Package 'deepspat'

May 8, 2026

Type Package

Title Deep Compositional Spatial Models

Date 2025-11-20

Version 0.3.1

Maintainer Quan Vu <quanvustats@gmail.com>

Description Deep compositional spatial models are standard spatial covariance models coupled with an injective warping function of the spatial domain. The warping function is constructed through a composition of multiple elemental injective functions in a deep-learning framework. The package implements two cases for the univariate setting; first, when these warping functions are known up to some weights that need to be estimated, and, second, when the weights in each layer are random. In the multivariate setting only the former case is available. Estimation and inference is done using `tensorflow`, which makes use of graphics processing units.
For more details see Zammit-Mangion et al. (2022) <doi:10.1080/01621459.2021.1887741>,
Vu et al. (2022) <doi:10.5705/ss.202020.0156>,
Vu et al. (2023) <doi:10.1016/j.spasta.2023.100742>, and
Shao et al. (2025) <doi:10.48550/arXiv.2505.12548>.

License Apache License 2.0

Imports data.table, dplyr, Matrix, methods, reticulate, keras, tensorflow, tfprobability, evd, SpatialExtremes, fields

SystemRequirements TensorFlow (https://www.tensorflow.org/),

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

URL https://github.com/andrewzm/deepspat

BugReports https://github.com/andrewzm/deepspat/issues

Author Andrew Zammit-Mangion [aut],
Quan Vu [aut, cre],
Xuanjie Shao [aut]

Repository CRAN

Date/Publication 2025-11-25 11:32:16 UTC

Contents

`AFF_1D` *Affine transformation on a 1D domain*

Description

Sets up an affine transformation on a 1D domain

Usage

```
AFF_1D(a = c(0, 1), dtype = "float32")
```

Arguments

a	vector of two real numbers describing an affine transformation on a 1D domain
dtype	data type

Value

AFF_1D returns a list containing a list with the following components:

"f" An encapsulated function that takes an input and evaluates the affine transformation using TensorFlow

"fR" Same as f but uses R

"r" The number of basis functions (fixed to 1 in this case)

"trans" The transformation applied to the weights before estimation (in this case the identity)

"fix_weights" Flag indicating whether the weights are fixed or not (TRUE in this case)

"name" Name of layer

"pars" List of parameters describing the affine transformation as TensorFlow objects

---

AFF_2D *Affine transformation on a 2D domain*

Description

Sets up an affine transformation on a 2D domain

Usage

```
AFF_2D(a = c(0, 1, 0, 0, 0, 1), dtype = "float32")
```

Arguments

a	vector of six real numbers describing an affine transformation on a 2D domain
dtype	data type

Value

AFF_2D returns a list containing a list with the following components:

"f" An encapsulated function that takes an input and evaluates the affine transformation using TensorFlow

"fR" Same as f but uses R

"r" The number of basis functions (fixed to 1 in this case)

"trans" The transformation applied to the weights before estimation (in this case the identity)

"fix_weights" Flag indicating whether the weights are fixed or not (TRUE in this case)

"name" Name of layer

"pars" List of parameters describing the affine transformation as TensorFlow objects

AWU *Axial Warping Unit*

Description

Sets up an axial warping unit (AWU) for used in a deep compositional spatial model. The function sets up sigmoids on a prescribed domain at regular intervals, with 'steepness' indicated by the user. It returns a list of length 1 containing an axial warping unit (AWU) and several encapsulated functions that evaluate the AWU over inputs of different types. See Value for more details.

Usage

```
AWU(r = 50L, dim = 1L, grad = 200, lims = c(-0.5, 0.5), dtype = "float32")
```

Arguments

r	number of basis functions
dim	dimension to warp
grad	steepness of the sigmoid functions
lims	the bounded 1D domain on which to set up the sigmoids
dtype	data type

Value

AWU returns a list containing a list with the following components:

"f" An encapsulated function that takes an input and evaluates the sigmoids over the dim-th dimension using TensorFlow

"fR" Same as f but uses R

"fMC" Same as f but does it in parallel for several inputs index by the first dimension of the tensor

"r" The number of sigmoid basis functions

"trans" The transformation applied to the weights before estimation

"fix_weights" Flag indicating whether the weights are fixed or not (FALSE for AWUs)

"name" Name of layer

Examples

```
if (reticulate::py_module_available("tensorflow")) {
layer <- AWU(r = 50L, dim = 1L, grad = 200, lims = c(-0.5, 0.5))
 }
```

bisquares1D *Bisquare functions on a 1D domain*

Description

Sets up a top-layer set of bisquare basis functions on a bounded 1D domain of length 1 for modelling the process Y. It returns a list of length 1 containing the basis functions and encapsulated functions that evaluate the bisquare functions over inputs of different types. See Value for more details.

Usage

```
bisquares1D(r = 30, lims = c(-0.5, 0.5), dtype = "float32")
```

Arguments

r	30
lims	the limits of one side of the square bounded 2D domain on which to set up the bisquare functions
dtype	data type

Value

bisquares1D returns a list containing a list with the following components:

"f" An encapsulated function that takes an input and evaluates the sigmoids over the dim-th dimension using TensorFlow

"r" The number of sigmoid basis functions

"knots_tf" The centroids of the basis functions as a TensorFlow object

bisquares2D *Bisquare functions on a 2D domain*

Description

Sets up a top-layer set of bisquare basis functions on a square 2D domain of size 1 x 1 for modelling the process Y. It returns a list of length 1 containing the basis functions and encapsulated functions that evaluate the bisquare functions over inputs of different types. See Value for more details.

