Package 'shrinkGPR'

May 9, 2026

Type Package

Title Scalable Gaussian Process Regression with Hierarchical Shrinkage Priors

Version 2.0.0

Maintainer Peter Knaus <peter.knaus@wu.ac.at>

Description Efficient variational inference methods for fully Bayesian univariate and multivariate Gaussian and t-process regression models. Hierarchical shrinkage priors, including the triple gamma prior, are used for effective variable selection and covariance shrinkage in high-dimensional settings. The package leverages normalizing flows to approximate complex posterior distributions. For details on implementation, see Knaus (2025) <doi:10.48550/arXiv.2501.13173>.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports gsl, progress, rlang, utils, methods, torch (>= 0.16.0), mniw

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0), shrinkTVP, plotly

Config/testthat/edition 3

SystemRequirements torch backend, for installation guide see https://cran.r-project.org/web/packages/torch/vignettes/installation.html

NeedsCompilation no

Author Peter Knaus [aut, cre] (ORCID: <https://orcid.org/0000-0001-6498-7084>)

Repository CRAN

Date/Publication 2026-03-30 14:10:03 UTC

Contents

`calc_pred_moments` *Calculate Predictive Moments*

Description

`calc_pred_moments` calculates the predictive means and variances for a fitted `shrinkGPR`, `shrinkTPR`, `shrinkMVGPR`, or `shrinkMVTPR` model at new data points.

Usage

```
calc_pred_moments(object, newdata, nsamp = 100)
```

Arguments

`object`	A `shrinkGPR`, `shrinkTPR`, `shrinkMVGPR`, or `shrinkMVTPR` object representing the fitted univariate or multivariate Gaussian or t process regression model.
`newdata`	*Optional* data frame containing the covariates for the new data points. If missing, the training data is used.
`nsamp`	Positive integer specifying the number of posterior samples to use for the calculation. Default is 100.

Details

This function computes predictive moments by marginalizing over posterior samples from the fitted model. If a mean equation was included in the model, the corresponding covariates are used to calculate the predictive mean.

Value

For univariate models (shrinkGPR, shrinkTPR), a list with:

- means: An array of predictive means, with the first dimension corresponding to samples, the second to data points.
- vars: An array of predictive variances, with the first dimension corresponding to samples and second and third to data points.

Additionally, for a shrinkTPR model, the list also includes:

- nu: A vector of posterior degrees of freedom of length nsamp.

For multivariate models (shrinkMVGPR, shrinkMVTPR), a list with:

- means: An array of predictive means of shape nsamp x N_new x M.
- K: An array of posterior row covariance matrices of shape nsamp x N_new x N_new.
- Omega: An array of posterior column covariance matrices of shape nsamp x M x M.
- nu: (shrinkMVTPR only) A vector of posterior degrees of freedom of length nsamp.

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
  data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

  # Fit GPR model
  res <- shrinkGPR(y ~ x1 + x2, data = data)

  # Calculate predictive moments
  momes <- calc_pred_moments(res, nsamp = 100)
  }
```

---

eval_pred_dens *Evaluate Predictive Densities*

Description

eval_pred_dens evaluates the predictive density for a set of points based on a fitted shrinkGPR, shrinkTPR, shrinkMVGPR, or shrinkMVTPR model.

Usage

```
eval_pred_dens(x, mod, data_test, nsamp = 100, log = FALSE)
```

Arguments

x	For univariate models (shrinkGPR, shrinkTPR): a numeric vector of response values at which to evaluate the density. For multivariate models (shrinkMVGPR, shrinkMVTPR): a numeric matrix with M columns, where each row is a candidate response vector.
mod	A shrinkGPR, shrinkTPR, shrinkMVGPR, or shrinkMVTPR object representing the fitted model.
data_test	Data frame with one row containing the covariates for the test set. Variables in data_test must match those used in model fitting.
nsamp	Positive integer specifying the number of posterior samples to use for the evaluation. Default is 100.
log	Logical; if TRUE, returns the log predictive density. Default is FALSE.

Details

This function computes predictive densities by marginalizing over posterior samples drawn from the fitted model. If a mean equation was included in the model, the corresponding covariates are used to calculate the predictive mean.

Value

A numeric vector containing the predictive densities (or log predictive densities) for the points in x.

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
  data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

  # Fit GPR model
  res <- shrinkGPR(y ~ x1 + x2, data = data)

  # Create point at which to evaluate predictive density
  data_test <- data.frame(x1 = 0.8, x2 = 0.5)
  eval_points <- c(-1.2, -1, 0)

  eval_pred_dens(eval_points, res, data_test)

  # Is vectorized, can also be used in functions like curve
  curve(eval_pred_dens(x, res, data_test), from = -1.5, to = -0.5)
  abline(v = sin(2 * pi * 0.8), col = "red")
  }
```

gen_marginal_samples *Generate Marginal Samples of Predictive Distribution*

Description

gen_marginal_samples() generates model predictions over a grid of values for one or two specified covariates, while filling in the remaining covariates either by drawing from the training data (if fixed_x is not provided) or by using a fixed values for the remaining covariates (if fixed_x is provided). The result is a set of conditional predictions that can be used to visualize the marginal effect of the selected covariates under varying input configurations.

Usage

```
gen_marginal_samples(
  mod,
  to_eval,
  nsamp = 200,
  fixed_x,
  n_eval_points,
  eval_range,
  display_progress = TRUE
)
```

Arguments

mod	A shrinkGPR, shrinkTPR, shrinkMVGPR or shrinkMVTPR object representing the fitted Gaussian/t process regression model.
to_eval	A character vector specifying the names of the covariates to evaluate. Can be one or two variables.
nsamp	Positive integer specifying the number of posterior samples to generate. Default is 200.
fixed_x	*optional* data frame specifying a fixed covariate configuration. If provided, this configuration is used for all nonswept covariates. If omitted, covariates are sampled from the training data.
n_eval_points	Positive integer specifying the number of evaluation points along each axis. If missing, defaults to 100 for 1D and 30 for 2D evaluations.
eval_range	*optional* numeric vector (1D) or list of two numeric vectors (2D) specifying the range over which to evaluate the covariates in to_eval. If omitted, the range is set to the range of the swept covariate(s) in the training data.
display_progress	logical value indicating whether to display progress bars and messages during training. The default is TRUE.

Details

This function generates conditional predictive surfaces by evaluating the fitted model across a grid of values for one or two selected covariates. For each of the `nsamp` draws, the remaining covariates are either held fixed (if `fixed_x` is provided) or filled in by sampling a single row from the training data. The selected covariates in `to_eval` are then varied across a regular grid defined by `n_eval_points` and `eval_range`, and model predictions are computed using `calc_pred_moments`.

The resulting samples represent conditional predictions across different covariate contexts, and can be used to visualize marginal effects, interaction surfaces, or predictive uncertainty.

Note that computational and memory requirements increase rapidly with grid size. In particular, for two-dimensional evaluations, the kernel matrix scales quadratically with the number of evaluation points per axis. Large values of `n_eval_points` may lead to high memory usage during prediction, especially when using a GPU. If memory constraints arise, consider reducing `n_eval_points`.

Value

A list containing posterior predictive summaries over the evaluation grid:

- `mean_pred`: A matrix (1D case) or array (2D case) of predicted means for each evaluation point and posterior sample.
- `grid`: The evaluation grid used to generate predictions. A numeric vector (1D) or a named list of two vectors (`grid1`, `grid2`) for 2D evaluations.

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
  data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

  # Fit GPR model
  res <- shrinkGPR(y ~ x1 + x2, data = data)

  # Generate marginal samples for x1
  marginal_samps <- gen_marginal_samples(res, to_eval = "x1", nsamp = 100)

  # Plot marginal predictions
  plot(marginal_samps)
  }
```

gen_posterior_samples *Generate Posterior Samples*

Description

gen_posterior_samples generates posterior samples of the model parameters from a fitted shrinkGPR, shrinkTPR, shrinkMVGPR or shrinkMVTPR model.

Usage

