Package 'deepICMGP'

November 3, 2026

Title Deep Intrinsic Coregionalization Gaussian Processes for Multiple Outputs

Version 0.1.1

Author Chun-Yi Chang [aut, cre],
Chih-Li Sung [aut],
Annie S. Booth [ctb] (Author of deepgp code adapted in this package)

Maintainer Chun-Yi Chang <chang103@msu.edu>

Description Fits two-layer deep intrinsic coregionalization Gaussian process surrogate models for multiple outputs observed at shared input locations. Uses Markov chain Monte Carlo sampling for kernel parameters and latent coordinates, with plug-in coregionalization estimates. Provides burn-in removal and thinning, posterior predictive means and within-location output covariances, and sequential experimental design using an active learning criterion. The method is described by Chang and Sung (2026) <doi:10.1615/Int.J.UncertaintyQuantification.2026062570>.
Adapts code from the 'deepgp' package, with underlying deep Gaussian process methodology described by Sauer, Gramacy, and Higdon (2023) <doi:10.1080/00401706.2021.2008505>.

License LGPL

Encoding UTF-8

Imports lhs, parallel, stats

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-11-03 15:30:02 UTC

Contents

ALC_deepICMGP 2
fit_deepICMGP 4
pred_deepICMGP 8
trim_deepICMGP 11

Index 13

`ALC_deepICMGP` *Active Learning Cohn for deepICMGP*

Description

Select the next design point by minimizing the Active Learning Cohn (ALC) criterion implemented in deepICMGP over a set of candidate inputs. The criterion is averaged over reference inputs and retained MCMC samples.

Usage

```
ALC_deepICMGP(object, Xcand = NULL, Xref = NULL)
```

Arguments

`object`	An object of class `deepICMGP` returned by `fit_deepICMGP` or `trim_deepICMGP`, containing at least two retained MCMC samples.
`Xcand`	Numeric matrix of candidate inputs, with one row per candidate and the same column order and original units as the training inputs. A vector is accepted for one input dimension. Default is `NULL`, which generates 100 candidates using `maximinLHS(100, d)`, where `d` is the number of input dimensions, and maps them to the observed training input ranges.
`Xref`	Numeric matrix of reference inputs over which the criterion is averaged. Use the same column order and original units as the training inputs. A vector is accepted for one input dimension. Default is `NULL`, which uses `Xcand`. The number of reference inputs can be different from the number of candidates.

Details

To select the next design point that provides information about all outputs jointly, ALC uses a determinant-based measure of the predictive uncertainty remaining after adding a candidate. This measure is averaged over reference inputs and retained MCMC samples. The candidate with the smallest value is selected.

The returned `x_next` is the row of `Xcand` corresponding to the lowest ALC value. Candidate inputs and the selected point are returned in their original units.

Value

A list containing:

- `value`: Numeric vector of ALC values at the candidate points, in the same row order as `Xcand`.
- `Xcand`: Matrix of candidate inputs supplied by the user or generated by the function, in original input units.
- `x_next`: One-row matrix containing the candidate with the smallest ALC value, in original input units.
- `time`: Computation time in seconds.

Code provenance

This implementation adapts code from the **deepgp** package and extends it to deep intrinsic coregionalization models. See Sauer, Gramacy, and Higdon (2023) for the underlying deep GP methodology, and https://CRAN.R-project.org/package=deepgp for the source package.

References

Chang, C.-Y. and Sung, C.-L. (2026). Deep Intrinsic Coregionalization Multi-Output Gaussian Process Surrogate with Active Learning. International Journal for Uncertainty Quantification, 16(4), 29–52. doi:10.1615/Int.J.UncertaintyQuantification.2026062570.

Sauer, A., Gramacy, R. B., and Higdon, D. (2023). Active Learning for Deep Gaussian Process Surrogates. Technometrics, 65(1), 4–18. doi:10.1080/00401706.2021.2008505.