Usage

```
bisquares2D(r = 30, lims = c(-0.5, 0.5), dtype = "float32")
```

Arguments

r	30
lims	the bounded 1D domain on which to set up the bisquare functions
dtype	data type

Value

bisquares1D returns a list containing a list with the following components:

"f" An encapsulated function that takes an input and evaluates the sigmoids over the dim-th dimension using TensorFlow

"fR" Same as f but uses R

"r" The number of sigmoid basis functions

"knots_tf" The centroids of the basis functions as a TensorFlow object

"knots" The centroids of the basis functions as an R object

---

deepspat *Deep compositional spatial models*

Description

Deep compositional spatial models are standard low-rank spatial models coupled with a bijective warping function of the spatial domain. The warping function is constructed through a composition of multiple elemental bijective functions in a deep-learning framework. The package implements two cases; first, when these functions are known up to some weights that need to be estimated, and, second, when the weights in each layer are random. Estimation and inference is done using TensorFlow, which makes use of graphical processing units.

Constructs a deep compositional spatial model

Usage

```
deepspat(
  f,
  data,
  layers = NULL,
  method = c("VB", "ML"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  MC = 10L,
  nsteps
)
```

Arguments

f	formula identifying the dependent variable and the spatial inputs (RHS can only have one or two variables)
data	data frame containing the required data
layers	list containing the warping layers
method	either 'ML' (for the SIWGP) or 'VB' (for the SDSP)
par_init	list of initial parameter values. Call the function `initvars()` to see the structure of the list
learn_rates	learning rates for the various quantities in the model. Call the function `init_learn_rates()` to see the structure of the list
MC	number of MC samples when doing stochastic variational inference
nsteps	number of steps when doing gradient descent times three (first time the weights are optimised, then the covariance-function parameters, then everything together)

Value

`deepspat` returns an object of class `deepspat` with the following items

"Cost" The final value of the cost (NMLL for the SIWGP and the lower bound for the SDSP, plus a constant)

"mupost_tf" Posterior means of the weights in the top layer as a `TensorFlow` object

"Qpost_tf" Posterior precision of the weights in the top layer as a `TensorFlow` object

"eta_tf" Estimated or posterior means of the weights in the warping layers as a list of `TensorFlow` objects

"precy_tf" Precision of measurement error, as a `TensorFlow` object

"sigma2eta_tf" Variance of the weights in the top layer, as a `TensorFlow` object

"l_tf" Length scale used to construct the covariance matrix of the weights in the top layer, as a `TensorFlow` object

"scalings" Minima and maxima used to scale the unscaled unit outputs for each layer, as a list of `TensorFlow` objects

"method" Either 'ML' or 'VB'

"nlayers" Number of layers in the model (including the top layer)

"MC" Number of MC samples when doing stochastic variational inference

"run" `TensorFlow` session for evaluating the `TensorFlow` objects

"f" The formula used to construct the deepspat model

"data" The data used to construct the deepspat model

"negcost" Vector of costs after each gradient-descent evaluation

"data_scale_mean" Empirical mean of the original data

"data_scale_mean_tf" Empirical mean of the original data as a `TensorFlow` object

Author(s)

Maintainer: Quan Vu <quanvustats@gmail.com>

Authors:

- Andrew Zammit-Mangion
- Xuanjie Shao

See Also

Useful links:

- https://github.com/andrewzm/deepspat
- Report bugs at https://github.com/andrewzm/deepspat/issues

`deepspat_bivar_GP` *Deep bivariate compositional spatial model for Gaussian processes*

Description

Constructs a deep bivariate compositional spatial model

Usage

```
deepspat_bivar_GP(
  f,
  data,
  g = ~1,
  layers_asym = NULL,
  layers = NULL,
  method = "REML",
 family = c("exp_stat_symm", "exp_stat_asymm", "exp_nonstat_symm", "exp_nonstat_asymm",
    "matern_stat_symm", "matern_stat_asymm", "matern_nonstat_symm",
    "matern_nonstat_asymm"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)
```

Arguments

`f`	formula identifying the dependent variables and the spatial inputs in the covariance
`data`	data frame containing the required data
`g`	formula identifying the independent variables in the linear trend
`layers_asym`	list containing the aligning function layers

`layers`	list containing the nonstationary warping layers
`method`	identifying the method for finding the estimates
`family`	identifying the family of the model constructed
`par_init`	list of initial parameter values. Call the function `initvars()` to see the structure of the list
`learn_rates`	learning rates for the various quantities in the model. Call the function `init_learn_rates()` to see the structure of the list
`nsteps`	number of steps when doing gradient descent times two, three or five (depending on the family of model)

Value

`deepspat_bivar_GP` returns an object of class `deepspat_bivar_GP` with the following items

"f" The formula used to construct the covariance model

"g" The formula used to construct the linear trend model

"data" The data used to construct the deepspat model

"X" The model matrix of the linear trend

"layers" The warping function layers in the model

"layers_asym" The aligning function layers in the model

"Cost" The final value of the cost

"eta_tf" Estimated weights in the warping layers as a list of `TensorFlow` objects

"eta_tf_asym" Estimated weights in the aligning layers as a list of `TensorFlow` objects

"a_tf" Estimated parameters in the LFT layers

"a_tf_asym" Estimated parameters in the AFF layers of the aligning function

"beta" Estimated coefficients of the linear trend

"precy_tf1" Precision of measurement error of the first process, as a `TensorFlow` object

"precy_tf2" Precision of measurement error of the second process, as a `TensorFlow` object

"sigma2_tf_1" Variance parameter (first process) in the covariance matrix, as a `TensorFlow` object

"sigma2_tf_2" Variance parameter (second process) in the covariance matrix, as a `TensorFlow` object

"sigma2_tf_12" Covariance parameter in the covariance matrix, as a `TensorFlow` object

"l_tf1" Length scale parameter (first process) in the covariance matrix, as a `TensorFlow` object

"l_tf2" Length scale parameter (second process) in the covariance matrix, as a `TensorFlow` object

"l_tf12" Length scale parameter (cross-covariance) in the covariance matrix, as a `TensorFlow` object

"nu_tf1" Smoothness parameter (first process) in the covariance matrix, as a `TensorFlow` object

"nu_tf2" Smoothness parameter (second process) in the covariance matrix, as a `TensorFlow` object