```
gen_posterior_samples(mod, nsamp = 1000)
```

Arguments

mod	A shrinkGPR, shrinkTPR, shrinkMVGPR or shrinkMVTPR object representing the fitted model.
nsamp	Positive integer specifying the number of posterior samples to generate. Default is 1000.

Details

This function draws posterior samples from the latent space and transforms them into the parameter space of the model. These samples can be used for posterior inference or further analysis, such as examining which inverse lengthscale parameters pulled to zero.

Value

For univariate models (shrinkGPR, shrinkTPR), a list containing posterior samples of the model parameters:

- thetas: A matrix of posterior samples for the inverse lengthscale parameters.
- sigma2: A matrix of posterior samples for the noise variance.
- tau: A matrix of posterior samples for the global shrinkage parameter.
- betas (optional): A matrix of posterior samples for the mean equation parameters (if included in the model).
- tau_mean (optional): A matrix of posterior samples for the mean equation's global shrinkage parameter (if included in the model).

Additionally, for a shrinkTPR model, the list also includes:

- nu: A matrix of posterior samples for the degrees of freedom parameter.

For multivariate models (shrinkMVGPR, shrinkMVTPR), the list contains:

- thetas: A matrix of posterior samples for the inverse lengthscale parameters.
- tau: A matrix of posterior samples for the global shrinkage parameter for the kernel.
- sigma2: A matrix of posterior samples for the noise variance.

- tau_Om: A matrix of posterior samples for the global shrinkage parameter for the output covariance.
- Omega: An array of posterior samples for the output covariance matrix.

Additionally, for a shrinkMVTPR model, the list also includes:

- nu: A matrix of posterior samples for the degrees of freedom parameter.

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
  data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

  # Fit GPR model
  res <- shrinkGPR(y ~ x1 + x2, data = data)

  # Generate posterior samples
  samps <- gen_posterior_samples(res, nsamp = 1000)

  # Plot the posterior samples
  boxplot(samps$thetas)
  }
```

---

kernel_functions *Kernel Functions for Gaussian and t Processes*

Description

A set of kernel functions for Gaussian and t processes, including the squared exponential (SE) kernel and Matérn kernels with smoothness parameters 1/2, 3/2, and 5/2. These kernels compute the covariance structure for Gaussian and t process regression models and are designed for compatibility with the shrinkGPR, shrinkTPR, shrinkMVGPR and shrinkMVTPR functions.

Usage

```
kernel_se(thetas, tau, x, x_star = NULL)

kernel_matern_12(thetas, tau, x, x_star = NULL)

kernel_matern_32(thetas, tau, x, x_star = NULL)

kernel_matern_52(thetas, tau, x, x_star = NULL)
```

Arguments

thetas	A torch_tensor of dimensions n_latent x d, representing the latent length-scale parameters.
tau	A torch_tensor of length n_latent, representing the latent scaling factors.
x	A torch_tensor of dimensions N x d, containing the input data points.
x_star	Either NULL or a torch_tensor of dimensions N_new x d. If NULL, the kernel is computed for x against itself. Otherwise, it computes the kernel between x and x_star.

Details

These kernel functions are used to define the covariance structure in Gaussian process regression models. Each kernel implements a specific covariance function:

- kernel_se: Squared exponential (SE) kernel, also known as the radial basis function (RBF) kernel. It assumes smooth underlying functions.
- kernel_matern_12: Matérn kernel with smoothness parameter $\nu = 1/2$, equivalent to the absolute exponential kernel.
- kernel_matern_32: Matérn kernel with smoothness parameter $\nu = 3/2$.
- kernel_matern_52: Matérn kernel with smoothness parameter $\nu = 5/2$.

The sqdist helper function is used internally by these kernels to compute squared distances between data points.

Note that these functions perform no input checks, as to ensure higher performance. Users should ensure that the input tensors are of the correct dimensions.

Value

A torch_tensor containing the batched covariance matrices (one for each latent sample):

- If x_star = NULL, the output is of dimensions n_latent x N x N, representing pairwise covariances between all points in x.
- If x_star is provided, the output is of dimensions n_latent x N_new x N, representing pairwise covariances between x_star and x.

Examples

```
if (torch::torch_is_installed()) {
  # Example inputs
  torch::torch_manual_seed(123)
  n_latent <- 3
  d <- 2
  N <- 5
  thetas <- torch::torch_randn(n_latent, d)$abs()
  tau <- torch::torch_randn(n_latent)$abs()
  x <- torch::torch_randn(N, d)

  # Compute the SE kernel
```

```
  K_se <- kernel_se(thetas, tau, x)
  print(K_se)

  # Compute the Matérn 3/2 kernel
  K_matern32 <- kernel_matern_32(thetas, tau, x)
  print(K_matern32)

  # Compute the Matérn 5/2 kernel with x_star
  x_star <- torch::torch_randn(3, d)
  K_matern52 <- kernel_matern_52(thetas, tau, x, x_star)
  print(K_matern52)
}
```

`load_shrinkGPR` *Load a saved shrinkGPR model object from disk*

Description

`load_shrinkGPR` restores a model object previously saved by `save_shrinkGPR`, reconstructing the trained model, optimizer state, and all metadata. The loaded object can be used directly for prediction or passed as `cont_model` to continue training.

Usage

```
load_shrinkGPR(file)
```

Arguments

`file`	a character string specifying the file path to load from.

Details

If the model was originally trained on CUDA and CUDA is available on the current machine, the model is loaded on CUDA. Otherwise, it is loaded on CPU with an informative message. The optimizer state (including Adam momentum and adaptive learning rate accumulators) is fully restored, so continued training via `cont_model` picks up where the original run left off.

Value

An object of the same class as the one that was saved, with the same structure as returned by `shrinkGPR`, `shrinkTPR`, `shrinkMVGPR`, or `shrinkMVTPR`.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

See Also

`save_shrinkGPR`

Examples

```
if (torch::torch_is_installed()) {
  # Fit a model and save it
  sim <- simGPR()
  mod <- shrinkGPR(y ~ ., data = sim$data)
  tmp <- tempfile(fileext = ".zip")
  save_shrinkGPR(mod, tmp)

  # Load and predict
  mod_loaded <- load_shrinkGPR(tmp)
  preds <- predict(mod_loaded, newdata = sim$data[1:10, ])
  }
```

LPDS *Log Predictive Density Score*

Description

LPDS calculates the log predictive density score for a fitted shrinkGPR, shrinkTPR, shrinkMVGPR, or shrinkMVTPR model using a test dataset.

Usage

```
LPDS(mod, data_test, nsamp = 100)
```

Arguments

mod	A shrinkGPR, shrinkTPR, shrinkMVGPR, or shrinkMVTPR object representing the fitted model.
data_test	Data frame with one row containing the covariates for the test set. Variables in data_test must match those used in model fitting.
nsamp	Positive integer specifying the number of posterior samples to use for the evaluation. Default is 100.

Details

The log predictive density score is a measure of model fit that evaluates how well the model predicts unseen data. It is computed as the log of the marginal predictive density at the true observed responses.

Value

A numeric value representing the log predictive density score for the test dataset.

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
  data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

  # Fit GPR model
  res <- shrinkGPR(y ~ x1 + x2, data = data)

  # Calculate true y value and calculate LPDS at specific point
  x1_new <- 0.8
  x2_new <- 0.5
  y_true <- sin(2 * pi * x1_new)
  data_test <- data.frame(y = y_true, x1 = x1_new, x2 = x2_new)
  LPDS(res, data_test)
  }
```

`plot.shrinkGPR` *Graphical summary of posterior of theta*

Description

`plot.shrinkGPR` generates a boxplot visualizing the posterior distribution of `theta` obtained from a fitted `shrinkGPR` object.

Usage

```
## S3 method for class 'shrinkGPR'
plot(x, nsamp = 1000, ...)
```

Arguments

`x`	a `shrinkGPR` object.
`nsamp`	a positive integer specifying the number of posterior samples to draw for plotting. The default is `1000`.
`...`	further arguments passed to the internal `boxplot` function, such as axis labeling or plotting options. By default, `las = 2` is used unless explicitly overridden by the user.

Value

Called for its side effects. Returns `invisible(NULL)`.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

See Also

Other plotting functions: plot.shrinkGPR_marg_samples_1D(), plot.shrinkGPR_marg_samples_2D(), plot.shrinkMVGPR(), plot.shrinkTPR()

Examples

```
if (torch::torch_is_installed()) {
 # Simulate and fit a shrinkGPR model, then plot:
 sim <- simGPR()
 mod <- shrinkGPR(y ~ ., data = sim$data)
 plot(mod)

 ## Change axis label orientation
 plot(mod, las = 1)
 }
```

---

plot.shrinkGPR_marg_samples_1D

Plot method for 1D marginal predictions

Description

Generates a plot of 1D conditional predictive samples produced by gen_marginal_samples with a single covariate.

Usage

```
## S3 method for class 'shrinkGPR_marg_samples_1D'
plot(x, ...)
```

Arguments

x	An object of class "shrinkGPR_marg_samples_1D", typically returned by gen_marginal_samples when providing a single covariate to sweep over.
...	Additional arguments passed to plot.mcmc.tvp for customizing the plot, such as axis labels or plotting options.

Details

By default, the function visualizes the posterior predictive median and 95% and 50% credible intervals for the selected covariate across a grid of evaluation points. Axis labels are automatically inferred if not explicitly provided.

Note: The **shrinkTVP** package must be installed to use this function.

Value

Called for its side effects. Returns `invisible(NULL)`.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

See Also

`gen_marginal_samples`

Other plotting functions: `plot.shrinkGPR()`, `plot.shrinkGPR_marg_samples_2D()`, `plot.shrinkMVGPR()`, `plot.shrinkTPR()`

Examples