See Also

pred_deepICMGP, trim_deepICMGP

Examples

```
library(lhs)

# Branin Function: 2d input with 3 outputs
f1 <- function(x) {
  x1 <- x[, 1] * 15 - 5
  x2 <- x[, 2] * 15
  10 * sqrt(
    (-1.275 * (1.2 * x1 + 0.4)^2 / pi^2 +
       5 * (1.2 * x1 + 0.4) / pi + (1.2 * x2 + 0.4) - 6)^2 +
      (10 - 5 / (4 * pi)) * cos(1.2 * x1 + 0.4) + 10
  ) + 2 * (1.2 * x1 + 1.9) - 3 * (3 * (1.2 * x2 + 2.4) - 1) -
    1 - 3 * x2 + 1
}
f2 <- function(x) {
  x1 <- x[, 1] * 15 - 5
  x2 <- x[, 2] * 15
  10 * sqrt(
    (-1.275 * (x1 + 2)^2 / pi^2 + 5 * (x1 + 2) / pi + (x2 + 2) - 6)^2 +
      (10 - 5 / (4 * pi)) * cos(x1 + 2) + 10
  ) + 2 * (x1 - 0.5) - 3 * (3 * x2 - 1) - 1
}
f3 <- function(x) {
  x1 <- x[, 1] * 15 - 5
  x2 <- x[, 2] * 15
  (-1.275 * x1^2 / pi^2 + 5 * x1 / pi + x2 - 6)^2 +
    (10 - 5 / (4 * pi)) * cos(x1) + 10
}
response <- function(x) cbind(f1(x), f2(x), f3(x))
set.seed(2025)
X <- maximinLHS(30, 2)
Y <- response(X)
```

```
Xcand <- maximinLHS(50, 2)

# Fit a deepICMGP model with 500 MCMC samples and set nugget to 1e-8 for deterministic case.
# For actual analysis, use a longer MCMC chain and assess convergence.
fit <- fit_deepICMGP(X, Y, nmcmc = 500, true_g = 1e-8)

# Compute ALC and select the next design point from Xcand
alc <- ALC_deepICMGP(fit, Xcand = Xcand)
alc$x_next


# Optionally, supply a separate reference grid.
Xref <- maximinLHS(200, 2)
alc <- ALC_deepICMGP(fit, Xcand = Xcand, Xref = Xref)
alc$x_next

# If candidates are not given, 100 candidate inputs will be generated automatically.
alc <- ALC_deepICMGP(fit)
nrow(alc$Xcand)
alc$x_next
```

`fit_deepICMGP` *Fit a two-layer multi-output Gaussian process*

Description

Fit a deep intrinsic coregionalization Gaussian process (deepICMGP) to multiple outputs observed at shared input locations. The sampler updates kernel parameters and a latent Gaussian process layer. The coregionalization matrices describe dependence among outputs and among latent coordinates.

Usage

```
fit_deepICMGP(
  x,
  y,
  nmcmc = 5000,
  D = max(ncol(x), ncol(y)),
  trace = TRUE,
  true_g = NULL,
  settings = NULL,
  eps = sqrt(.Machine$double.eps)
)
```

Arguments

`x`	Numeric input matrix with n rows and d columns. A numeric vector is converted to a one-column matrix. Each column is automatically scaled to $[0, 1]$ using its training minimum and range.

y	Numeric response matrix with n rows and q columns, in the same row order as x. Each column represents one output and is automatically centered and divided by its training sample standard deviation. For a single output, supply a one-column matrix, i.e., `matrix(y, ncol = 1)`.
nmcmc	Length of the MCMC chain. Must be an integer greater than 1. Default is `5000` MCMC runs.
D	Positive integer giving the number of latent variables. Defaults to `max(ncol(x), ncol(y))`.
trace	Logical. Print progress for every 500 iterations when `TRUE`.
true_g	A positive scalar specifying a fixed value for the outer-layer nugget g. Default is `NULL`, in which case g is sampled by MCMC with eps as its lower bound. A small positive value approximates a noise-free response model, e.g., `true_g = 1e-8`.
settings	List of proposal and prior settings. Default is `NULL`, which uses the defaults described in Details. One may provide alternative values for any settings for customizing.
eps	A small positive constant for numerical stability. The default is `sqrt(.Machine$double.eps)`, which is approximately `1e-8`.