"nu_tf12" Smoothness parameter (cross-covariance) in the covariance matrix, as a `TensorFlow` object

"scalings" Minima and maxima used to scale the unscaled unit outputs for each warping layer, as a list of `TensorFlow` objects

"scalings_asym" Minima and maxima used to scale the unscaled unit outputs for each aligning layer, as a list of `TensorFlow` objects

"method" Method used for inference

"nlayers" Number of warping layers in the model

"nlayers_asym" Number of aligning layers in the model

"swarped_tf1" Spatial locations of the first process on the warped domain

"swarped_tf2" Spatial locations of the second process on the warped domain

"negcost" Vector of costs after each gradient-descent evaluation

"z_tf_1" Data of the first process

"z_tf_2" Data of the second process

"family" Family of the model

Examples

```
if (reticulate::py_module_available("tensorflow")) {
df <- data.frame(s1 = rnorm(100), s2 = rnorm(100), z1 = rnorm(100), z2 = rnorm(100))
layers <- c(AWU(r = 50L, dim = 1L, grad = 200, lims = c(-0.5, 0.5)),
            AWU(r = 50L, dim = 2L, grad = 200, lims = c(-0.5, 0.5)))
d <- deepspat_bivar_GP(f = z1 + z2 ~ s1 + s2 - 1,
                       data = df, g = ~ 1,
                       layers = layers, method = "REML",
                       family = "matern_nonstat_symm",
                       nsteps = 10L)
 }
```

`deepspat_GP` *Deep compositional spatial model for Gaussian processes*

Description

Constructs a deep compositional spatial model

Usage

```
deepspat_GP(
  f,
  data,
  g = ~1,
  layers = NULL,
  method = c("REML"),
  family = c("exp_stat", "exp_nonstat", "matern_stat", "matern_nonstat"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)
```

Arguments

f	formula identifying the dependent variables and the spatial inputs in the covariance
data	data frame containing the required data
g	formula identifying the independent variables in the linear trend
layers	list containing the nonstationary warping layers
method	identifying the method for finding the estimates
family	identifying the family of the model constructed
par_init	list of initial parameter values. Call the function `initvars()` to see the structure of the list
learn_rates	learning rates for the various quantities in the model. Call the function `init_learn_rates()` to see the structure of the list
nsteps	number of steps when doing gradient descent times two or three (depending on the family of model)

Value

`deepspat_GP` returns an object of class `deepspat_GP` with the following items

"f" The formula used to construct the covariance model

"g" The formula used to construct the linear trend model

"data" The data used to construct the deepspat model

"X" The model matrix of the linear trend

"layers" The warping function layers in the model

"Cost" The final value of the cost

"eta_tf" Estimated weights in the warping layers as a list of `TensorFlow` objects

"a_tf" Estimated parameters in the LFT layers

"beta" Estimated coefficients of the linear trend

"precy_tf" Precision of measurement error, as a `TensorFlow` object

"sigma2_tf" Variance parameter in the covariance matrix, as a `TensorFlow` object

"l_tf" Length scale parameter in the covariance matrix, as a `TensorFlow` object

"nu_tf" Smoothness parameter in the covariance matrix, as a `TensorFlow` object

"scalings" Minima and maxima used to scale the unscaled unit outputs for each warping layer, as a list of `TensorFlow` objects

"method" Method used for inference

"nlayers" Number of warping layers in the model

"swarped_tf" Spatial locations on the warped domain

"negcost" Vector of costs after each gradient-descent evaluation

"z_tf" Data of the process

"family" Family of the model

Examples

```
if (reticulate::py_module_available("tensorflow")) {
df <- data.frame(s1 = rnorm(100), s2 = rnorm(100), z = rnorm(100))
layers <- c(AWU(r = 50L, dim = 1L, grad = 200, lims = c(-0.5, 0.5)),
            AWU(r = 50L, dim = 2L, grad = 200, lims = c(-0.5, 0.5)))
d <- deepspat_GP(f = z ~ s1 + s2 - 1,
                 data = df, g = ~ 1,
                 layers = layers, method = "REML",
                 family = "matern_nonstat",
                 nsteps = 10L)
 }
```

---

deepspat_MSP *Deep compositional spatial model for max-stable processes*

Description

Constructs an extended deep compositional spatial model that supports different estimation methods ("MPL", "MRPL", or "WLS") and spatial dependence families (stationary or non-stationary). This function extends the basic deepspat model by incorporating additional dependence modeling and pre-training steps for the warping layers.

Usage

```
deepspat_MSP(
  f,
  data,
  layers = NULL,
  method = c("MPL", "MRPL", "WLS"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  family = c("power_stat", "power_nonstat"),
  dtype = "float64",
  nsteps = 100L,
  nsteps_pre = 100L,
  edm_emp = NULL,
  p = c(0, 1),
  pen_coef = 0,
  show = TRUE,
  ...
)
```

Arguments

f	A formula identifying the dependent variable(s) and the spatial inputs. Use `get_depvars_multivar3` to extract the dependent variable names.

data	A data frame containing the required data.
layers	A list containing the warping layers; required for non-stationary models (i.e., when family = "power_nonstat").
method	A character string specifying the estimation method. Must be one of "MPL", "MRPL", or "WLS" for max-stable Brown-Resnick processes
par_init	A list of initial parameter values. Call the function initvars() to see the structure of the list.
learn_rates	A list of learning rates for the various quantities in the model. Call the function init_learn_rates() to see the structure of the list.
family	A character string specifying the spatial dependence model. Use "power_nonstat" for non-stationary models and "sta" for stationary models.
dtype	A character string indicating the data type for TensorFlow computations ("float32" or "float64"). Default is "float32"
nsteps	An integer specifying the number of training steps for dependence parameter learning.
nsteps_pre	An integer specifying the number of pre-training steps for warping layer parameters.
edm_emp	For the WLS method, a numeric vector or matrix providing an empirical extremal coefficients.
p	For pairwise likelihood based methods, p is used to specify the size of pair subset for pairwise likelihood, or the probability parameter of Bernoulli r.v. for randomized pairwise likelihood.
pen_coef	A penalty parameter for weights of SR-RBF(2) to relieve overfitting.
show	Logical; if TRUE progress information is printed during training.
...	Currently unused.