```
if (torch::torch_is_installed()) {
# Simulate data
set.seed(123)
torch::torch_manual_seed(123)
n <- 100
x <- matrix(runif(n * 2), n, 2)
y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

# Fit GPR model
res <- shrinkGPR(y ~ x1 + x2, data = data)

# Generate marginal samples
marginal_samps_x1 <- gen_marginal_samples(res, to_eval = "x1", nsamp = 100)
marginal_samps_x2 <- gen_marginal_samples(res, to_eval = "x2", nsamp = 100)

# Plot marginal predictions
plot(marginal_samps_x1)
plot(marginal_samps_x2)

# Customize plot appearance (see plot.mcmc.tvp from shrinkTVP package for more options)
plot(marginal_samps_x2, shaded = FALSE, quantlines = TRUE, quantcol = "red")
}
```

---

plot.shrinkGPR_marg_samples_2D

Plot method for 2D marginal predictions

Description

Generates a 3D surface plot of 2D conditional predictive samples produced by `gen_marginal_samples`.

Usage

```
## S3 method for class 'shrinkGPR_marg_samples_2D'
plot(x, ...)
```

Arguments

x	An object of class "shrinkGPR_marg_samples_2D", typically returned by gen_marginal_samples when evaluating two covariates.
...	Additional arguments passed to add_surface for customizing the appearance of the surface plot (e.g., opacity, colorscale, showscale).

Details

The function visualizes the posterior predictive mean across a 2D grid of evaluation points for two covariates. Interactive plotting is handled via the **plotly** package. Axis labels are automatically inferred from the object.

Note: The **plotly** package must be installed to use this function.

Value

A plotly plot object (interactive 3D surface). Can be further customized via pipes and plotly functions.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

See Also

gen_marginal_samples, plot_ly, add_surface

Other plotting functions: plot.shrinkGPR(), plot.shrinkGPR_marg_samples_1D(), plot.shrinkMVGPR(), plot.shrinkTPR()

Examples

```
if (torch::torch_is_installed()) {
# Simulate data
set.seed(123)
torch::torch_manual_seed(123)
n <- 100
x <- matrix(runif(n * 2), n, 2)
y <- sin(2 * pi * x[, 1]) * cos(2 * pi * x[, 2]) + rnorm(n, sd = 0.1)
data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

# Fit GPR model
res <- shrinkGPR(y ~ x1 + x2, data = data)

# Generate marginal predictions for x1 and x2 jointly
marginal_samps_both <- gen_marginal_samples(res, to_eval = c("x1", "x2"), nsamp = 100)
```

```
# Plot interactive surface
plot(marginal_samps_both)

# Customize surface appearance
plot(marginal_samps_both, opacity = 0.8, colorscale = "Magma")

# Customize further via plotly
p <- plot(marginal_samps_both, opacity = 0.8, colorscale = "Magma")

p |> plotly::layout(
  scene = list(
    xaxis = list(title = "Covariate 1"),
    yaxis = list(title = "Covariate 2"),
    zaxis = list(title = "Expected value")
  )
)
}
```

`plot.shrinkMVGPR` *Graphical summary of posterior of theta*

Description

`plot.shrinkMVGPR` generates a boxplot visualizing the posterior distribution of `theta` obtained from a fitted `shrinkMVGPR` object.

Usage

```
## S3 method for class 'shrinkMVGPR'
plot(x, nsamp = 1000, ...)
```

Arguments

`x`	a `shrinkMVGPR` object.
`nsamp`	a positive integer specifying the number of posterior samples to draw for plotting. The default is `1000`.
`...`	further arguments passed to the internal `boxplot` function, such as axis labeling or plotting options. By default, `las = 2` is used unless explicitly overridden by the user.

Value

Called for its side effects. Returns `invisible(NULL)`.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

See Also

Other plotting functions: plot.shrinkGPR(), plot.shrinkGPR_marg_samples_1D(), plot.shrinkGPR_marg_samples_2D(), plot.shrinkTPR()

Examples

```
if (torch::torch_is_installed()) {
# Simulate and fit a shrinkMVGPR model, then plot:
sim <- simMVGPR()
mod <- shrinkMVGPR(cbind(y.1, y.2) ~ ., data = sim$data)
plot(mod)

## Change axis label orientation
plot(mod, las = 1)
}
```

---

plot.shrinkTPR *Graphical summary of posterior of theta*

Description

plot.shrinkTPR generates a boxplot visualizing the posterior distribution of theta obtained from a fitted shrinkTPR object.

Usage

```
## S3 method for class 'shrinkTPR'
plot(x, nsamp = 1000, ...)
```

Arguments

x	a shrinkTPR object.
nsamp	a positive integer specifying the number of posterior samples to draw for plotting. The default is 1000.
...	further arguments passed to the internal boxplot function, such as axis labeling or plotting options. By default, las = 2 is used unless explicitly overridden by the user.

Value

Called for its side effects. Returns invisible(NULL).

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

See Also

Other plotting functions: plot.shrinkGPR(), plot.shrinkGPR_marg_samples_1D(), plot.shrinkGPR_marg_samples_2D(), plot.shrinkMVGPR()

Examples

```
if (torch::torch_is_installed()) {
# Simulate and fit a shrinkTPR model, then plot:
sim <- simGPR()
mod <- shrinkTPR(y ~ ., data = sim$data)
plot(mod)

## Change axis label orientation
plot(mod, las = 1)
}
```

---

predict.shrinkGPR *Generate Predictions*

Description

predict.shrinkGPR generates posterior predictive samples from a fitted shrinkGPR model at specified covariates.

Usage

```
## S3 method for class 'shrinkGPR'
predict(object, newdata, nsamp = 100, ...)
```

Arguments

object	A shrinkGPR object representing the fitted Gaussian process regression model.
newdata	*Optional* data frame containing the covariates for the prediction points. If missing, the training data is used.
nsamp	Positive integer specifying the number of posterior samples to generate. Default is 100.
...	Currently ignored.

Details

This function generates predictions by sampling from the posterior predictive distribution. If the mean equation is included in the model, the corresponding covariates are incorporated.

Value

A matrix containing posterior predictive samples for each covariate combination in newdata.

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
  data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

  # Fit GPR model
  res <- shrinkGPR(y ~ x1 + x2, data = data)
  # Example usage for in-sample prediction
  preds <- predict(res)

  # Example usage for out-of-sample prediction
  newdata <- data.frame(x1 = runif(10), x2 = runif(10))
  preds <- predict(res, newdata = newdata)
  }
```

---

predict.shrinkMVGPR *Generate Predictions*

Description

predict.shrinkMVGPR generates posterior predictive samples from a fitted shrinkMVGPR model at specified covariates.

Usage

```
## S3 method for class 'shrinkMVGPR'
predict(object, newdata, nsamp = 100, ...)
```

Arguments

object	A shrinkMVGPR or shrinkMVTPR object representing the fitted multivariate process regression model.
newdata	*Optional* data frame containing the covariates for the prediction points. If missing, the training data is used.
nsamp	Positive integer specifying the number of posterior samples to generate. Default is 100.
...	Currently ignored.

Details

This function generates predictions by sampling from the posterior predictive distribution.

Value

A 3-dimensional array of dimensions nsamp x N_new x M containing posterior predictive samples for each covariate combination in newdata.

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y1 <- sin(2 * pi * x[, 1])
  y2 <- cos(2 * pi * x[, 2])
  y <- cbind(y1, y2) + matrix(rnorm(n * 2, sd = 0.1), n, 2)
  data <- data.frame(y1 = y1, y2 = y2, x1 = x[, 1], x2 = x[, 2])

  # Fit MVGPR model
  res <- shrinkMVGPR(cbind(y1, y2) ~ x1 + x2, data = data)
  # Example usage for in-sample prediction
  preds <- predict(res)

  # Example usage for out-of-sample prediction
  newdata <- data.frame(x1 = runif(10), x2 = runif(10))
  preds <- predict(res, newdata = newdata)
  }
```

---

predict.shrinkTPR *Generate Predictions*

Description

predict.shrinkTPR generates posterior predictive samples from a fitted shrinkTPR model at specified covariates.

Usage

```
## S3 method for class 'shrinkTPR'
predict(object, newdata, nsamp = 100, ...)
```

Arguments

object	A shrinkTPR object representing the fitted Student-t process regression model.
newdata	*Optional* data frame containing the covariates for the prediction points. If missing, the training data is used.
nsamp	Positive integer specifying the number of posterior samples to generate. Default is 100.
...	Currently ignored.

Details

This function generates predictions by sampling from the posterior predictive distribution. If the mean equation is included in the model, the corresponding covariates are incorporated.

Value

A matrix containing posterior predictive samples for each covariate combination in `newdata`.

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
  data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

  # Fit GPR model
  res <- shrinkTPR(y ~ x1 + x2, data = data)
  # Example usage for in-sample prediction
  preds <- predict(res)

  # Example usage for out-of-sample prediction
  newdata <- data.frame(x1 = runif(10), x2 = runif(10))
  preds <- predict(res, newdata = newdata)
  }
```

---

save_shrinkGPR *Save a fitted shrinkGPR model object to disk*

Description

`save_shrinkGPR` saves a fitted model object to a single `.zip` file, preserving the trained model, optimizer state, and all metadata needed for prediction and continued training via `cont_model`.