Details

The model uses squared-exponential kernels with a scalar lengthscale parameter for each layer. The outer layer maps the D-dimensional latent values to the q outputs. The prior settings apply to the model after the inputs are scaled and each output is standardized to mean zero and standard deviation one.

Metropolis-Hastings updates are used for the lengthscale parameters and, when not fixed, the nugget. The latent layer is updated by elliptical slice sampling. The proposal for a positive parameter a is uniform on `[settings$l * a / settings$u, settings$u * a / settings$l]`. Defaults are `l = 1` and `u = 2`. Gamma prior densities are evaluated at the parameter minus eps, using shape and rate:

- `alpha$g = 1.5`, `beta$g = 3.9`.
- `alpha$theta_y = 1.5`, `beta$theta_y = 3.9 / 6`.
- `alpha$theta_w = 1.5`, `beta$theta_w = 3.9 / 4`.

where g is the nugget parameter, `theta_y` and `theta_w` are the lengthscale parameters for outer and inner-layer, respectively.

For example, `settings = list(l = 1, u = 1.5, beta = list(g = 5))` overrides only those entries. Coregionalization matrices are estimated during MCMC sampling, rather than sampled directly from their posterior distributions.

Initial latent values use the first `D` scaled input columns when `D <= ncol(x)`, or reuse the scaled input matrix otherwise. The inner-layer prior mean is zero, and the inner kernel scale is one. Input and output transformations are learned from training data only and retained for prediction and active learning.

One can inspect trace plots and convergence before using the samples. We suggest removing burn-in with `trim_deepICMGP` before prediction or active learning.

Value

A list of class `deepICMGP`. Let nt be the number of MCMC samples.

- `x`, `y`: the scaled input and standardized response matrices used by the sampler.
- `scaling`: training transformations `x_min`, `x_range`, `y_mean`, and `y_sd`.
- `nmcmc`, `settings`, `eps`: stored MCMC count and settings.
- `g`, `theta_y`, `theta_w`: length-nt vectors of nugget and lengthscale values.
- `B_y`: $q \times q \times nt$ array of outer-layer coregionalization estimates.
- `B_w`: $D \times D \times nt$ array of inner-layer coregionalization estimates.
- `w`: list of nt latent matrices, each $n \times D$.
- `ll`: length-nt vector of outer-layer log-likelihood values, up to omitted constants.
- `time`: elapsed computation time in seconds.

Code provenance

This implementation adapts code from the **deepgp** package and extends it to deep intrinsic coregionalization models. See Sauer, Gramacy, and Higdon (2023) for the underlying deep GP methodology, and https://CRAN.R-project.org/package=deepgp for the source package.

References

Chang, C.-Y. and Sung, C.-L. (2026). Deep Intrinsic Coregionalization Multi-Output Gaussian Process Surrogate with Active Learning. International Journal for Uncertainty Quantification, 16(4), 29–52. doi:10.1615/Int.J.UncertaintyQuantification.2026062570.

Sauer, A., Gramacy, R. B., and Higdon, D. (2023). Active Learning for Deep Gaussian Process Surrogates. Technometrics, 65(1), 4–18. doi:10.1080/00401706.2021.2008505.