Value

deepspat_MSP returns an object of class deepspat_MSP which is a list containing the following components:

layers The list of warping layers used in the model.

Cost The final cost value after training (e.g., negative log-likelihood, least squares, or gradient score).

transeta_tf TensorFlow objects for the transformed dependence parameters in the warping layers.

eta_tf TensorFlow objects for the warped dependence parameters.

a_tf TensorFlow object for the parameters of the LFT layers (if applicable).

logphi_tf TensorFlow variable representing the logarithm of the spatial range parameter.

logitkappa_tf TensorFlow variable representing the logit-transformed degrees of freedom.

scalings A list of scaling limits (minima and maxima) for the input and warped spatial coordinates.

s_tf TensorFlow object for the scaled spatial coordinates.

`z_tf` TensorFlow object for the observed response values.

`swarped_tf` List of TensorFlow objects representing the warped spatial coordinates at each layer.

`swarped` Matrix of final warped spatial coordinates.

`method` The estimation method used (`"MPL"`, `"MRPL"`, or `"WLS"`).

`family` The spatial dependence family (`"power_stat"` or `"power_nonstat"`).

`dtype` The data type used in TensorFlow computations.

`nlayers` Number of warping layers (for non-stationary models).

`f` The model formula.

`data` The data frame used for model fitting.

`ndata` Number of observations in `data`.

`negcost` Vector of cost values recorded during training.

`pairs_tf` TensorFlow variable representing the spatial location pairs (and, for MRPL, the replicate indices) used in the pairwise / randomized pairwise likelihood or WLS objective..

`p` Input size of pair subset for pairwise likelihood, or the parameter of Bernoulli r.v. for randomized pairwise likelihood.

`time` Elapsed time for model fitting.

`deepspat_nn_GP` *Deep compositional spatial model (with nearest neighbors) for Gaussian processes*

Description

Constructs a deep compositional spatial model (with nearest neighbors)

Usage

```
deepspat_nn_GP(
  f,
  data,
  g = ~1,
  layers = NULL,
  m = 25L,
  order_id,
  nn_id,
  method = c("REML"),
  family = c("exp_stat", "exp_nonstat"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)
```

Arguments

f	formula identifying the dependent variables and the spatial inputs in the covariance
data	data frame containing the required data
g	formula identifying the independent variables in the linear trend
layers	list containing the nonstationary warping layers
m	number of nearest neighbors
order_id	indices of the order of the observations
nn_id	indices of the nearest neighbors of the ordered observations
method	identifying the method for finding the estimates
family	identifying the family of the model constructed
par_init	list of initial parameter values. Call the function `initvars()` to see the structure of the list
learn_rates	learning rates for the various quantities in the model. Call the function `init_learn_rates()` to see the structure of the list
nsteps	number of steps when doing gradient descent times two or three (depending on the family of model)

Value

`deepspat_nn_GP` returns an object of class `deepspat_nn_GP` with the following items

"f" The formula used to construct the covariance model

"g" The formula used to construct the linear trend model

"data" The data used to construct the deepspat model

"X" The model matrix of the linear trend

"layers" The warping function layers in the model

"Cost" The final value of the cost

"eta_tf" Estimated weights in the warping layers as a list of `TensorFlow` objects

"a_tf" Estimated parameters in the LFT layers

"beta" Estimated coefficients of the linear trend

"precy_tf" Precision of measurement error, as a `TensorFlow` object

"sigma2_tf" Variance parameter in the covariance matrix, as a `TensorFlow` object

"l_tf" Length scale parameter in the covariance matrix, as a `TensorFlow` object

"scalings" Minima and maxima used to scale the unscaled unit outputs for each warping layer, as a list of `TensorFlow` objects

"method" Method used for inference

"nlayers" Number of warping layers in the model

"swarped_tf" Spatial locations on the warped domain

"negcost" Vector of costs after each gradient-descent evaluation

"z_tf" Data of the process

"m" The number of nearest neighbors

"family" Family of the model

deepspat_nn_ST_GP *Deep compositional spatio-temporal model (with nearest neighbors) for Gaussian processes*

Description

Constructs a deep compositional spatio-temporal model (with nearest neighbors)

Usage

```
deepspat_nn_ST_GP(
  f,
  data,
  g = ~1,
  layers_spat = NULL,
  layers_temp = NULL,
  m = 25L,
  order_id,
  nn_id,
  method = c("REML"),
 family = c("exp_stat_sep", "exp_stat_asym", "exp_nonstat_sep", "exp_nonstat_asym"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)
```

Arguments

f	formula identifying the dependent variables and the spatial inputs in the covariance
data	data frame containing the required data
g	formula identifying the independent variables in the linear trend
layers_spat	list containing the spatial warping layers
layers_temp	list containing the temporal warping layers
m	number of nearest neighbors
order_id	indices of the order of the observations
nn_id	indices of the nearest neighbors of the ordered observations
method	identifying the method for finding the estimates
family	identifying the family of the model constructed
par_init	list of initial parameter values. Call the function `initvars()` to see the structure of the list
learn_rates	learning rates for the various quantities in the model. Call the function `init_learn_rates()` to see the structure of the list
nsteps	number of steps when doing gradient descent times two or three (depending on the family of model)

Value

deepspat_nn_ST_GP returns an object of class deepspat_nn_ST_GP with the following items

"f" The formula used to construct the covariance model

"g" The formula used to construct the linear trend model

"data" The data used to construct the deepspat model

"X" The model matrix of the linear trend

"layers_spat" The spatial warping function layers in the model

"layers_temp" The temporal warping function layers in the model

"Cost" The final value of the cost

"family" Family of the model

"eta_tf" Estimated weights in the spatial warping layers as a list of TensorFlow objects

