Package 'SSM'

July 4, 2017

Type Package

Title Fit and Analyze Smooth Supersaturated Models

Version 1.0.1

Description Creates an S4 class ``SSM'' and defines functions for fitting smooth supersaturated models, a polynomial model with spline-like behaviour. Functions are defined for the computation of Sobol indices for sensitivity analysis and plotting the main effects using FANOVA methods. It also implements the estimation of the SSM metamodel error using a GP model with a variety of defined correlation functions.

License GPL-3

LazyData true

RoxygenNote 5.0.1

Imports methods

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/peterrobertcurtis/SSM

BugReports http://github.com/peterrobertcurtis/SSM/issues

NeedsCompilation no

Author Peter Curtis [aut, cre]

Maintainer Peter Curtis <peterrobertcurtis@gmail.com>

Repository CRAN

Date/Publication 2017-07-04 13:00:05 UTC

R topics documented:

`fit.ssm` *Fit a smooth supersaturated model*

Description

`fit.ssm` fits a smooth supersaturated model to given data. By default the model smooths over $[-1, 1]^d$ and uses a basis of Legendre polynomials. Many model parameters such as basis size and smoothing criterion can be user-defined. Optionally, sensitivity indices and a metamodel error estimating Gaussian process can be computed.

Usage

```
fit.ssm(design, response, ssm, basis, basis_size, K, P, design_model_matrix,
  SA = FALSE, GP = FALSE, type = "exp", validation = FALSE,
  exclude = list(), distance_type = "distance")
```

Arguments

`design`	A matrix containing the design. Each design point is a row in the matrix. Accepts a vector for a design in one variable. If the default options for `P` and `K` are used then it is recommended that the design is transformed to lay within $[-1, 1]^d$. The function `transform11` is useful in this regard.
`response`	A vector containing the responses at design points. The length must correspond with the number of rows in `design`.
`ssm`	(optional) A pre-existing SSM class object. If this argument is supplied then `basis`, `basis_size`, `K`, and `P` will be carried over rather than re-computed. This is useful for simulation studies where the model structure remains the same and only the design and responses change.
`basis`	(optional) A matrix where each row is an exponent vector of a monomial. This is used in conjunction with `P` to construct the model basis. If not supplied, a hierarchical basis will be used.
`basis_size`	(optional) A number. Specifies the desired number of polynomials in the model basis. If not supplied, `basis_size` is set to $20 * d + n$.
`K`	(optional) A semi-positive definite matrix specifying the weighting criterion of basis terms. If not supplied, default behaviour is to use the Frobenius norm of the term Hessian integrated over $[-1, 1]^d$ with respect to a basis of Legendre polynomials.
`P`	(optional) A matrix defining the polynomials used to construct the model basis. Each column corresponds to one polynomial. If not supplied, a Legendre polynomial basis is used.

`design_model_matrix`
: (optional) Specify a design model matrix. If provided the new model will be fit to the basis and design implied by the design model matrix regardless of the values of `basis`, `P` and `design`.

`SA`
: (optional) Logical. If TRUE then Sobol indices, Total indices and Total interaction indices will be computed.

`GP`
: (optional) Logical. If TRUE then a GP metamodel error estimate will be computed.

`type`
: (optional) Character. One of "exp" or "matern32". Specifies the correlation function used to compute the GP covariance matrix. Irrelevant if `GP` is FALSE. For further details see `compute.covariance`.

`validation`
: (optional) Logical. If TRUE then the Leave-One-Out errors are computed for each design point and the standardised root mean square error calculated. The rmse is standardised against the variance of the ybar estimator. If `GP` is TRUE then these will be calculated regardless of the value of `validation`.

`exclude`
: (optional) A list of vectors of integers. These indicate terms in the listed variables should be omitted from the model. *e.g.* `exclude = list(1)` removes all terms dependent on the first variable only. `exclude = list(1, c(1, 2))` removes terms in the first variable only and interactions between the first and second variables only. To remove a variable and all of its higher order interactions, it is better to remove the appropriate column from the design otherwise the algorithm will generate a lot of basis vectors that will be excluded, wasting computation.

`distance_type`
: (optional) Character. Selects the distance function used for the GP metamodel error estimate correlation function. One of "distance", "line", "product", "area", "proddiff", or "smoothdiff". Uses "distance", the standard Euclidean distance between points by default. For further details see `new.distance`. Not needed if `GP` is FALSE.

