Package ‘varycoef’

October 12, 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>
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 \((\rho_1, \sigma_1^2, \ldots, \rho_q, \sigma_q^2, \tau^2)\) and checks if all \(\rho_k > 0\), all \(\sigma_k^2 \geq 0\), and \(\tau^2 > 0\).

Usage

\[
\text{check_cov_lower}(\text{cv}, q)
\]
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

```r
# 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)
```

Description

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

Usage

```r
## 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. $\mu$ from `SVC_mle`

Author(s)

Jakob Dambon
cov_par | Extract Covariance Parameters

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

Usage
```r
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", character, 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 | Extract 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
```r
## S3 method for class 'SVC_mle'
fitted(object, ...)
```
GLS\_chol

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 $\Sigma$, especially since we have to compute the Cholesky decomposition of $\Sigma$ either way.

Usage

GLS\_chol(R, X, y)

```r
## S3 method for class 'spam.chol.NgPeyton'
GLS\_chol(R, X, y)
```

```r
## 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 $\Sigma$. 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", "metlvynl", "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


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


---

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 \( cAIC = -2 \ast \log - \text{likelihood} + 2 \ast (edof + df) \) and the Bayesian Information Criteria \( BIC = -2 \ast \log - \text{likelihood} + \log(n) \ast npar \). 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.
### logLik.SVC_mle

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

#### Usage
```r
## 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
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.
...

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)
### S3 method for class 'SVC_mle'

```r
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


See Also

*SVC_mle*

Examples

```r
## ---- 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

```r
## 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

```r
## 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  

**Extract 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**

```r
## 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**

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

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

```r
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(
```
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**


---

**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 parameters 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.
- \( \mu \) 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 leads to \( q \) SVCs in the model.
- \( \eta \) are the SVCs represented by a GP.
- \( \epsilon \) 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 \texttt{lm}. From the \texttt{data.frame} provided in argument \texttt{data}, the respective model matrices as described above are implicitly built. Using simple arguments \texttt{formula} and \texttt{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 \texttt{RE_formula} is not provided, we use the one as in argument \texttt{formula}. Further, note that the intercept is implicitly constructed in the model matrix if not prohibited.

\textbf{Value}

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

- \texttt{obj_fun}: the objective function used in the optimization
- \texttt{args}: the arguments to evaluate the objective function.

For further details, see description of \texttt{SVC_mle_control}.

\textbf{Author(s)}

Jakob Dambon

\textbf{References}


\textbf{See Also}

\texttt{predict.SVC_mle}

\textbf{Examples}

```r
## ---- 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(
    # initial 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**

**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 \(p, \text{miles}, \text{R}\), 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.

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 \texttt{init.bounds.optim}.

The argument \texttt{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 \texttt{SVC.mle} can be controlled.

Author(s)

Jakob Dambon

See Also

\texttt{SVC.mle}

Examples

```r
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
c1 <- makeCluster(1, setup_strategy = "sequential")
clusterEvalQ(
  cl = c1,
  {
    library(spam)
    library(varycoef)
  })
# use this list for parallel argument in SVC.mle_control
parallel.control <- list(c1 = c1, forward = TRUE, loginfo = TRUE)
# SVC.mle goes here ...
# DO NOT FORGET TO STOP THE CLUSTER!
```
SVC_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

SVC_selection_control

**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**

```r
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


### Examples

```r
# 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
```
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


**Examples**

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

**datasets**
- house, 6
- SVCdata, 18

AIC.SVC_mle (IC.SVC_mle), 7
BIC.SVC_mle (IC.SVC_mle), 7
check_cov_lower, 2
c chol, 5
c coef.SVC_mle, 3
c coef.SVC_selection (coef.SVC_mle), 3
c cov_par, 4
dist, 24
fitted, 24
fitted.SVC_mle, 4

GLS_chol, 5

house, 6, 6
IC.SVC_mle, 7
init_bounds_optim, 8, 25

legend, 11
lm, 21, 29
loess, 29
logLik.SVC_mle, 9

nearest.dist, 24
nlocs, 10
nobs.SVC_mle, 10

optim, 8, 20, 24
optimParallel, 20, 23, 28

plot.SVC_mle, 11
predict.SVC_mle, 12, 21
print.summary.SVC_mle, 14
print.SVC_mle, 15

residuals, 24
residuals.SVC_mle, 16

sample_SVCdata, 16
solve, 5
summary.SVC_mle, 15, 18, 24
SVC_mle, 3–5, 7–11, 13–16, 18, 19, 23–27, 29
SVC_mle_control, 4, 8, 11, 17, 20, 21, 22, 26
SVC_selection, 3, 4, 26, 27
SVC_selection_control, 26, 27
SVCdata, 18

varycoef, 29