"eta_t_tf" Estimated weights in the temporal warping layers as a list of TensorFlow objects

"a_tf" Estimated parameters in the LFT layers

"beta" Estimated coefficients of the linear trend

"precy_tf" Precision of measurement error, as a TensorFlow object

"sigma2_tf" Variance parameter in the covariance matrix, as a TensorFlow object

"v_tf" Parameters of the covariance matrix (indicating asymmetric spatio-temporal covariance)

"l_tf" Length scale (for spatial dimension) parameter in the covariance matrix, as a TensorFlow object

"l_t_tf" Length scale (for temporal dimension) parameter in the covariance matrix, as a TensorFlow object

"scalings" Minima and maxima used to scale the unscaled unit outputs for each spatial warping layer, as a list of TensorFlow objects

"scalings_t" Minima and maxima used to scale the unscaled unit outputs for each temporal warping layer, as a list of TensorFlow objects

"method" Method used for inference

"nlayers_spat" Number of spatial warping layers in the model

"nlayers_temp" Number of temporal warping layers in the model

"swarped_tf" Spatial locations on the warped domain

"twarped_tf" Temporal locations on the warped domain

"negcost" Vector of costs after each gradient-descent evaluation

"z_tf" Data of the process

"m" The number of nearest neighbors

deepspat_rPP *Deep compositional spatial model for r-Pareto processes*

Description

Constructs an extended deep compositional spatial model that supports different estimation methods ("GSM" or "WLS") and spatial dependence families (stationary or non-stationary). This function extends the basic deepspat model by incorporating additional dependence modeling and pre-training steps for the warping layers.

Usage

```
deepspat_rPP(
  f,
  data,
  layers = NULL,
  method = c("WLS", "GSM"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  family = c("power_stat", "power_nonstat"),
  dtype = "float32",
  nsteps = 100L,
  nsteps_pre = 100L,
  edm_emp = NULL,
  risk = NULL,
  thre = NULL,
  weight_fun = NULL,
  dWeight_fun = NULL,
  pen_coef = 0,
  show = TRUE,
  ...
)
```

Arguments

f	A formula identifying the dependent variable(s) and the spatial inputs. Use `get_depvars_multivar3` to extract the dependent variable names.
data	A data frame containing the required data.
layers	A list containing the warping layers; required for non-stationary models (i.e., when `family = "power_nonstat"`).
method	A character string specifying the estimation method. Must be one of `"GSM"`, or `"WLS"` for r-Pareto processes
par_init	A list of initial parameter values. Call the function `initvars()` to see the structure of the list.
learn_rates	A list of learning rates for the various quantities in the model. Call the function `init_learn_rates()` to see the structure of the list.

family	A character string specifying the spatial dependence model. Use "power_nonstat" for non-stationary models and "sta" for stationary models.
dtype	A character string indicating the data type for TensorFlow computations ("float32" or "float64"). Default is "float32"
nsteps	An integer specifying the number of training steps for dependence parameter learning.
nsteps_pre	An integer specifying the number of pre-training steps for warping layer parameters.
edm_emp	For the LS method, a numeric vector or matrix providing an empirical conditional exceedance probabilities.
risk	For the GS method, a numeric value indicating the risk parameter.
thre	A numeric threshold used in the GS method.
weight_fun	A function used to weight pairwise differences in the GS method.
dWeight_fun	A function representing the derivative of weight_fun (used in the GS method).
pen_coef	A penalty parameter for weights of SR-RBF(2) to relieve overfitting.
show	Logical; if TRUE progress information is printed during training.
...	Currently unused.

Value

deepspat_rPP returns an object of class deepspat_rPP which is a list containing the following components:

layers The list of warping layers used in the model.

Cost The final cost value after training (e.g., negative log-likelihood, least squares, or gradient score).

transeta_tf TensorFlow objects for the transformed dependence parameters in the warping layers.

eta_tf TensorFlow objects for the warped dependence parameters.

a_tf TensorFlow object for the parameters of the LFT layers (if applicable).

logphi_tf TensorFlow variable representing the logarithm of the spatial range parameter.

logitkappa_tf TensorFlow variable representing the logit-transformed degrees of freedom.

scalings A list of scaling limits (minima and maxima) for the input and warped spatial coordinates.

s_tf TensorFlow object for the scaled spatial coordinates.

z_tf TensorFlow object for the observed response values.

u_tf TensorFlow object for the threshold used in the GS method (if applicable).

swarped_tf List of TensorFlow objects representing the warped spatial coordinates at each layer.

swarped Matrix of final warped spatial coordinates.

method The estimation method used ("WLS" or "GSM").

risk The risk parameter used in the GS method (if applicable).

`family` The spatial dependence family (`"power_stat"` or `"power_nonstat"`).

`dtype` The data type used in TensorFlow computations.

`nlayers` Number of warping layers (for non-stationary models).

`weight_fun` The weighting function used in the GS method.

`dWeight_fun` The derivative of the weighting function used in the GS method.

`f` The model formula.

`data` The data frame used for model fitting.

`negcost` Vector of cost values recorded during training.

`pairs_tf` TensorFlow variable representing the spatial location pairs (and, for MRPL, the replicate indices) used in the pairwise / randomized pairwise likelihood or WLS objective..

`pairs_t_tf` Tranposed pairs_tf.

`time` Elapsed time for model fitting.