Details

Returns an SSM object containing the fitted smooth supersaturated model. Minimal arguments required are `design` and `response`. This will result in a model using Legendre polynomials and smoothing over $[-1, 1]^d$. All other arguments will be assigned automatically if not specified.

If the model is unable to be fit due to numerical instability a warning will be given and the returned SSM object will have a zero vector of appropriate length in the `theta` slot. The `basis_size` parameter is often useful in this situation. Try reducing the basis_size until a model fit is successful. Ideally the basis size should be as large as possible without causing instability in the predictions (see the example below).

If `SA` is TRUE then sensitivty analysis will be performed on the model. Sobol indices for main effects, Total indices, and Total interaction indices for second order interactions will be computed and held in the slots `main_sobol`, `total_sobol` and `total_int` respectively. If the number of factors is < 11 then Sobol indices will be computed for all order interactions and stored in `int_sobol`. Default behaviour is to assume each input is uniformly distributed over $[-1, 1]$. If `P` has been used-defined then the polynomials defined by `P` are assumed to be an orthonormal system with respect to some measure on the reals. See `update.sensitivity` for more details.

If GP is TRUE (default behaviour is false due to computational cost) then a the metamodel error is estimated using a zero-mean Gaussian process with a constant trend. Scaling and length parameters are estimated using maximum likelihood methods using the Leave-One-Out model errors and stored in the sigma and r slots respectively. Model predictions using predict.SSM will then include credible intervals. The distance between points is defined by the distance_type argument and is set to the standard Euclidean distance by default. See new.distance for other options although they are experimental and subject to erratic behaviour. The default correlation function used is the square exponential although this can be changed to a Matern 3/2 function by setting the type argument to "matern32".

If validation is TRUE then the Leave-One_Out error at each design point will be computed and stored in the residuals slot, and the LOO RMSE computed and stored in the LOO_RMSE slot. Note that if GP is TRUE then these values will be computed regardless of the value of validation as they are required to fit the metamodel error estimate GP.

Value

An SSM object.

See Also

predict.SSM for model predictions for SSM, and plot.SSM for plotting main effects of SSM. transform11 is useful for transforming data to $[-1, 1]^d$.

Examples

```
# A simple one factor example
X <- seq(-1,1,0.5) # design
Y <- c(0,1,0,0.5,0) # response
s <- fit.ssm(X,Y)
s
plot(s)
predict(s,0.3)

# used defined basis sizes

# A model that is too large to fit
## Not run:
s <- fit.ssm(X, Y, basis_size=80)

## End(Not run)
# A large model that can be fit but is unstable
s <- fit.ssm(X, Y, basis_size=70)
plot(s)
# A model larger than default that is not unstable
s <- fit.ssm(X, Y, basis_size=40)
plot(s)

# with metamodel error estimate

s <- fit.ssm(X, Y, GP=TRUE)
plot(s)
```

```
predict(s,0.3)

# Sensitivity analysis and main effect plotting

# A design of 20 points over [-1, 1]^d
X <- matrix(runif(20, -1, 1), ncol = 2)
Y <- runif(10)
s <- fit.ssm(X, Y, SA = TRUE)
s
sensitivity.plot(s)
plot(s)
```

`likelihood.plot` *Plot the concentrated likelihood of an SSM.*

Description

Plot the concentrated likelihood used to estimate the parameters of the metamodel error estimating Gaussian process.

Usage

```
likelihood.plot(ssm, xrange = c(0, 1000), grid = 200)
```

Arguments

`ssm`	An SSM object.
`xrange`	(optional) The range of the x axis. Set to `c(0, 1000)` by default.
`grid`	(optional) A number. The number of points used to plot the curve.

Details

As a diagnostic it can be helpful to look at the concentrated likelihood function of the correlation function used to estimate the metamodel error. Flat likelihood functions make it difficult to pick a suitable `r` length parameter. Note that `r` and `sigma` can be set manually.

Examples

```
data(attitude)
X <- transform11(attitude[ 2:7])
Y <- attitude[ , 1]
s <- fit.ssm(X, Y, GP = TRUE)
likelihood.plot(s)
likelihood.plot(s, xrange = c(0, 20))
```

plot.SSM *Plot smooth supersaturated model main effects*

Description

plot.SSM is a plot method for SSM objects. It plots the main effects of the SSM only, that is the subset of basis terms that are dependent on a single variable only. For single variable data this is a plot of the complete model.