Usage

```
save_shrinkGPR(obj, file)
```

Arguments

`obj`	an object of class `shrinkGPR`, `shrinkTPR`, `shrinkMVGPR`, or `shrinkMVTPR`, as returned by the corresponding fitting function.
`file`	a character string specifying the file path to save to.

Details

Internally, the model components are saved as separate files via `torch_save` and bundled into a single `.zip` archive. This is necessary because torch `nn_module` objects contain external pointers that cannot be serialized within a nested list. The plain R components (loss history, model internals) are saved via `saveRDS`.

Value

Invisibly returns the file path.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

See Also

`load_shrinkGPR`

Examples

```
if (torch::torch_is_installed()) {
  # Fit a model and save it
  sim <- simGPR()
  mod <- shrinkGPR(y ~ ., data = sim$data)
  tmp <- tempfile(fileext = ".zip")
  save_shrinkGPR(mod, tmp)

  # Load it back
  mod_loaded <- load_shrinkGPR(tmp)

  # Continue training
  mod2 <- shrinkGPR(y ~ ., data = sim$data, cont_model = mod_loaded)
  }
```

`shrinkGPR` *Gaussian Process Regression with Shrinkage and Normalizing Flows*

Description

Fits a Gaussian process regression (GPR) model to an $N \times 1$ response vector Y. The joint distribution is multivariate normal $Y \sim \mathcal{N}(0,\, K + \sigma^2 I)$, where K is the GP kernel matrix with triple-gamma shrinkage priors on the inverse length-scales. Covariates whose length-scales are shrunk to zero have no influence on the covariance structure. An optional linear mean $x_\mu^\top \beta$ can be added via `formula_mean`. The joint posterior is approximated by normalizing flows trained to maximize the ELBO.

Usage

```
shrinkGPR(
  formula,
  data,
  a = 0.5,
  c = 0.5,
  formula_mean,
  a_mean = 0.5,
  c_mean = 0.5,
  sigma2_rate = 10,
  kernel_func = kernel_se,
  n_layers = 10,
  n_latent = 10,
  flow_func = sylvester,
  flow_args,
  n_epochs = 1000,
  auto_stop = TRUE,
  cont_model,
  device,
  display_progress = TRUE,
  optim_control
)
```

Arguments

formula	object of class "formula": a symbolic representation of the model for the covariance equation, as in lm. The response variable and covariates are specified here.
data	*optional* data frame containing the response variable and the covariates. If not found in data, the variables are taken from environment(formula). No NAs are allowed in the response variable or covariates.
a	positive real number controlling the behavior at the origin of the shrinkage prior for the covariance structure. The default is 0.5.
c	positive real number controlling the tail behavior of the shrinkage prior for the covariance structure. The default is 0.5.
formula_mean	*optional* formula for the linear mean equation. If provided, the covariates for the mean structure are specified separately from the covariance structure. A response variable is not required in this formula.
a_mean	positive real number controlling the behavior at the origin of the shrinkage for the mean structure. The default is 0.5.
c_mean	positive real number controlling the tail behavior of the shrinkage prior for the mean structure. The default is 0.5.
sigma2_rate	positive real number controlling the prior rate parameter for the residual variance. The default is 10.
kernel_func	function specifying the covariance kernel. The default is kernel_se, a squared exponential kernel. For guidance on how to provide a custom kernel function, see Details.

`n_layers` positive integer specifying the number of flow layers in the normalizing flow. The default is 10.

`n_latent` positive integer specifying the dimensionality of the latent space for the normalizing flow. The default is 10.

`flow_func` function specifying the normalizing flow transformation. The default is `sylvester`. For guidance on how to provide a custom flow function, see Details.

`flow_args` *optional* named list containing arguments for the flow function. If not provided, default arguments are used. For guidance on how to provide a custom flow function, see Details.

`n_epochs` positive integer specifying the number of training epochs. The default is 1000.

`auto_stop` logical value indicating whether to enable early stopping based on convergence. The default is `TRUE`.

`cont_model` *optional* object returned from a previous `shrinkGPR` call, enabling continuation of training from the saved state.

`device` *optional* device to run the model on, e.g., `torch_device("cuda")` for GPU or `torch_device("cpu")` for CPU. Defaults to GPU if available; otherwise, CPU.

`display_progress` logical value indicating whether to display progress bars and messages during training. The default is `TRUE`.

`optim_control` *optional* named list containing optimizer parameters. If not provided, default settings are used.

Details

Model Specification

Given N observations with d-dimensional covariates x_i, the model is

$$y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2),$$

$$f \sim \mathcal{GP}(\mu(\cdot),\ k(\cdot, \cdot\,;\, \theta, \tau)).$$

The default squared exponential kernel is

$$k(x, x'; \theta, \tau) = \frac{1}{\tau} \exp\left(-\frac{1}{2} \sum_{j=1}^{d} \theta_j (x_j - x'_j)^2 \right),$$

where $\theta_j \geq 0$ are inverse squared length-scales and $\tau > 0$ is the output scale. Users can specify custom kernels by following the guidelines below, or use one of the other provided kernel functions in `kernel_functions`. If `formula_mean` is provided, the GP mean becomes $x_{\mu,i}^{\top}\beta$ instead of zero.

Priors

The triple-gamma (TG) prior (Cadonna et al. 2020) induces sparsity on inactive length-scales:

$$\theta_j \mid \tau \sim \mathrm{TG}(a, c, \tau), \quad j = 1, \ldots, d,$$

$$\tau \sim F(2c, 2a),$$

$$\sigma^2 \sim \text{Exp}(\sigma^2_{\text{rate}}).$$

With a mean structure, the regression coefficients receive a normal-gamma-gamma (NGG) prior:

$$\beta_k \mid \tau_\mu \sim \text{NGG}(a_\mu, c_\mu, \tau_\mu), \quad \tau_\mu \sim F(2c_\mu, 2a_\mu).$$

Parameters a and c jointly control the spike at zero (a) and the tail decay (c) of the prior.

Inference

The posterior is approximated by a normalizing flow q_ϕ trained to maximize the ELBO. The default flow is a Sylvester flow (van den Berg et al. 2018), but users can specify custom flows by following the guidelines below. `auto_stop` triggers early stopping when the ELBO shows no significant improvement over the last 100 iterations.

Custom Kernel Functions

Users can define custom kernel functions by passing them to the `kernel_func` argument. A valid kernel function must follow the same structure as `kernel_se`. The function should:

1. **Accept the following arguments:**
   - `thetas`: A `torch_tensor` of dimensions `n_latent` x `d`, representing latent length-scale parameters.
   - `tau`: A `torch_tensor` of length `n_latent`, representing latent scaling factors.
   - `x`: A `torch_tensor` of dimensions `N` x `d`, containing the input data points.
   - `x_star`: Either `NULL` or a `torch_tensor` of dimensions `N_new` x `d`. If `NULL`, the kernel is computed for `x` against itself. Otherwise, it computes the kernel between `x` and `x_star`.
2. **Return:**
   - If `x_star` = `NULL`: a `torch_tensor` of dimensions `n_latent` x `N` x `N`.
   - If `x_star` is provided: a `torch_tensor` of dimensions `n_latent` x `N_new` x `N`.
3. **Requirements:**
   - The kernel must produce a valid positive semi-definite covariance matrix.
   - Use `torch` tensor operations to ensure GPU/CPU compatibility.

See `kernel_functions` for documented examples.

Custom Flow Functions

Users can define custom flow functions by implementing an `nn_module` in `torch`. The module must have a `forward` method that accepts a tensor `z` of shape `n_latent` x `D` and returns a list with:

- `zk`: the transformed samples, shape `n_latent` x `D`.
- `log_diag_j`: the log-absolute-determinant of the Jacobian, shape `n_latent`.

See `sylvester` for a documented example.

