## summary and residual output
summary(fit)
plot(fit)

## predict
# new locations
newlocs <- expand.grid(
  x = seq(min(locs[, 1]), max(locs[, 1]), length.out = 30),
  y = seq(min(locs[, 2]), max(locs[, 2]), length.out = 30))
# predict SVC for new locations
SVC <- predict(fit, newlocs = as.matrix(newlocs))
# visualization
sp.SVC <- SVC
coordinates(sp.SVC) <- ~loc_1+loc_2
spplot(sp.SVC, colorkey = TRUE)
```

SVC_mle_control *Set Parameters for* SVC_mle

Description

Function to set up control parameters for SVC_mle. In the following, we assume the GP-based SVC model to have q GPs which model the SVCs and p fixed effects.

Usage

```
SVC_mle_control(...)

## Default S3 method:
SVC_mle_control(
  cov.name = c("exp", "sph", "mat32", "mat52", "wend1", "wend2"),
  tapering = NULL,
  parallel = NULL,
  init = NULL,
  lower = NULL,
  upper = NULL,
  save.fitted = TRUE,
  profileLik = FALSE,
  mean.est = c("GLS", "OLS"),
  pc.prior = NULL,
  extract_fun = FALSE,
  hessian = TRUE,
  dist = list(method = "euclidean"),
  parscale = TRUE,
  ...
)

## S3 method for class 'SVC_mle'
SVC_mle_control(object, ...)
```

Arguments

`...` Further Arguments yet to be implemented

`cov.name` (`character(1)`)
Name of the covariance function of the GPs. Currently, the following are implemented: `"exp"` for the exponential, `"sph"` for spherical, `"mat32"` and `"mat52"` for Matern class covariance functions with smoothness 3/2 or 5/2, as well as `"wend1"` and `"wend2"` for Wendland class covariance functions with kappa 1 or 2.

`tapering` (`NULL` or `numeric(1)`)
If `NULL`, no tapering is applied. If a scalar is given, covariance tapering with this taper range is applied, for all Gaussian processes modeling the SVC. Only defined for Matern class covariance functions, i.e., set `cov.name` either to `"exp"`, `"mat32"`, or `"mat52"`.

`parallel` (`NULL` or `list`)
If `NULL`, no parallelization is applied. If cluster has been established, define arguments for parallelization with a list, see documentation of `optimParallel`. See Examples.

init (NULL or numeric(2q+1+p*as.numeric(profileLik)))
Initial values for optimization procedure. If NULL is given, an initial vector is calculated (see Details). Otherwise, the vector is assumed to consist of q-times (alternating) range and variance, the nugget variance and if profileLik = TRUE p mean effects.

lower (NULL or numeric(2q+1+p*as.numeric(profileLik)))
Lower bound for init in optim. Default NULL calculates the lower bounds (see Details).

upper (NULL or numeric(2q+1+p*as.numeric(profileLik)))
Upper bound for init in optim. Default NULL calculates the upper bounds (see Details).

save.fitted (logical(1))
If TRUE, calculates the fitted values and residuals after MLE and stores them. This is necessary to call residuals and fitted methods afterwards.

profileLik (logical(1))
If TRUE, MLE is done over profile Likelihood of covariance parameters.

mean.est (character(1))
If profileLik = TRUE, the means have to be estimated seperately for each step. "GLS" uses the generalized least square estimate while "OLS" uses the ordinary least squares estimate.

pc.prior (NULL or numeric(4))
If numeric vector is given, penalized complexity priors are applied. The order is $\rho_0, \alpha_\rho, \sigma_0, \alpha_\sigma$ to give some prior believes for the range and the standard deviation of GPs, such that $P(\rho < \rho_0) = \alpha_\rho, P(\sigma > \sigma_0) = \alpha_\sigma$. This regulates the optimization process. Currently, only supported for GPs with of Matérn class covariance functions. Based on the idea by Fulgstad et al. (2018) doi: 10.1080/01621459.2017.1415907.

extract_fun (logical(1))
If TRUE, the function call of SVC_mle stops before the MLE and gives back the objective function of the MLE as well as all used arguments. If FALSE, regular MLE is conducted.

hessian (logical(1))
If TRUE, Hessian matrix is computed, see optim. This required to give the standard errors for covariance parameters and to do a Wald test on the variances, see summary.SVC_mle.

dist (list)
List containing the arguments of dist or nearest.dist. This controls the method of how the distances and therefore dependency structures are calculated. The default gives Euclidean distances in a d-dimensional space. Further editable arguments are p, miles, R, see respective help files of dist or nearest.dist.

parscale (logical(1))
Triggers parameter scaling within the optimization in optim. If TRUE, the optional parameter scaling in optim.control in function SVC_mle is overwritten by the initial value used in the numeric optimization. The initial value is either computed from the data or provided by the user, see init argument above or Details below. Note that we check whether the initial values are unequal to

zero. If they are zero, the corresponding scaling factor is 0.001. If `FALSE`, the `parscale` argument in `optim.control` is let unchanged.

`object` (SVC_mle)
The function then extracts the control settings from the function call used to compute in the given `SVC_mle` object.

Details

If not provided, the initial values as well as the lower and upper bounds are calculated given the provided data. In particular, we require the median distance between observations, the variance of the response and, the ordinary least square (OLS) estimates, see `init_bounds_optim`.

The argument `extract_fun` is useful, when one wants to modify the objective function. Further, when trying to parallelize the optimization, it is useful to check whether a single evaluation of the objective function takes longer than 0.05 seconds to evaluate, cf. Gerber and Furrer (2019) doi: 10.32614/RJ2019030. Platform specific issues can be sorted out by the user by setting up their own optimization.

Value

A list with which `SVC_mle` can be controlled.

Author(s)

Jakob Dambon

See Also

`SVC_mle`

Examples