See Also

`trim_deepICMGP`, `pred_deepICMGP`, `ALC_deepICMGP`

Examples

```
library(lhs)

# Forrester Function: 1d input with 2 outputs
f1 <- function(x) {
  (6 * x - 2)^2 * sin(12 * x - 4)
}
f2 <- function(x) {
  0.5 * f1(x) + 10 * (x - 0.5) + 5
}
response <- function(x) cbind(f1(x), f2(x))
set.seed(2025)
X <- maximinLHS(9, 1)
Y <- response(X)

# Fit a deepICMGP model with 2000 MCMC samples and set nugget to 1e-8 for deterministic case.
```

```
fit <- fit_deepICMGP(X, Y, nmcmc = 2000, true_g = 1e-8)

# Check the last coregionalization matrices for both layers.
fit$B_w[, , 2000]
fit$B_y[, , 2000]

# Trace plots for checking convergence of lengthscale parameters.
## Inner-layer lengthscale
ts.plot(fit$theta_w)

## Outer-layer lengthscale
ts.plot(fit$theta_y)


# Branin Function: 2d input with 3 outputs
f1 <- function(x) {
  x1 <- x[, 1] * 15 - 5
  x2 <- x[, 2] * 15
  10 * sqrt(
    (-1.275 * (1.2 * x1 + 0.4)^2 / pi^2 +
       5 * (1.2 * x1 + 0.4) / pi + (1.2 * x2 + 0.4) - 6)^2 +
      (10 - 5 / (4 * pi)) * cos(1.2 * x1 + 0.4) + 10
  ) + 2 * (1.2 * x1 + 1.9) - 3 * (3 * (1.2 * x2 + 2.4) - 1) -
    1 - 3 * x2 + 1
}
f2 <- function(x) {
  x1 <- x[, 1] * 15 - 5
  x2 <- x[, 2] * 15
  10 * sqrt(
    (-1.275 * (x1 + 2)^2 / pi^2 + 5 * (x1 + 2) / pi + (x2 + 2) - 6)^2 +
      (10 - 5 / (4 * pi)) * cos(x1 + 2) + 10
  ) + 2 * (x1 - 0.5) - 3 * (3 * x2 - 1) - 1
}
f3 <- function(x) {
  x1 <- x[, 1] * 15 - 5
  x2 <- x[, 2] * 15
  (-1.275 * x1^2 / pi^2 + 5 * x1 / pi + x2 - 6)^2 +
    (10 - 5 / (4 * pi)) * cos(x1) + 10
}
response <- function(x) cbind(f1(x), f2(x), f3(x))
set.seed(2025)
X <- maximinLHS(30, 2)
Y <- response(X)

# Fit a deepICMGP model with 5000 MCMC samples and set nugget to 1e-8 for deterministic case.
fit <- fit_deepICMGP(X, Y, nmcmc = 5000, true_g = 1e-8)

# Trace plots for checking convergence of lengthscale parameters.
## Inner-layer lengthscale
ts.plot(fit$theta_w)

## Outer-layer lengthscale
ts.plot(fit$theta_y)
```

pred_deepICMGP *Predict output means, variances, and within-location covariances*

Description

Compute posterior predictive mean, variance and within-location output covariances at new inputs using all retained MCMC samples. Parallel computation for prediction is supported with the **parallel** package.

Usage

```
pred_deepICMGP(
  object,
  x_new,
  store_latent = FALSE,
  return_all = FALSE,
  cores = 1
)
```

Arguments

object	An object of class deepICMGP returned by fit_deepICMGP or trim_deepICMGP, containing at least two retained MCMC samples.
x_new	Numeric matrix with n_new = nrow(x_new) rows and the d columns, using the same column order and original units as the training inputs. For one input dimension a numeric vector is accepted. For one location in multiple dimensions, please use a one-row matrix.
store_latent	Logical. Include latent predictions and kernel quadratic terms in the returned prediction object. Default is FALSE.
return_all	Logical. Return the pointwise variance matrix for each retained sample in s2_all. Default is FALSE.
cores	Number of CPU cores for parallel prediction using the **parallel** package. Must be a positive integer. Default is 1.

Details

The function samples latent values at the new inputs, then combines conditional moments of outer-layer using the laws of total expectation and covariance.

C2[i, ,] describes dependence among outputs at location i. s2_all stores conditional variances for individual MCMC samples. s2 combines their average with the variation in predicted means across samples to obtain the posterior predictive variance.

Value

The supplied `deepICMGP` object with the following fields provided. Let nt be the number of retained samples and q the number of outputs. The number of prediction locations is `n_new = nrow(x_new)`.