`deepspat_trivar_GP` *Deep trivariate compositional spatial model for Gaussian processes*

Description

Constructs a deep trivariate compositional spatial model

Usage

```
deepspat_trivar_GP(
  f,
  data,
  g = ~1,
  layers_asym_2 = NULL,
  layers_asym_3 = NULL,
  layers = NULL,
  method = c("REML"),
  family = c("matern_stat_symm", "matern_stat_asymm", "matern_nonstat_symm",
    "matern_nonstat_asymm"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)
```

Arguments

`f`	formula identifying the dependent variables and the spatial inputs in the covariance
`data`	data frame containing the required data
`g`	formula identifying the independent variables in the linear trend

layers_asym_2	list containing the aligning function layers for the second process
layers_asym_3	list containing the aligning function layers for the third process
layers	list containing the nonstationary warping layers
method	identifying the method for finding the estimates
family	identifying the family of the model constructed
par_init	list of initial parameter values. Call the function initvars() to see the structure of the list
learn_rates	learning rates for the various quantities in the model. Call the function init_learn_rates() to see the structure of the list
nsteps	number of steps when doing gradient descent times two, three or five (depending on the family of model)

Value

deepspat_trivar_GP returns an object of class deepspat_trivar_GP with the following items

"f" The formula used to construct the covariance model

"g" The formula used to construct the linear trend model

"data" The data used to construct the deepspat model

"X" The model matrix of the linear trend

"layers" The warping function layers in the model

"layers_asym_2" The aligning function layers for the second process in the model

"layers_asym_3" The aligning function layers for the third process in the model

"Cost" The final value of the cost

"eta_tf" Estimated weights in the warping layers as a list of TensorFlow objects

"eta_tf_asym_2" Estimated weights in the aligning layers for the second process as a list of TensorFlow objects

"eta_tf_asym_3" Estimated weights in the aligning layers for the third process as a list of TensorFlow objects

"beta" Estimated coefficients of the linear trend

"precy_tf1" Precision of measurement error of the first process, as a TensorFlow object

"precy_tf2" Precision of measurement error of the second process, as a TensorFlow object

"precy_tf3" Precision of measurement error of the third process, as a TensorFlow object

"sigma2_tf_1" Variance parameter (first process) in the covariance matrix, as a TensorFlow object

"sigma2_tf_2" Variance parameter (second process) in the covariance matrix, as a TensorFlow object

"sigma2_tf_3" Variance parameter (third process) in the covariance matrix, as a TensorFlow object

"sigma2_tf_12" Covariance parameter (between first and second process) in the covariance matrix, as a TensorFlow object

"sigma2_tf_13" Covariance parameter (between first and third process) in the covariance matrix, as a `TensorFlow` object

"sigma2_tf_23" Covariance parameter (between second and third process) in the covariance matrix, as a `TensorFlow` object

"l_tf1" Length scale parameter (first process) in the covariance matrix, as a `TensorFlow` object

"l_tf2" Length scale parameter (second process) in the covariance matrix, as a `TensorFlow` object

"l_tf3" Length scale parameter (third process) in the covariance matrix, as a `TensorFlow` object

"l_tf12" Length scale parameter (cross-covariance between first and second process) in the covariance matrix, as a `TensorFlow` object

"l_tf13" Length scale parameter (cross-covariance between first and third process) in the covariance matrix, as a `TensorFlow` object

"l_tf23" Length scale parameter (cross-covariance between second and third process) in the covariance matrix, as a `TensorFlow` object

"nu_tf1" Smoothness parameter (first process) in the covariance matrix, as a `TensorFlow` object

"nu_tf2" Smoothness parameter (second process) in the covariance matrix, as a `TensorFlow` object

"nu_tf3" Smoothness parameter (third process) in the covariance matrix, as a `TensorFlow` object

"nu_tf12" Smoothness parameter (cross-covariance between first and second process) in the covariance matrix, as a `TensorFlow` object

"nu_tf13" Smoothness parameter (cross-covariance between first and third process) in the covariance matrix, as a `TensorFlow` object

"nu_tf23" Smoothness parameter (cross-covariance between second and third process) in the covariance matrix, as a `TensorFlow` object

"scalings" Minima and maxima used to scale the unscaled unit outputs for each warping layer, as a list of `TensorFlow` objects

"scalings_asym" Minima and maxima used to scale the unscaled unit outputs for each aligning layer, as a list of `TensorFlow` objects

"method" Method used for inference

"nlayers" Number of warping layers in the model

"nlayers_asym" Number of aligning layers in the model

"run" `TensorFlow` session for evaluating the `TensorFlow` objects

"swarped_tf1" Spatial locations of the first process on the warped domain

"swarped_tf2" Spatial locations of the second process on the warped domain

"swarped_tf3" Spatial locations of the third process on the warped domain

"negcost" Vector of costs after each gradient-descent evaluation

"z_tf_1" Data of the first process

"z_tf_2" Data of the second process

"z_tf_3" Data of the third process

"family" Family of the model

initvars *Initialise weights and parameters*

Description

Provides utility to alter the initial weights and parameters when fitting a deepspat model

Usage

```
initvars(
  sigma2y = 0.1,
  l_top_layer = 0.5,
  sigma2eta_top_layer = 1,
  nu = 1.5,
  variogram_logrange = log(0.3),
  variogram_logitdf = 0.5,
  transeta_mean_init = list(AWU = -3, RBF = -0.8068528, RBF1 = -0.8068528, RBF2 =
    -0.8068528, LFT = 1, AFF_1D = 1, AFF_2D = 1),
  transeta_mean_prior = list(AWU = -3, RBF = -0.8068528, RBF1 = -0.8068528, RBF2 =
    -0.8068528, LFT = NA),
  transeta_sd_init = list(AWU = 0.01, RBF = 0.01, RBF1 = 0.01, RBF2 = 0.01, LFT = 0.01),
  transeta_sd_prior = list(AWU = 2, RBF = 2, RBF1 = 2, RBF2 = 0.01, LFT = NA)
)
```

Arguments

sigma2y	initial value for the measurement-error variance
l_top_layer	initial value for the length scale at the top layer
sigma2eta_top_layer	initial value for the variance of the weights at the top layer
nu	initial value for the smoothness parameter
variogram_logrange	initial value for variogram_logrange
variogram_logitdf	initial value for variogram_logitdf
transeta_mean_init	list of initial values for the initial weights (or the initial variational means of these weights). The list contains five values, one for the AWU, one for the RBF, one for the LFT (Mobius), and two for the affine transformation
transeta_mean_prior	same as transeta_mean_init but for the prior mean of the weights (SDSP only)
transeta_sd_init	same as transeta_mean_init but for the variational standard deviations (SDSP only)

transeta_sd_prior
: same as transeta_mean_init but for the prior standard deviations of the weights (SDSP only)

Value

initvars returns a list with the initial values. Call str(initvars()) to see the structure of this list.