```
control <- SVC_mle_control(init = rep(0.3, 10))
# or
control <- SVC_mle_control()
control$init <- rep(0.3, 10)


# Code for setting up parallel computing
require(parallel)
# exchange number of nodes (1) for detectCores()-1 or appropriate number
cl <- makeCluster(1, setup_strategy = "sequential")
clusterEvalQ(
  cl = cl,
  {
    library(spam)
    library(varycoef)
  })
# use this list for parallel argument in SVC_mle_control
parallel.control <- list(cl = cl, forward = TRUE, loginfo = TRUE)
# SVC_mle goes here ...
# DO NOT FORGET TO STOP THE CLUSTER!
```

```
stopCluster(cl); rm(cl)
```

SVC_selection *SVC Model Selection*

Description

This function implements the variable selection for Gaussian process-based SVC models using a penalized maximum likelihood estimation (PMLE, Dambon et al., 2021, <arXiv:2101.01932>). It jointly selects the fixed and random effects of GP-based SVC models.

Usage

```
SVC_selection(obj.fun, mle.par, control = NULL, ...)
```

Arguments

`obj.fun`	(`SVC_obj_fun`) Function of class `SVC_obj_fun`. This is the output of `SVC_mle` with the `SVC_mle_control` parameter `extract_fun` set to `TRUE`. This objective function comprises of the whole SVC model on which the selection should be applied.
`mle.par`	(`numeric(2*q+1)`) Numeric vector with estimated covariance parameters of unpenalized MLE.
`control`	(`list` or `NULL`) List of control parameters for variable selection. Output of `SVC_selection_control`. If `NULL` is given, the default values of `SVC_selection_control` are used.
`...`	Further arguments.

Value

Returns an object of class `SVC_selection`. It contains parameter estimates under PMLE and the optimization as well as choice of the shrinkage parameters.

Author(s)

Jakob Dambon

References

Dambon, J. A., Sigrist, F., Furrer, R. (2021). *Joint Variable Selection of both Fixed and Random Effects for Gaussian Process-based Spatially Varying Coefficient Models*, ArXiv Preprint https://arxiv.org/abs/2101.01932

SVC_selection_control *SVC Selection Parameters*

Description

Function to set up control parameters for SVC_selection. The underlying Gaussian Process-based SVC model is defined in SVC_mle. SVC_selection then jointly selects fixed and random effects of the GP-based SVC model using a penalized maximum likelihood estimation (PMLE). In this function, one can set the parameters for the PMLE and its optimization procedures (Dambon et al., 2022).

Usage