- `x_new`: the $n_{\text{new}} \times d$ prediction matrix.
- `mean`: $n_{\text{new}} \times q$ matrix of posterior predictive means.
- `s2`: $n_{\text{new}} \times q$ matrix of posterior predictive variances.
- `C2`: $n_{\text{new}} \times q \times q$ array of within-location output covariances. `C2[i, , ]` is the covariance matrix at row `i`.
- `s2_all`: when `return_all = TRUE`, a list of nt variance matrices, each $n_{\text{new}} \times q$.
- `w_new`: when `store_latent = TRUE`, a list of nt sampled latent matrices, each $n_{\text{new}} \times D$.
- `quad`: when `store_latent = TRUE`, an $n_{\text{new}} \times nt$ matrix of outer-kernel quadratic terms used for active learning.
- `time`: Total elapsed computation time in seconds, including the time stored in `object` and the current prediction call.

Code provenance

This implementation adapts code from the **deepgp** package and extends it to deep intrinsic coregionalization models. See Sauer, Gramacy, and Higdon (2023) for the underlying deep GP methodology, and https://CRAN.R-project.org/package=deepgp for the source package.

References

Chang, C.-Y. and Sung, C.-L. (2026). Deep Intrinsic Coregionalization Multi-Output Gaussian Process Surrogate with Active Learning. International Journal for Uncertainty Quantification, 16(4), 29–52. doi:10.1615/Int.J.UncertaintyQuantification.2026062570.

Sauer, A., Gramacy, R. B., and Higdon, D. (2023). Active Learning for Deep Gaussian Process Surrogates. Technometrics, 65(1), 4–18. doi:10.1080/00401706.2021.2008505.

See Also

`fit_deepICMGP`, `trim_deepICMGP`, `ALC_deepICMGP`

Examples

```
library(lhs)

# Forrester Function: 1d input with 2 outputs
f1 <- function(x) {
  (6 * x - 2)^2 * sin(12 * x - 4)
}
f2 <- function(x) {
  0.5 * f1(x) + 10 * (x - 0.5) + 5
}
response <- function(x) cbind(f1(x), f2(x))
set.seed(2025)
X <- maximinLHS(9, 1)
```

```
Y <- response(X)

Xtest <- maximinLHS(50, 1)
Ytrue <- response(Xtest)

# Fit a deepICMGP model with 2000 MCMC samples and set nugget to 1e-8
# for deterministic case.
fit <- fit_deepICMGP(X, Y, nmcmc = 2000, true_g = 1e-8)

# Trim a fitted deepICMGP model by removing first 1500 MCMC samples and
# retaining every second sample.
trim <- trim_deepICMGP(fit, burn = 1500, thin = 2)

# Predict a trimmed deepICMGP model on Xtest.
pred <- pred_deepICMGP(trim, x_new = Xtest)

# Calculate RMSE for each output.
RMSE <- numeric(ncol(Y))
for (j in seq_len(ncol(Y))) {
  RMSE[j] <- sqrt(mean((pred$mean[, j] - Ytrue[, j])^2))
}
RMSE


# Branin Function: 2d input with 3 outputs
f1 <- function(x) {
  x1 <- x[, 1] * 15 - 5
  x2 <- x[, 2] * 15
  10 * sqrt(
    (-1.275 * (1.2 * x1 + 0.4)^2 / pi^2 +
       5 * (1.2 * x1 + 0.4) / pi + (1.2 * x2 + 0.4) - 6)^2 +
      (10 - 5 / (4 * pi)) * cos(1.2 * x1 + 0.4) + 10
  ) + 2 * (1.2 * x1 + 1.9) - 3 * (3 * (1.2 * x2 + 2.4) - 1) -
    1 - 3 * x2 + 1
}
f2 <- function(x) {
  x1 <- x[, 1] * 15 - 5
  x2 <- x[, 2] * 15
  10 * sqrt(
    (-1.275 * (x1 + 2)^2 / pi^2 + 5 * (x1 + 2) / pi + (x2 + 2) - 6)^2 +
      (10 - 5 / (4 * pi)) * cos(x1 + 2) + 10
  ) + 2 * (x1 - 0.5) - 3 * (3 * x2 - 1) - 1
}
f3 <- function(x) {
  x1 <- x[, 1] * 15 - 5
  x2 <- x[, 2] * 15
  (-1.275 * x1^2 / pi^2 + 5 * x1 / pi + x2 - 6)^2 +
    (10 - 5 / (4 * pi)) * cos(x1) + 10
}
response <- function(x) cbind(f1(x), f2(x), f3(x))
set.seed(2025)
X <- maximinLHS(30, 2)
Y <- response(X)
```

```
Xtest <- maximinLHS(100, 2)

# Fit a deepICMGP model with 5000 MCMC samples and set nugget to 1e-8
# for deterministic case.
fit <- fit_deepICMGP(X, Y, nmcmc = 5000, true_g = 1e-8)

# Predict a fitted deepICMGP model on Xtest and return sampled latent values.
pred <- pred_deepICMGP(fit, Xtest, store_latent = TRUE)
length(pred$w_new)

# For parallel prediction:
pred <- pred_deepICMGP(fit, Xtest, cores = 2)
```