---

init_learn_rates *Initialise learning rates*

Description

Provides utility to alter the learning rates when fitting a deepspat model

Usage

```
init_learn_rates(
  sigma2y = 5e-04,
  covfun = 0.01,
  sigma2eta = 1e-04,
  eta_mean = 0.1,
  eta_mean2 = 0.1,
  eta_sd = 0.1,
  LFTpars = 0.01,
  AFFpars = 0.01,
  rho = 0.1,
  vario = 0.1
)
```

Arguments

sigma2y	learning rate for the measurement-error variance
covfun	learning rate for the covariance-function (or matrix) parameters at the top layer
sigma2eta	learning rate for the process variance
eta_mean	learning rate for the weight estimates or variational means
eta_mean2	learning rate for the weight estimates or variational means
eta_sd	learning rate for the variational standard deviations (SDSP only)
LFTpars	learning rate for the parameters of the Mobius transformation
AFFpars	learning rate for the parameters of the affine transformation
rho	learning rate for the correlation parameter in the multivariate model
vario	learning rate for the parameter in the variogram

Value

`init_learn_rates` returns a list with the learning rates. Call `str(init_learn_rates())` to see the structure of this list.

`LFT` *LFT (Möbius transformation)*

Description

Sets up a Möbius transformation unit

Usage

```
LFT(a = NULL, dtype = "float32")
```

Arguments

`a`	vector of four complex numbers describing the Möbius transformation
`dtype`	data type

Value

`LFT` returns a list containing a list with the following components:

"f" An encapsulated function that takes an input and evaluates the Möbius transformation using `TensorFlow`

"fR" Same as `f` but uses `R`

"fMC" Same as `f` but does it in parallel for several inputs index by the first dimension of the tensor

"r" The number of basis functions (fixed to 1 in this case)

"trans" The transformation applied to the weights before estimation (in this case the identity)

"fix_weights" Flag indicating whether the weights are fixed or not (TRUE for LFTs)

"name" Name of layer

"pars" List of parameters describing the Möbius transformation as `TensorFlow` objects

Examples

```
if (reticulate::py_module_available("tensorflow")) {
layer <- LFT()
 }
```

predict.deepspat *Deep compositional spatial model*

Description

Prediction function for the fitted deepspat object

Usage

```
## S3 method for class 'deepspat'
predict(object, newdata, nsims = 100L, ...)
```

Arguments

object	the deepspat object
newdata	data frame containing the prediction locations
nsims	number of simulations from the Gaussian mixture components (SDSP only)
...	currently unused

Value

predict.deepspat returns a list with the two following items

"df_pred" Data frame containing the predictions/prediction intervals at the prediction locations

"allsims" Combined simulations from the Gaussian mixtures (SDSP only)

predict.deepspat_bivar_GP *Deep bivariate compositional spatial model*

Description

Prediction function for the fitted deepspat_bivar_GP object

Usage

```
## S3 method for class 'deepspat_bivar_GP'
predict(object, newdata, ...)
```

Arguments

object	the deepspat_bivar_GP object
newdata	data frame containing the prediction locations
...	currently unused

Value

predict.deepspat_bivar_GP returns a list with the following item

"df_pred" Data frame containing the predictions/prediction intervals at the prediction locations

"obs_swarped1" Observation locations on the warped domain (for the first process)

"obs_swarped2" Observation locations on the warped domain (for the second process)

"newdata_swarped1" New prediction locations on the warped domain (for the first process)

"newdata_swarped2" New prediction locations on the warped domain (for the second process)

---

predict.deepspat_GP *Deep compositional spatial model*

Description

Prediction function for the fitted deepspat_GP object

Usage

```
## S3 method for class 'deepspat_GP'
predict(object, newdata, ...)
```

Arguments

object	the deepspat_GP object
newdata	data frame containing the prediction locations
...	currently unused

Value

predict.deepspat_GP returns a list with the following item

"df_pred" Data frame containing the predictions/prediction intervals at the prediction locations

"obs_swarped" Observation locations on the warped domain

"newdata_swarped" New prediction locations on the warped domain

predict.deepspat_nn_GP

Deep compositional spatial model (with nearest neighbors)

Description

Prediction function for the fitted deepspat_nn_GP object

Usage

```
## S3 method for class 'deepspat_nn_GP'
predict(object, newdata, nn_id, ...)
```

Arguments

object	the deepspat_nn_GP object
newdata	data frame containing the prediction locations
nn_id	nearest neighbors index
...	currently unused

Value

predict.deepspat_nn_GP returns a list with the following item

"df_pred" Data frame containing the predictions/prediction intervals at the prediction locations

"obs_swarped" Observation locations on the warped domain

"newdata_swarped" New prediction locations on the warped domain

predict.deepspat_nn_ST_GP

Deep compositional spatio-temporal model (with nearest neighbors)

Description

Prediction function for the fitted deepspat_nn_ST_GP object

Usage

```
## S3 method for class 'deepspat_nn_ST_GP'
predict(object, newdata, nn_id, ...)
```

Arguments

object	the deepspat_nn_ST_GP object
newdata	data frame containing the prediction locations
nn_id	nearest neighbors index
...	currently unused

Value

predict.deepspat_nn_ST_GP returns a list with the following item

"df_pred" Data frame containing the predictions/prediction intervals at the prediction locations