Value

A list object of class `shrinkGPR`, containing:

`model`	The best-performing trained model.
`loss`	The best loss value (ELBO) achieved during training.

loss_stor	A numeric vector storing the ELBO values at each training iteration.
last_model	The model state at the final iteration.
optimizer	The optimizer object used during training.
model_internals	Internal objects required for predictions and further training, such as model matrices and formulas.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
  data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

  # Fit GPR model
  res <- shrinkGPR(y ~ x1 + x2, data = data)

  # Check convergence
  plot(res$loss_stor, type = "l", main = "Loss Over Iterations")

  # Check posterior
  samps <- gen_posterior_samples(res, nsamp = 1000)
  boxplot(samps$thetas) # Second theta is pulled towards zero

  # Predict
  x1_new <- seq(from = 0, to = 1, length.out = 100)
  x2_new <- runif(100)
  y_new <- predict(res, newdata = data.frame(x1 = x1_new, x2 = x2_new), nsamp = 2000)

  # Plot
  quants <- apply(y_new, 2, quantile, c(0.025, 0.5, 0.975))
  plot(x1_new, quants[2, ], type = "l", ylim = c(-1.5, 1.5),
       xlab = "x1", ylab = "y", lwd = 2)
  polygon(c(x1_new, rev(x1_new)), c(quants[1, ], rev(quants[3, ])),
       col = adjustcolor("skyblue", alpha.f = 0.5), border = NA)
  points(x[,1], y)
  curve(sin(2 * pi * x), add = TRUE, col = "forestgreen", lwd = 2, lty = 2)



  # Add mean equation
  res2 <- shrinkGPR(y ~ x1 + x2, formula_mean = ~ x1, data = data)
  }
```

shrinkMVGPR *Multivariate Gaussian Process Regression with Shrinkage and Normalizing Flows*

Description

Fits a multivariate Gaussian process regression (MVGPR) model to an $N \times M$ response matrix Y. The joint distribution is matrix normal, $Y \sim \mathcal{MN}(0,\, K + \sigma^2 I,\, \Omega)$, where K is the GP kernel matrix with triple-gamma shrinkage priors on the inverse length-scales, and Ω is an $M \times M$ output covariance matrix with an LKJ prior on its correlations and triple-gamma priors on its scale parameters. The joint posterior is approximated by normalizing flows trained to maximize the ELBO.

Usage

```
shrinkMVGPR(
  formula,
  data,
  a = 0.5,
  c = 0.5,
  eta = 4,
  a_Om = 0.5,
  c_Om = 0.5,
  sigma2_rate = 10,
  kernel_func = kernel_se,
  n_layers = 10,
  n_latent = 10,
  flow_func = sylvester,
  flow_args,
  n_epochs = 1000,
  auto_stop = TRUE,
  cont_model,
  device,
  display_progress = TRUE,
  optim_control
)
```

Arguments

formula	object of class "formula": a symbolic representation of the model for the covariance equation, as in lm. The response variable and covariates are specified here. Specifically, the response is created by binding the M response variables together with cbind() on the left-hand side of the formula, e.g., cbind(y1, y2) ~ x.
data	*optional* data frame containing the response variable and the covariates. If not found in data, the variables are taken from environment(formula). No NAs are allowed in the response variable or covariates.

a	positive real number controlling the behavior at the origin of the shrinkage prior for the covariance structure. The default is 0.5.
c	positive real number controlling the tail behavior of the shrinkage prior for the covariance structure. The default is 0.5.
eta	positive real number controlling the concentration of the LKJ prior on the correlation matrix of the output covariance. Higher values push the prior towards the identity matrix. The default is 4.
a_Om	positive real number controlling the behavior at the origin of the shrinkage prior for the output covariance scale parameters. The default is 0.5.
c_Om	positive real number controlling the tail behavior of the shrinkage prior for the output covariance scale parameters. The default is 0.5.
sigma2_rate	positive real number controlling the prior rate parameter for the residual variance. The default is 10.
kernel_func	function specifying the covariance kernel. The default is `kernel_se`, a squared exponential kernel. For guidance on how to provide a custom kernel function, see Details.
n_layers	positive integer specifying the number of flow layers in the normalizing flow. The default is 10.
n_latent	positive integer specifying the dimensionality of the latent space for the normalizing flow. The default is 10.
flow_func	function specifying the normalizing flow transformation. The default is `sylvester`. For guidance on how to provide a custom flow function, see Details.
flow_args	*optional* named list containing arguments for the flow function. If not provided, default arguments are used. For guidance on how to provide a custom flow function, see Details.
n_epochs	positive integer specifying the number of training epochs. The default is 1000.
auto_stop	logical value indicating whether to enable early stopping based on convergence. The default is `TRUE`.
cont_model	*optional* object returned from a previous `shrinkMVGPR` call, enabling continuation of training from the saved state.
device	*optional* device to run the model on, e.g., `torch_device("cuda")` for GPU or `torch_device("cpu")` for CPU. Defaults to GPU if available; otherwise, CPU.
display_progress	logical value indicating whether to display progress bars and messages during training. The default is `TRUE`.
optim_control	*optional* named list containing optimizer parameters. If not provided, default settings are used.

Details

Model Specification

Given N observations with d-dimensional covariates and M response variables, the response matrix $Y \in \mathbb{R}^{N \times M}$ follows a matrix normal distribution:

$$Y \sim \mathcal{MN}_{N,M}(0,\ K(\theta, \tau) + \sigma^2 I_N,\ \Omega),$$

which is equivalent to

$$\mathrm{vec}(Y) \sim \mathcal{N}_{NM}(0,\ \Omega \otimes (K + \sigma^2 I_N)).$$

Here $K_{ij} = k(x_i, x_j;\, \theta, \tau)$ is the kernel matrix and Ω is the $M \times M$ between-response covariance. The output covariance is parameterized as $\Omega = SDS$, where D is a correlation matrix and $S = \mathrm{diag}(s_1, \ldots, s_M)$ contains the marginal standard deviations. The product of the diagonal elements of S is constrained to equal 1 to ensure identifiability. The default squared exponential kernel is

$$k(x, x';\, \theta, \tau) = \frac{1}{\tau} \exp\left(-\frac{1}{2} \sum_{j=1}^{d} \theta_j (x_j - x'_j)^2 \right),$$

where $\theta_j \geq 0$ are inverse squared length-scales and $\tau > 0$ is the output scale. Users can specify custom kernels by following the guidelines below, or use one of the other provided kernel functions in `kernel_functions`.

Priors

$$\begin{gathered}
\theta_j \mid \tau \sim \mathrm{TG}(a, c, \tau), \quad j = 1, \ldots, d, \\
\tau \sim F(2c, 2a), \\
\sigma^2 \sim \mathrm{Exp}(\sigma^2_{\mathrm{rate}}), \\
D \sim \mathrm{LKJ}(\eta), \\
s_m \mid \tau_\Omega \sim \mathrm{TG}(a_\Omega, c_\Omega, \tau_\Omega), \quad m = 1, \ldots, M, \\
\tau_\Omega \sim F(2c_\Omega, 2a_\Omega).
\end{gathered}$$

The LKJ(η) prior on the correlation matrix D is uniform over correlations when $\eta = 1$ and concentrates near the identity as η increases.

Inference

The posterior is approximated by a normalizing flow q_ϕ trained to maximize the ELBO. `auto_stop` triggers early stopping when the ELBO shows no significant improvement over the last 100 iterations.

Custom Kernel Functions

Users can define custom kernel functions by passing them to the `kernel_func` argument. A valid kernel function must follow the same structure as `kernel_se`. The function must:

1. Accept arguments `thetas` (`n_latent` x `d`), `tau` (length `n_latent`), `x` (`N` x `d`), and optionally `x_star` (`N_new` x `d`).
2. Return a `torch_tensor` of dimensions `n_latent` x `N` x `N` (if `x_star` = `NULL`) or `n_latent` x `N_new` x `N` (if `x_star` is provided).
3. Produce a valid positive semi-definite covariance matrix using `torch` tensor operations.

See `kernel_functions` for documented examples.

Custom Flow Functions

Users can define custom flow functions by implementing an `nn_module` in `torch`. The module must have a `forward` method that accepts a tensor `z` of shape `n_latent` x `D` and returns a list with:

- `zk`: the transformed samples, shape `n_latent` x `D`.
- `log_diag_j`: log-absolute-determinant of the Jacobian, shape `n_latent`.

See `sylvester` for a documented example.

Value

A list object of class shrinkMVGPR, containing:

model	The best-performing trained model.
loss	The best loss value (ELBO) achieved during training.
loss_stor	A numeric vector storing the ELBO values at each training iteration.
last_model	The model state at the final iteration.
optimizer	The optimizer object used during training.
model_internals	Internal objects required for predictions and further training, such as model matrices and formulas.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

Examples

```
if (torch::torch_is_installed()) {
  # Simulate multivariate data
  torch::torch_manual_seed(123)
  sim <- simMVGPR(N = 100, M = 2, d = 2)