```
SVC_selection_control(
  IC.type = c("BIC", "cAIC_BW", "cAIC_VB"),
  method = c("grid", "MBO"),
  r.lambda = c(1e-10, 10),
  n.lambda = 10L,
  n.init = 10L,
  n.iter = 10L,
  CD.conv = list(N = 20L, delta = 1e-06, logLik = TRUE),
  hessian = FALSE,
  adaptive = FALSE,
  parallel = NULL,
  optim.args = list()
)
```

Arguments

IC.type	(character(1)) Select Information Criterion.
method	(character(1)) Select optimization method for lambdas, i.e., shrinkage parameters. Either model-based optimization (MBO, Bischl et al., 2017 <arXiv:1703.03373>) or over grid.
r.lambda	(numeric(2)) Range of lambdas, i.e., shrinkage parameters.
n.lambda	(numeric(1)) If grid method is selected, number of lambdas per side of grid.
n.init	(numeric(1)) If MBO method is selected, number of initial values for surrogate model.
n.iter	(numeric(1)) If MBO method is selected, number of iteration steps of surrogate models.
CD.conv	(list(3)) List containing the convergence conditions, i.e., first entry is the maximum number of iterations, second value is the relative change necessary to stop iteration,

third is logical to toggle if relative change in log likelihood (TRUE) or rather the parameters themselves (FALSE) is the criteria for convergence.

hessian
: (logical(1))
If TRUE, Hessian will be computed for final model.

adaptive
: (logical(1))
If TRUE, adaptive LASSO is executed, i.e., the shrinkage parameter is defined as $\lambda_j := \lambda/|\theta_j|$.

parallel
: (list)
List with arguments for parallelization, see documentation of optimParallel.

optim.args
: (list)
List of further arguments of optimParallel, such as the lower bounds.

Value

A list of control parameters for SVC selection.

Author(s)

Jakob Dambon

References

Bischl, B., Richter, J., Bossek, J., Horn, D., Thomas, J., Lang, M. (2017). *mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions*, ArXiv preprint https://arxiv.org/abs/1703.03373

Dambon, J. A., Sigrist, F., Furrer, R. (2022). *Joint Variable Selection of both Fixed and Random Effects for Gaussian Process-based Spatially Varying Coefficient Models*, International Journal of Geographical Information Science doi: 10.1080/13658816.2022.2097684

Examples

```
# Initializing parameters and switching logLik to FALSE
selection_control <- SVC_selection_control(
  CD.conv = list(N = 20L, delta = 1e-06, logLik = FALSE)
)
# or
selection_control <- SVC_selection_control()
selection_control$CD.conv$logLik <- FALSE
```

varycoef *varycoef: Modeling Spatially Varying Coefficients*

Description

This package offers functions to estimate and predict Gaussian process-based spatially varying coefficient (SVC) models. Briefly described, one generalizes a linear regression equation such that the coefficients are no longer constant, but have the possibility to vary spatially. This is enabled by modeling the coefficients using Gaussian processes with (currently) either an exponential or spherical covariance function. The advantages of such SVC models are that they are usually quite easy to interpret, yet they offer a very high level of flexibility.

Estimation and Prediction

The ensemble of the function `SVC_mle` and the method `predict` estimates the defined SVC model and gives predictions of the SVC as well as the response for some pre-defined locations. This concept should be rather familiar as it is the same for the classical regression (`lm`) or local polynomial regression (`loess`), to name a couple. As the name suggests, we are using a *maximum likelihood estimation* (MLE) approach in order to estimate the model. The predictor is obtained by the empirical best linear unbiased predictor. to give location-specific predictions. A detailed tutorial with examples is given in a vignette; call `vignette("example", package = "varycoef")`. We also refer to the original article Dambon et al. (2021) which lays the methodological foundation of this package.

With the before mentioned `SVC_mle` function one gets an object of class `SVC_mle`. And like the method `predict` for predictions, there are several more methods in order to diagnose the model, see `methods(class = "SVC_mle")`.

Variable Selection

As of version 0.3.0 of `varycoef`, a joint variable selection of both fixed and random effect of the Gaussian process-based SVC model is implemented. It uses a *penalized maximum likelihood estimation* (PMLE) which is implemented via a gradient descent. The estimation of the shrinkage parameter is available using a *model-based optimization* (MBO). Here, we use the framework by Bischl et al. (2017). The methodological foundation of the PMLE is described in Dambon et al. (2022).

Author(s)

Jakob Dambon

References

Bischl, B., Richter, J., Bossek, J., Horn, D., Thomas, J., Lang, M. (2017). *mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions*, ArXiv preprint https://arxiv.org/abs/1703.03373

Dambon, J. A., Sigrist, F., Furrer, R. (2021). *Maximum likelihood estimation of spatially varying coefficient models for large data with an application to real estate price prediction*, Spatial Statistics 41 100470 doi: 10.1016/j.spasta.2020.100470

Dambon, J. A., Sigrist, F., Furrer, R. (2022). *Joint Variable Selection of both Fixed and Random Effects for Gaussian Process-based Spatially Varying Coefficient Models*, International Journal of Geographical Information Science doi: 10.1080/13658816.2022.2097684

Examples

```
vignette("manual", package = "varycoef")
methods(class = "SVC_mle")
```

Index