"obs_swarped" Observation locations on the spatial warped domain

"obs_twarped" Observation locations on the temporal warped domain

"newdata_swarped" New prediction locations on the spatial warped domain

"newdata_twarped" New prediction locations on the temporal warped domain

---

predict.deepspat_trivar_GP

Deep trivariate compositional spatial model

Description

Prediction function for the fitted deepspat_trivar_GP object

Usage

```
## S3 method for class 'deepspat_trivar_GP'
predict(object, newdata, ...)
```

Arguments

object	the deepspat_trivar_GP object
newdata	data frame containing the prediction locations
...	currently unused

Value

predict.deepspat_trivar_GP returns a list with the following item

"df_pred" Data frame containing the predictions/prediction intervals at the prediction locations

RBF_block *Radial Basis Function Warpings*

Description

Sets up a composition of radial basis functions (RBFs) for used in a deep compositional spatial model. The function sets up RBFs on a prescribed domain on a grid at a certain resolution. It returns a list containing all the functions in the single-resolution RBF unit. See Value for more details.

Usage

```
RBF_block(res = 1L, lims = c(-0.5, 0.5), dtype = "float32")
```

Arguments

res	the resolution
lims	the limits of one side of the square 2D domain on which to set up the RBFs
dtype	data type

Value

RBF_block returns a list containing a list for each RBF in the block with the following components:

"f" An encapsulated function that takes an input and evaluates the RBF over some input using TensorFlow

"fR" Same as f but uses R

"fMC" Same as f but does it in parallel for several inputs index by the first dimension of the tensor

"r" The number of basis functions (one for each layer)

"trans" The transformation applied to the weights before estimation

"fix_weights" Flag indicating whether the weights are fixed or not (FALSE for RBFs)

"name" Name of layer

Examples

```
if (reticulate::py_module_available("tensorflow")) {
layer <- RBF_block(res = 1L)
 }
```

set_deepspat_seed *Set TensorFlow seed*

Description

Set TensorFlow seed in deepspat package

Usage

```
set_deepspat_seed(seed = 1L)
```

Arguments

seed	the seed

Value

No return value, called for side effects.

Examples

```
if (reticulate::py_module_available("tensorflow")) {
set_deepspat_seed(seed = 1L)
 }
```

sim_data *Generate simulation data for testing*

Description

Generates simulated data for use in experiments

Usage

```
sim_data(type = "step1D", ds = 0.001, n = 300L, sigma2y = NULL)
```

Arguments

type	type of function. Can be 'step1D', 'Monterrubio1D', 'dampedwave1D', 'step2D', 'AWU_RBF_2D', or 'AWU_RBF_LFT_2D'
ds	spatial grid length
n	number of data points
sigma2y	measurement-error variance

Value

`sim_data` returns a list containing the following items:

"s" Process locations on a fine grid with spacing `ds`

"sobs" Observation locations

"swarped" The warping function (when this is also simulated)

"f_true" The true process on the fine grid

"y" The simulated observation data

Examples

```
if (reticulate::py_module_available("tensorflow")) {
sim <- sim_data(type = "step1D", ds = 0.001)
 }
```

`summary.deepspat_MSP` *Deep compositional spatial model for max-stable processes*

Description

Prediction function for the fitted deepspat_ext object

Usage

```
## S3 method for class 'deepspat_MSP'
summary(object, newdata, uncAss = TRUE, edm_emp = NULL, ...)
```

Arguments

`object`	a deepspat object obtained from fitting a deep compositional spatial model for extremes using max-stable processes.
`newdata`	a data frame containing the prediction locations.
`uncAss`	assess the uncertainty of dependence parameters or not
`edm_emp`	empirical estimates of extremal dependence measure for weighted least square inference method
`...`	currently unused

Value

A list with the following components:

srescaled A matrix of rescaled spatial coordinates produced by scaling the input locations.

swarped A matrix of warped spatial coordinates. For `family = "power_stat"` this equals `srescaled`, while for `family = "power_nonstat"` the coordinates are further transformed through additional layers.

fitted.phi A numeric value representing the fitted spatial range parameter, computed as `exp(logphi_tf)`.

fitted.kappa A numeric value representing the fitted smoothness parameter, computed as `2 * sigmoid(logitkappa_tf)`.

Sigma.psi A numeric matrix giving the estimated covariance matrix of the dependence parameters $\psi = (\phi, \kappa)$, computed via the pairwise likelihood / WLS sandwich-type estimator. `NULL` if `uncAss = FALSE`.

`summary.deepspat_rPP` *Deep compositional spatial model for r-Pareto processes*

Description

Prediction function for the fitted deepspat_ext object

Usage

```
## S3 method for class 'deepspat_rPP'
summary(object, newdata, uncAss = TRUE, edm_emp = NULL, uprime = NULL, ...)
```

Arguments

`object`	a deepspat object obtained from fitting a deep compositional spatial model for extremes using r-Pareto processes.
`newdata`	a data frame containing the prediction locations.
`uncAss`	assess the uncertainty of dependence parameters or not
`edm_emp`	empirical estimates of extremal dependence measure for weighted least square inference method
`uprime`	uprime for weighted least square inference method
`...`	currently unused.

Value

A list with the following components:

srescaled A matrix of rescaled spatial coordinates produced by scaling the input locations.

swarped A matrix of warped spatial coordinates. For `family = "power_stat"` this equals `srescaled`, while for `family = "power_nonstat"` the coordinates are further transformed through additional layers.

fitted.phi A numeric value representing the fitted spatial range parameter, computed as `exp(logphi_tf)`.

fitted.kappa A numeric value representing the fitted smoothness parameter, computed as `2 * sigmoid(logitkappa_tf)`.

Sigma.psi A numeric matrix giving the estimated covariance matrix of the dependence parameters $\psi = (\phi, \kappa)$, computed via the pairwise likelihood / WLS sandwich-type estimator. `NULL` if `uncAss = FALSE`.

Index