  # Fit MVGPR model
  res <- shrinkMVGPR(cbind(y.1, y.2) ~ x.1 + x.2, data = sim$data)

  # Check convergence
  plot(res$loss_stor, type = "l", main = "Loss Over Iterations")

  # Check posterior of length-scale parameters
  samps <- gen_posterior_samples(res, nsamp = 1000)
  boxplot(samps$thetas)

  # Predict at new covariate values
  newdata <- data.frame(x.1 = runif(10), x.2 = runif(10))
  y_new <- predict(res, newdata = newdata, nsamp = 500)
  # y_new is an array of shape nsamp x N_new x M
  }
```

---

shrinkMVTPR *Multivariate Student-t Process Regression with Shrinkage and Normalizing Flows*

Description

Fits a multivariate Student-t process regression (MVTPR) model to an $N \times M$ response matrix Y. The joint distribution is matrix-variate Student-t, $Y \sim \mathcal{MT}(\nu, 0, K + \sigma^2 I, \Omega)$, where K is the GP kernel matrix with triple-gamma shrinkage priors on the inverse length-scales, Ω is the $M \times M$ output covariance, and ν is the degrees of freedom parameter. Compared to `shrinkMVGPR`, the heavier tails provide greater robustness to outliers. The joint posterior is approximated by normalizing flows trained to maximize the ELBO.

Usage

```
shrinkMVTPR(
  formula,
  data,
  a = 0.5,
  c = 0.5,
  eta = 4,
  a_Om = 0.5,
  c_Om = 0.5,
  sigma2_rate = 10,
  nu_alpha = 0.5,
  nu_beta = 2,
  kernel_func = kernel_se,
  n_layers = 10,
  n_latent = 10,
  flow_func = sylvester,
  flow_args,
  n_epochs = 1000,
  auto_stop = TRUE,
  cont_model,
  device,
  display_progress = TRUE,
  optim_control
)
```

Arguments

`formula`	object of class "formula": a symbolic representation of the model for the covariance equation, as in `lm`. The response variable and covariates are specified here. Specifically, the response is created by binding the M response variables together with `cbind()` on the left-hand side of the formula, e.g., `cbind(y1, y2) ~ x`.
`data`	*optional* data frame containing the response variable and the covariates. If not found in `data`, the variables are taken from `environment(formula)`. No `NA`s are allowed in the response variable or covariates.
`a`	positive real number controlling the behavior at the origin of the shrinkage prior for the covariance structure. The default is 0.5.
`c`	positive real number controlling the tail behavior of the shrinkage prior for the covariance structure. The default is 0.5.

`eta`	positive real number controlling the concentration of the LKJ prior on the correlation matrix of the output covariance. Higher values push the prior towards the identity matrix. The default is 4.
`a_Om`	positive real number controlling the behavior at the origin of the shrinkage prior for the output covariance scale parameters. The default is 0.5.
`c_Om`	positive real number controlling the tail behavior of the shrinkage prior for the output covariance scale parameters. The default is 0.5.
`sigma2_rate`	positive real number controlling the prior rate parameter for the residual variance. The default is 10.
`nu_alpha`	positive real number controlling the shape parameter of the gamma prior for the degrees of freedom of the matrix-t process. The default is 0.5.
`nu_beta`	positive real number controlling the rate parameter of the shifted gamma prior for the degrees of freedom of the matrix-t process. The default is 2.
`kernel_func`	function specifying the covariance kernel. The default is `kernel_se`, a squared exponential kernel. For guidance on how to provide a custom kernel function, see Details.
`n_layers`	positive integer specifying the number of flow layers in the normalizing flow. The default is 10.
`n_latent`	positive integer specifying the dimensionality of the latent space for the normalizing flow. The default is 10.
`flow_func`	function specifying the normalizing flow transformation. The default is `sylvester`. For guidance on how to provide a custom flow function, see Details.
`flow_args`	*optional* named list containing arguments for the flow function. If not provided, default arguments are used. For guidance on how to provide a custom flow function, see Details.
`n_epochs`	positive integer specifying the number of training epochs. The default is 1000.
`auto_stop`	logical value indicating whether to enable early stopping based on convergence. The default is `TRUE`.
`cont_model`	*optional* object returned from a previous `shrinkMVTPR` call, enabling continuation of training from the saved state.
`device`	*optional* device to run the model on, e.g., `torch_device("cuda")` for GPU or `torch_device("cpu")` for CPU. Defaults to GPU if available; otherwise, CPU.
`display_progress`	logical value indicating whether to display progress bars and messages during training. The default is `TRUE`.
`optim_control`	*optional* named list containing optimizer parameters. If not provided, default settings are used.

Details

Model Specification

Given N observations with d-dimensional covariates and M response variables, the response matrix $Y \in \mathbb{R}^{N \times M}$ follows a matrix-variate Student-t distribution:

$$Y \sim \mathcal{MT}_{N,M}(\nu,\ 0,\ K(\theta, \tau) + \sigma^2 I_N,\ \Omega),$$

which is equivalent to

$$\mathrm{vec}(Y) \sim t_{NM}\big(\nu,\ \mathbf{0},\ \Omega \otimes (K + \sigma^2 I_N)\big)\,.$$

Here $K_{ij} = k(x_i, x_j;\, \theta, \tau)$ is the kernel matrix and Ω is the $M \times M$ between-response covariance. The output covariance is parameterized as $\Omega = SDS$, where D is a correlation matrix and $S = \mathrm{diag}(s_1, \ldots, s_M)$ contains the marginal standard deviations. The product of the diagonal elements of S is constrained to equal 1 to ensure identifiability. The default squared exponential kernel is

$$k(x, x';\, \theta, \tau) = \frac{1}{\tau} \exp\left(-\frac{1}{2} \sum_{j=1}^{d} \theta_j (x_j - x'_j)^2 \right),$$

where $\theta_j \geq 0$ are inverse squared length-scales and $\tau > 0$ is the output scale. Users can specify custom kernels by following the guidelines below, or use one of the other provided kernel functions in `kernel_functions`.

Priors

$$\begin{gathered}
\theta_j \mid \tau \sim \mathrm{TG}(a, c, \tau), \quad j = 1, \ldots, d, \\
\tau \sim F(2c, 2a), \\
\sigma^2 \sim \mathrm{Exp}(\sigma^2_{\mathrm{rate}}), \\
D \sim \mathrm{LKJ}(\eta), \\
s_m \mid \tau_\Omega \sim \mathrm{TG}(a_\Omega, c_\Omega, \tau_\Omega), \quad m = 1, \ldots, M, \\
\tau_\Omega \sim F(2c_\Omega, 2a_\Omega), \\
\nu - 2 \sim \mathrm{Gamma}(\nu_\alpha, \nu_\beta).
\end{gathered}$$

The shift by 2 ensures $\nu > 2$ so that the process covariance is finite.

Inference

The posterior is approximated by a normalizing flow q_ϕ trained to maximize the ELBO. `auto_stop` triggers early stopping when the ELBO shows no significant improvement over the last 100 iterations.

Custom Kernel Functions

Users can define custom kernel functions by passing them to the `kernel_func` argument. A valid kernel function must follow the same structure as `kernel_se`. The function must:

1. Accept arguments `thetas` (`n_latent` x `d`), `tau` (length `n_latent`), `x` (`N` x `d`), and optionally `x_star` (`N_new` x `d`).
2. Return a `torch_tensor` of dimensions `n_latent` x `N` x `N` (if `x_star` = `NULL`) or `n_latent` x `N_new` x `N` (if `x_star` is provided).
3. Produce a valid positive semi-definite covariance matrix using `torch` tensor operations.

See `kernel_functions` for documented examples.

Custom Flow Functions

Users can define custom flow functions by implementing an `nn_module` in `torch`. The module must have a `forward` method that accepts a tensor `z` of shape `n_latent` x `D` and returns a list with:

- `zk`: the transformed samples, shape `n_latent` x `D`.
- `log_diag_j`: log-absolute-determinant of the Jacobian, shape `n_latent`.

See `sylvester` for a documented example.

Value

A list object of classes shrinkMVGPR and shrinkMVTPR, containing:

model	The best-performing trained model.
loss	The best loss value (ELBO) achieved during training.
loss_stor	A numeric vector storing the ELBO values at each training iteration.
last_model	The model state at the final iteration.
optimizer	The optimizer object used during training.
model_internals	Internal objects required for predictions and further training, such as model matrices and formulas.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

Examples

```
if (torch::torch_is_installed()) {
  # Simulate multivariate data
  torch::torch_manual_seed(123)
  sim <- simMVGPR(N = 100, M = 2, d = 2)