Usage

```
## S3 method for class 'SSM'
plot(x, ..., grid = 200, yrange = "full", GP = TRUE)
```

Arguments

x	An SSM object.
...	(optional) arguments to pass to the plot function.
grid	(optional) A number. This specifies the resolution of the plot, *i.e.* how many model evaluations are used to construct the curve.
yrange	(optional) Character. Only "full" will have an effect.
GP	(optional) Logical. For single variable data, the credible interval of the meta-model error estimator will be plotted if TRUE.

Details

For each variable, the effect is plotted over $[-1, 1]$ by default although passing an alternate range to the xlim argument will override this.

The yrange argument is designed to automatically compute the relevant plot range for each effect. By default a ylim value is passed to plot that covers the range of responses. "full" results in a ylim value that covers the range of predictions or, if appropriate, the range of the metamodel error credible interval.

For single variable data, setting GP to TRUE will plot a credible interval for the metamodel error estimating Gaussian process if this has been computed for the SSM object.

Examples

```
# A single variable example
X <- seq(-1, 1, 0.25)
Y <- sapply(X, "^", 3)
s <- fit.ssm(X, Y, GP = TRUE)
plot(s)

# A six variable example
data(attitude)
X <- transform11(attitude[ 2:7])
```

```
Y <- attitude[ , 1]
s <- fit.ssm(X, Y)
plot(s)
```

`predict.SSM` *Point prediction of smooth supersaturated models.*

Description

This method gives the prediction of an SSM object at a point. If the SSM has a metamodel error estimate then a $(1 - \alpha)$ credible interval is also output.

Usage

```
## S3 method for class 'SSM'
predict(object, x, alpha = 0.05, ...)
```

Arguments

`object`	An SSM object.
`x`	A d length vector identifying the prediction point.
`alpha`	(optional) A number in $[0, 1]$ for the $(1 - \alpha)$ metamodel error estimate credible interval. Set to `0.05` by default.
`...`	further arguments passed to or from other methods.

Value

Either a number if the SSM has no metamodel error estimating Gaussian process, or three numbers giving the model prediction (`$model`), and the lower and upper bounds of the credible interval (`$lower` and `$upper`) respectively.

Examples

```
data(attitude)
X <- transform11(attitude[ 2:7])
Y <- attitude[ , 1]
# with no metamodel error estimating GP.
s <- fit.ssm(X, Y)
predict(s, rep(1,6))

# with metamodel error estimating GP.
s <- fit.ssm(X, Y, GP = TRUE)
predict(s, rep(1,6))
```

`sensitivity.plot` *Plot the sensitivity indices of a smooth supersaturated model.*

Description

`sensitivity.plot` visualises the sensitivity indices of a given smooth supersaturated model using `barplot`. If the `SA` flag was not set to TRUE when `fit.ssm` was run to fit the model, then `update.sensitivity` should be used to compute the model variances. If not, this function will return an error message.

Usage

```
sensitivity.plot(ssm, type = "main_total", ...)
```

Arguments

`ssm`	An SSM object. Must have relevant sensitivity indices in the appropriate slots, either from setting `SA = TRUE` in `fit.ssm` or by using `update.sensitivity`.
`type`	(optional) Character. Determines the type of barplot. One of `"sobol"`, `"main_sobol"`, `"main_total"`, or `"total"`. Default behaviour is `"main_total"`.
`...`	arguments passed to the `barplot` function call.

Details

There are four classes of plot available:

- `"sobol"` Produces a barplot of all Sobol indices. If there are more than 10 factors then Sobol indices will not have been computed for interactions and only the Sobol indices for main effects will be plotted. Main effects are in red, interactions are in grey.
- `"main_sobol"` Produces a barplot of Sobol indices for main effects only.
- `"main_total"` Produces a barplot of Total indices for main effects only. This is the default behaviour.
- `"total"` Produces a barplot of Total indices for main effects and all second order interactions. Main effects are in red, interactions are in grey.

Note that variables and interactions are not labelled in the plots since there can be too many bars to label clearly.

Examples

```
# A 20 point design in four variables
X <- matrix(runif(80, -1, 1), ncol = 4)
Y <- runif(20)
s <- fit.ssm(X, Y, SA = TRUE)
sensitivity.plot(s)

# In the next plots, the grey bars indicate interactions.
```

```
sensitivity.plot(s, "sobol")
sensitivity.plot(s, "total")
# Identifying particular indices is best done using the information held in
# the SSM object.  The following orders s@total_int_factors so the
# interaction indicated by the top row is the most important.
ind <- order(s@total_int, decreasing = TRUE)
s@total_int_factors[ind, ]
```

`show,SSM-method` *Summarise SSM class object*

Description

Printing an SSM will summarise the parameters d, N, and n. If no model has been fit then this will be noted, otherwise The smoothness of the SSM will be shown. If sensitivity analysis has been performed, the Sobol indices and Total indices for main effects are displayed and if cross-validation has been performed the Standardised LOO RMSE is also shown.