`trim_deepICMGP` *Remove burn-in and thin stored MCMC samples*

Description

Retain selected samples from a fitted deepICMGP object before prediction or active learning.

Usage

```
trim_deepICMGP(object, burn, thin = 1)
```

Arguments

`object`	An object of class `deepICMGP` returned by `fit_deepICMGP` or `trim_deepICMGP`.
`burn`	Nonnegative integer smaller than `object$nmcmc`.
`thin`	Positive integer thinning interval. Retains original sample indices divisible by `thin`. When `thin = 1`, it retains every sample after burn-in.

Details

The number of stored samples nt must be greater than `burn`. After discarding burn-in, keep samples at the thinning interval. When both nt and `burn` are multiples of `thin`, the retained count is `(nt - burn) / thin`. For example, 1000 samples with `burn = 500` and `thin = 2` leave 250 samples. The actual retained count is stored in `nmcmc`.

Retain at least two samples for prediction, which estimates variance across samples. Invalid burn-in/thinning values and choices that retain no samples are rejected.

Value

An object of class `deepICMGP` with `g`, `theta_y`, `theta_w`, `B_y`, `B_w`, `w`, and `ll` subset to the retained samples, and `nmcmc` updated accordingly. Elapsed trimming time is added to `time`. Training data and settings are unchanged.

References

Chang, C.-Y. and Sung, C.-L. (2026). Deep Intrinsic Coregionalization Multi-Output Gaussian Process Surrogate with Active Learning. International Journal for Uncertainty Quantification, 16(4), 29–52. doi:10.1615/Int.J.UncertaintyQuantification.2026062570.

Sauer, A., Gramacy, R. B., and Higdon, D. (2023). Active Learning for Deep Gaussian Process Surrogates. Technometrics, 65(1), 4–18. doi:10.1080/00401706.2021.2008505.

See Also

fit_deepICMGP, pred_deepICMGP

Examples

```
library(lhs)

# Forrester Function: 1d input with 2 outputs
f1 <- function(x) {
  (6 * x - 2)^2 * sin(12 * x - 4)
}
f2 <- function(x) {
  0.5 * f1(x) + 10 * (x - 0.5) + 5
}
response <- function(x) cbind(f1(x), f2(x))
set.seed(2025)
X <- maximinLHS(9, 1)
Y <- response(X)

# Fit a deepICMGP model with 2000 MCMC samples and set nugget to 1e-8
# for deterministic case.
fit <- fit_deepICMGP(X, Y, nmcmc = 2000, true_g = 1e-8)

# Trim a fitted deepICMGP model by removing first 1500 MCMC samples and
# retaining every second sample.
trim <- trim_deepICMGP(fit, burn = 1500, thin = 2)

# Check the number of retained MCMC samples.
trim$nmcmc
```

Index

ALC_deepICMGP, 2, *6*, *9*

fit_deepICMGP, 4, *9*, *12*

pred_deepICMGP, *3*, *6*, 8, *12*

trim_deepICMGP, *3*, *6*, *9*, 11