  # Fit MVTPR model
  res <- shrinkMVTPR(cbind(y.1, y.2) ~ x.1 + x.2, data = sim$data)

  # Check convergence
  plot(res$loss_stor, type = "l", main = "Loss Over Iterations")

  # Check posterior of length-scale parameters
  samps <- gen_posterior_samples(res, nsamp = 1000)
  boxplot(samps$thetas)

  # Predict at new covariate values
  newdata <- data.frame(x.1 = runif(10), x.2 = runif(10))
  y_new <- predict(res, newdata = newdata, nsamp = 500)
  # y_new is an array of shape nsamp x N_new x M
  }
```

Description

Fits a Student-t process regression (TPR) model (Shah et al. 2014) with triple-gamma shrinkage priors on the inverse length-scale parameters θ_j. Compared to shrinkGPR, the Student-t process has heavier tails governed by the degrees of freedom ν, providing greater robustness to outliers. An optional linear mean can be added via formula_mean. The joint posterior is approximated by normalizing flows trained to maximize the ELBO.

Usage

```
shrinkTPR(
  formula,
  data,
  a = 0.5,
  c = 0.5,
  formula_mean,
  a_mean = 0.5,
  c_mean = 0.5,
  sigma2_rate = 10,
  nu_alpha = 0.5,
  nu_beta = 2,
  kernel_func = kernel_se,
  n_layers = 10,
  n_latent = 10,
  flow_func = sylvester,
  flow_args,
  n_epochs = 1000,
  auto_stop = TRUE,
  cont_model,
  device,
  display_progress = TRUE,
  optim_control
)
```

Arguments

formula	object of class "formula": a symbolic representation of the model for the covariance equation, as in lm. The response variable and covariates are specified here.
data	*optional* data frame containing the response variable and the covariates. If not found in data, the variables are taken from environment(formula). No NAs are allowed in the response variable or covariates.
a	positive real number controlling the behavior at the origin of the shrinkage prior for the covariance structure. The default is 0.5.
c	positive real number controlling the tail behavior of the shrinkage prior for the covariance structure. The default is 0.5.
formula_mean	*optional* formula for the linear mean equation. If provided, the covariates for the mean structure are specified separately from the covariance structure. A response variable is not required in this formula.

a_mean	positive real number controlling the behavior at the origin of the shrinkage for the mean structure. The default is 0.5.
c_mean	positive real number controlling the tail behavior of the shrinkage prior for the mean structure. The default is 0.5.
sigma2_rate	positive real number controlling the prior rate parameter for the residual variance. The default is 10.
nu_alpha	positive real number controlling the shape parameter of the shifted gamma prior for the degrees of freedom of the Student-t process. The default is 0.5.
nu_beta	positive real number controlling the rate parameter of the shifted gamma prior for the degrees of freedom of the Student-t process. The default is 2.
kernel_func	function specifying the covariance kernel. The default is kernel_se, a squared exponential kernel. For guidance on how to provide a custom kernel function, see Details.
n_layers	positive integer specifying the number of flow layers in the normalizing flow. The default is 10.
n_latent	positive integer specifying the dimensionality of the latent space for the normalizing flow. The default is 10.
flow_func	function specifying the normalizing flow transformation. The default is sylvester. For guidance on how to provide a custom flow function, see Details.
flow_args	*optional* named list containing arguments for the flow function. If not provided, default arguments are used. For guidance on how to provide a custom flow function, see Details.
n_epochs	positive integer specifying the number of training epochs. The default is 1000.
auto_stop	logical value indicating whether to enable early stopping based on convergence. The default is TRUE.
cont_model	*optional* object returned from a previous shrinkTPR call, enabling continuation of training from the saved state.
device	*optional* device to run the model on, e.g., torch_device("cuda") for GPU or torch_device("cpu") for CPU. Defaults to GPU if available; otherwise, CPU.
display_progress	logical value indicating whether to display progress bars and messages during training. The default is TRUE.
optim_control	*optional* named list containing optimizer parameters. If not provided, default settings are used.

Details

Model Specification

f is a Student-t process if any finite collection of function values has a joint multivariate Student-t distribution. Given N observations with d-dimensional covariates x_i, the joint density is thus

$$(f(x_1), \ldots, f(x_N)) \sim t_N(\nu,\, \mu(x_1, \ldots, x_N),\, K(\theta, \tau))\,,$$

. which means that f follows $\mathcal{TP}(\nu, \mu, k(\cdot, \cdot\,; \theta, \tau))$ Student-t process with ν degrees of freedom, mean function μ, and covariance kernel k. As opposed to a Gaussian process regression

model, the noise is added directly to the kernel, so the likelihood for the observations is $Y \sim t_N\big(\nu,\, \mu(x_1, \ldots, x_N),\, K(\theta, \tau) + \sigma^2 I\big)$. The default squared exponential kernel is

$$k(x, x';\, \theta, \tau) = \frac{1}{\tau} \exp\left(-\frac{1}{2} \sum_{j=1}^{d} \theta_j (x_j - x'_j)^2 \right),$$

where $\theta_j \geq 0$ are inverse squared length-scales and $\tau > 0$ is the output scale. Users can specify custom kernels by following the guidelines below, or use one of the other provided kernel functions in `kernel_functions`.

If `formula_mean` is provided, the process mean becomes $x_{\mu,i}^{\top}\beta$.

Priors

$$\theta_j \mid \tau \sim \mathrm{TG}(a, c, \tau), \quad j = 1, \ldots, d,$$

$$\tau \sim F(2c, 2a),$$

$$\sigma^2 \sim \mathrm{Exp}(\sigma^2_{\mathrm{rate}}),$$

$$\nu - 2 \sim \mathrm{Gamma}(\nu_\alpha, \nu_\beta).$$

The shift by 2 ensures $\nu > 2$ so that the process variance is finite. With a mean structure, $\beta_k \mid \tau_\mu \sim \mathrm{NGG}(a_\mu, c_\mu, \tau_\mu)$ and $\tau_\mu \sim F(2c_\mu, 2a_\mu)$.

Inference

The posterior is approximated by a normalizing flow q_ϕ trained to maximize the ELBO. `auto_stop` triggers early stopping when the ELBO shows no significant improvement over the last 100 iterations.

Custom Kernel Functions

Users can define custom kernel functions by passing them to the `kernel_func` argument. A valid kernel function must follow the same structure as `kernel_se`. The function must:

1. Accept arguments `thetas` (`n_latent` x `d`), `tau` (length `n_latent`), `x` (`N` x `d`), and optionally `x_star` (`N_new` x `d`).
2. Return a `torch_tensor` of dimensions `n_latent` x `N` x `N` (if `x_star` = `NULL`) or `n_latent` x `N_new` x `N` (if `x_star` is provided).
3. Produce a valid positive semi-definite covariance matrix using `torch` tensor operations.

See `kernel_functions` for documented examples.

Custom Flow Functions

Users can define custom flow functions by implementing an `nn_module` in `torch`. The module must have a `forward` method that accepts a tensor `z` of shape `n_latent` x `D` and returns a list with:

- `zk`: the transformed samples, shape `n_latent` x `D`.
- `log_diag_j`: log-absolute-determinant of the Jacobian, shape `n_latent`.

See `sylvester` for a documented example.

Value

A list object of class `shrinkTPR`, containing:

`model`	The best-performing trained model.
`loss`	The best loss value (ELBO) achieved during training.
`loss_stor`	A numeric vector storing the ELBO values at each training iteration.
`last_model`	The model state at the final iteration.
`optimizer`	The optimizer object used during training.
`model_internals`	Internal objects required for predictions and further training, such as model matrices and formulas.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

References

Shah, A., Wilson, A., & Ghahramani, Z. (2014, April). Student-t processes as alternatives to Gaussian processes. In Artificial intelligence and statistics (pp. 877-885). PMLR.

Examples

```
if (torch::torch_is_installed()) {
  # Simulate data
  set.seed(123)
  torch::torch_manual_seed(123)
  n <- 100
  x <- matrix(runif(n * 2), n, 2)
  y <- sin(2 * pi * x[, 1]) + rnorm(n, sd = 0.1)
  data <- data.frame(y = y, x1 = x[, 1], x2 = x[, 2])

  # Fit TPR model
  res <- shrinkTPR(y ~ x1 + x2, data = data)

  # Check convergence
  plot(res$loss_stor, type = "l", main = "Loss Over Iterations")

  # Check posterior
  samps <- gen_posterior_samples(res, nsamp = 1000)
  boxplot(samps$thetas) # Second theta is pulled towards zero

  # Predict
  x1_new <- seq(from = 0, to = 1, length.out = 100)
  x2_new <- runif(100)
  y_new <- predict(res, newdata = data.frame(x1 = x1_new, x2 = x2_new), nsamp = 2000)