Usage

```
## S4 method for signature 'SSM'
show(object)
```

Arguments

`object`	An SSM object

`SSM` *SSM: A package for fitting smooth supersaturated models (SSM).*

Description

The SSM package provides an S4 class to represent smooth supersaturated models, along with functions for fitting SSM to data, computing Sobol indices for the SSM and estimating metamodel error with a Gaussian process.

SSM functions

There are three important functions in the package.

`fit.ssm` returns an SSM object that fits an SSM to the data and, by default, computes the Sobol indices, and Total interaction indices of the model. Optionally, the metamodel error can be estimated using a Gaussian process. The fitted SSM smooths over $[-1, 1]^d$ and uses a hierarchical basis of 20*d+n Legendre polynomials by default but the function is highly customisable.

`predict.SSM` returns the model prediction at a point, including a credible interval if a metamodel error GP has been fit.

`plot.SSM` plots the main effects of the SSM.

SSM-class *An S4 class to represent a smooth supersaturated model*

Description

An S4 class to represent a smooth supersaturated model

Slots

`dimension` A number indicating the number of variables in the design.

`design` A matrix with rows indicating the design point and columns indicating the variable.

`design_size` A number indicating the number of design points.

`response` A `design_size` length vector of responses.

`theta` A vector containing the fitted model coefficients.

`basis` A matrix with each row being the exponent vector of a polynomial term.

`basis_size` A number indicating the number of basis terms used in the model. This may be different from `nrow(basis)` if terms are excluded.

`include` A vector containing the row numbers of the basis polynomials used in the model. This is used when interactions or variables are being excluded from the model.

`K` A semi-positive definite matrix that defines the smoothing criteria.

`P` A matrix that defines the polynomial basis in terms of a monomial basis.

`design_model_matrix` A matrix.

`variances` A vector of length `basis_size` containing the term variances.

`total_variance` A length one vector containing the total variance.

`main_sobol` A `dimension` length vector containing the Sobol index for each variable.

`main_ind` A logical matrix indicating whether each term is included in the main effect corresponding to the column.

`total_sobol` A `dimension` length vector containing the Total sensitivity index for each variable.

`total_ind` A logical matrix indicating whether each term is included in the Total sensitivity index corresponding to the column.

`int_sobol` A vector containing the Sobol index for interactions.

`int_factors` A list of length the same as `int_sobol` indicating which interaction corresponds with each entry in `int_sobol`.

`total_int` A vector containing the Total interaction indices of all second order interactions.

`total_int_factors` A matrix where each row indicates the variables associated with the corresponding interaction in `total_int`.

`distance` A matrix containing the distances used for computing the covariance matrix of the GP metamodel error estimate.

`distance_type` A character defining the distance type used for computing `distance`. Can be one of "distance", "line", "product", "area", "proddiff", or "smoothdiff".

`type` A character, either "exp", "matern32", that selects the correlation function used for the GP metamodel error estimate.

`covariance` A positive definite matrix. The covariance matrix of the GP metamodel error estimate prior to scaling by `sigma`.

`residuals` A `design_size` length vector containing the Leave-One-Out errors of the model at each design point.

`sigma` A number indicating the scaling factor for `covariance`.

`r` A number indicating the length factor for the correlation function.

`local_smoothness` A `design_size` length vector containing the model smoothness at each design point.

`LOO_RMSE` A number. The Leave-One-Out root mean square error.

`legendre` logical. Indicates whether the default Legendre polynomial basis is being used.

`fail` logical. Indicates whether the model fit was successful.

`transform11` *Transform a design to [-1, 1]^d*

Description

This function transforms a design (supplied as a matrix) into the space [-1, 1]^d. This has numerical and computational advantages when using smooth supersaturated models and is assumed by the default `fit.ssm` behaviour.

Usage

```
transform11(design)
```

Arguments

`design`	A matrix where each row is a design point.

Value

A matrix where each column contains values in $[-1, 1]^d$.

Examples

```
X <- transform11(quakes[, 1:4])
apply(X, 2, range)
```

Index