  # Plot
  quants <- apply(y_new, 2, quantile, c(0.025, 0.5, 0.975))
  plot(x1_new, quants[2, ], type = "l", ylim = c(-1.5, 1.5),
```

```
      xlab = "x1", ylab = "y", lwd = 2)
polygon(c(x1_new, rev(x1_new)), c(quants[1, ], rev(quants[3, ])),
      col = adjustcolor("skyblue", alpha.f = 0.5), border = NA)
points(x[,1], y)
curve(sin(2 * pi * x), add = TRUE, col = "forestgreen", lwd = 2, lty = 2)



# Add mean equation
res2 <- shrinkTPR(y ~ x1 + x2, formula_mean = ~ x1, data = data)
}
```

---

simGPR *Simulate Data for Gaussian Process Regression*

Description

simGPR generates simulated data for Gaussian Process Regression (GPR) models, including the true hyperparameters used for simulation.

Usage

```
simGPR(
  N = 200,
  d = 3,
  d_mean = 0,
  sigma2 = 0.1,
  tau = 2,
  kernel_func = kernel_se,
  perc_spars = 0.5,
  rho = 0,
  theta,
  beta,
  device
)
```

Arguments

N	Positive integer specifying the number of observations to simulate. Default is 200.
d	Positive integer specifying the number of covariates for the covariance structure. Default is 3.
d_mean	Positive integer specifying the number of covariates for the mean structure. Default is 0.
sigma2	Positive numeric value specifying the noise variance. Default is 0.1.
tau	Positive numeric value specifying the global shrinkage parameter. Default is 2.

kernel_func	Function specifying the covariance kernel. Default is kernel_se.
perc_spars	Numeric value in [0, 1] indicating the proportion of elements in theta and beta to sparsify. Default is 0.5.
rho	Numeric value in [0, 1] indicating the correlation between the covariates. Default is 0.
theta	*Optional* numeric vector specifying the true inverse length-scale parameters. If not provided, they are randomly generated.
beta	*Optional* numeric vector specifying the true regression coefficients for the mean structure. If not provided, they are randomly generated.
device	*Optional* torch_device object specifying whether to run the simulation on CPU or GPU. Defaults to GPU if available.

Details

This function simulates data from a Gaussian Process Regression model. The response variable y is sampled from a multivariate normal distribution with a covariance matrix determined by the specified kernel function, theta, tau, and sigma2. If d_mean > 0, a mean structure is included in the simulation, with covariates x_mean and regression coefficients beta.

Value

A list containing:

- data: A data frame with y (response variable), x (covariates for the covariance structure), and optionally x_mean (covariates for the mean structure).
- true_vals: A list containing the true values used for the simulation:
  - theta: The true inverse length-scale parameters.
  - sigma2: The true noise variance.
  - tau: The true global shrinkage parameter.
  - beta (optional): The true regression coefficients for the mean structure.

Examples

```
if (torch::torch_is_installed()) {
  torch::torch_manual_seed(123)

  # Simulate data with default settings
  sim_data <- simGPR()

  # Simulate data with custom settings
  sim_data <- simGPR(N = 100, d = 5, d_mean = 2, perc_spars = 0.3, sigma2 = 0.5)

  # Access the simulated data
  head(sim_data$data)

  # Access the true values used for simulation
  sim_data$true_vals
  }
```

simMVGPR *Simulate Data for Multivariate Gaussian Process Regression*

Description

simMVGPR generates simulated data for Multivariate Gaussian Process Regression (MVGPR) models, including the true hyperparameters used for simulation.

Usage

```
simMVGPR(
  N = 200,
  M = 2,
  d = 3,
  sigma2 = 0.1,
  tau = 2,
  kernel_func = kernel_se,
  perc_spars = 0.5,
  rho = 0,
  theta,
  Omega,
  device
)
```

Arguments

N	Positive integer specifying the number of observations to simulate. Default is 200.
M	positive integer specifying the number of response variables. Default is 2.
d	Positive integer specifying the number of covariates for the covariance structure. Default is 3.
sigma2	Positive numeric value specifying the noise variance. Default is 0.1.
tau	Positive numeric value specifying the global shrinkage parameter. Default is 2.
kernel_func	Function specifying the covariance kernel. Default is kernel_se.
perc_spars	Numeric value in [0, 1] indicating the proportion of inactive (zero) inverse length-scale parameters in theta. Default is 0.5.
rho	Numeric value in [0, 1] indicating the correlation between the covariates. Default is 0.
theta	*Optional* numeric vector specifying the true inverse length-scale parameters. If not provided, they are randomly generated.
Omega	*Optional* positive definite matrix of size M x M. If not provided, it is generated as S x D x S, where D is a correlation matrix generated by the LKJ distribution with shape parameter 1, and S is a diagonal matrix with entries sampled from a gamma(1, 1) distribution.
device	*Optional* torch_device object specifying whether to run the simulation on CPU or GPU. Defaults to GPU if available.

Details

This function simulates data from a multivariate Gaussian process regression model. The response variable y is sampled from a matrix normal distribution with a covariance matrix determined by the specified kernel function, theta, tau, the correlation matrix Omega and sigma2 in the following way:

$$\boldsymbol{Y} \sim \mathcal{MN}_{N,M}(\boldsymbol{0}, \boldsymbol{K}(\boldsymbol{x}; \boldsymbol{\theta}, \tau) + \boldsymbol{I}\sigma^2, \boldsymbol{\Omega})$$

which is equivalent to

$$vec(\boldsymbol{Y}) \sim \mathcal{N}_{NM}(\boldsymbol{0}, \boldsymbol{\Omega} \otimes (\boldsymbol{K}(\boldsymbol{x}; \boldsymbol{\theta}, \tau) + \boldsymbol{I}\sigma^2))$$

.

Value

A list containing:

- data: A data frame with M + d columns y.1, ..., y.M (response variables) and x.1, ..., x.d (covariates for the covariance structure).
- true_vals: A list containing the true values used for the simulation:
  - theta: The true inverse length-scale parameters.
  - sigma2: The true noise variance.
  - tau: The true global shrinkage parameter.

Examples

```
if (torch::torch_is_installed()) {
  torch::torch_manual_seed(123)

  # Simulate data with default settings
  sim_data <- simMVGPR()

  # Simulate data with custom settings
  sim_data <- simMVGPR(N = 100, d = 5, perc_spars = 0.3, sigma2 = 0.5)

  # Access the simulated data
  head(sim_data$data)

  # Access the true values used for simulation
  sim_data$true_vals
  }
```

`sylvester` *Sylvester Normalizing Flow*

Description

The `sylvester` function implements Sylvester normalizing flows as described by van den Berg et al. (2018) in "Sylvester Normalizing Flows for Variational Inference." This flow applies a sequence of invertible transformations to map a simple base distribution to a more complex target distribution, allowing for flexible posterior approximations in Gaussian process regression models.

Usage

```
sylvester(d, n_householder)
```

Arguments

`d`	An integer specifying the latent dimensionality of the input space.
`n_householder`	An optional integer specifying the number of Householder reflections used to orthogonalize the transformation. Defaults to `min(5, d - 1)`.

Details

The Sylvester flow uses two triangular matrices (`R1` and `R2`) and Householder reflections to construct invertible transformations. The transformation is parameterized as follows:

$$z = QR_1h(Q^TR_2zk + b) + zk,$$

where:

- `Q` is an orthogonal matrix obtained via Householder reflections.
- `R1` and `R2` are upper triangular matrices with learned diagonal elements.
- `h` is a non-linear activation function (default: `torch_tanh`).
- `b` is a learned bias vector.

The log determinant of the Jacobian is computed to ensure the invertibility of the transformation and is given by:

$$\log|detJ| = \sum_{i=1}^{d} \log|diag_1[i] \cdot diag_2[i] \cdot h'(RQ^Tzk + b) + 1|,$$

where `diag_1` and `diag_2` are the learned diagonal elements of `R1` and `R2`, respectively, and `h\'` is the derivative of the activation function.

Value

An `nn_module` object representing the Sylvester normalizing flow. The module has the following key components:

- `forward(zk)`: The forward pass computes the transformed variable `z` and the log determinant of the Jacobian.
- Internal parameters include matrices `R1` and `R2`, diagonal elements, and Householder reflections used for orthogonalization.

References

van den Berg, R., Hasenclever, L., Tomczak, J. M., & Welling, M. (2018). "Sylvester Normalizing Flows for Variational Inference." *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI 2018)*.

Examples

```
if (torch::torch_is_installed()) {
  # Example: Initialize a Sylvester flow
  d <- 5
  n_householder <- 4
  flow <- sylvester(d, n_householder)

  # Forward pass through the flow
  zk <- torch::torch_randn(10, d)  # Batch of 10 samples
  result <- flow(zk)

  print(result$zk)
  print(result$log_diag_j)
}
```

Index