Package: GEMSS (via r-universe)

October 30, 2026

Title Generalization Error Minimization in SubSampling for Gaussian Processes

Version 0.1.1

Description Implements the GEMSS algorithm for sequential subdata selection in large-scale Gaussian process modeling (Chang, Hua, and Wu, 2026) <doi:10.1080/00401706.2026.2670596>. The method selects data points by a criterion consisting of predictive and space-filling parts, enabling efficient surrogate modeling for massive datasets.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports Rcpp (>= 1.0.0), hetGP, twinning

Suggests ContourFunctions,

LinkingTo Rcpp, RcppArmadillo

Config/roxygen2/version 8.0.0

Repository https://szhua-stat.r-universe.dev

Date/Publication 2026-07-01 14:49:08 UTC

RemoteUrl https://github.com/szhua-stat/gemss

RemoteRef HEAD

RemoteSha 9531810e6999fe47f817701ad1622c126f18eea5

Contents

compute_kernel 2
gemss_remove 3
gemss_select 5
gp_predict 9

Index **10**

`compute_kernel` *Compute the Correlation Matrix for Specified Kernels*

Description

Calculates the correlation matrix between two sets of locations using stationary kernels. This function supports Gaussian, Matern 5/2, and Matern 3/2 correlation structures in a multivariate product form.

Usage

```
compute_kernel(X1, X2 = NULL, theta, type)
```

Arguments

`X1`	a matrix of input locations, where each row represents an observation.
`X2`	a matrix of design locations. If `NULL`, the correlation is calculated between `X1` and itself.
`theta`	a `p` x `1` numeric vector of lengthscale parameters corresponding to each dimension.
`type`	a character string specifying the kernel type. Must be one of `"Gaussian"`, `"Matern5_2"`, or `"Matern3_2"`.

Details

For multivariate inputs, the correlation function is defined as the product of univariate kernels across all p dimensions:

$$C(\mathbf{x}, \mathbf{y}, \theta) = \prod_{k=1}^{p} c_k(|x_k - y_k|, \theta_k)$$

The available univariate correlation functions $c(d)$ (where $d = |x - y|$) are:

- `"Gaussian"`:
$$c(d, \theta) = \exp\left(-\frac{d^2}{\theta}\right)$$
- `"Matern5_2"`:
$$c(d, \theta) = \left(1 + \frac{\sqrt{5}d}{\theta} + \frac{5d^2}{3\theta^2}\right) \exp\left(-\frac{\sqrt{5}d}{\theta}\right)$$
- `"Matern3_2"`:
$$c(d, \theta) = \left(1 + \frac{\sqrt{3}d}{\theta}\right) \exp\left(-\frac{\sqrt{3}d}{\theta}\right)$$

Value

a numeric matrix of dimensions `nrow(X1)` x `nrow(X2)`.

gemss_remove *Redundant Data Removal via GEMSS*

Description

Implements a backward elimination process to sequentially remove redundant data points from a Gaussian Process model using the GEMSS criterion.

Usage

```
gemss_remove(
  X,
  Y,
  X_val,
  Y_val,
  n_remove,
  covtype,
  c1 = NULL,
  threshold = 0.01,
  verbose = TRUE
)
```

Arguments

X	an n x p matrix of input design locations.
Y	an n x 1 numeric vector of observed responses.
X_val, Y_val	validation data used to evaluate predictive performance. X_val must have the same number of columns as X.
n_remove	an integer specifying the maximum number of data points to sequentially remove.
covtype	a character string specifying the covariance kernel type. Must be one of "Gaussian", "Matern5_2", or "Matern3_2".
c1	a numeric value between 0 and 1 representing the weight of the first term in the GEMSS criterion ($c_2 = 1 - c_1$). The default (NULL) uses the adaptive weighting proposed in the paper. A smaller c1 places greater emphasis on the space-filling properties of the reduced subset.
threshold	a numeric value specifying the minimum required value of predictive R-squared to justify removing points. Default is 0.01.
verbose	a logical value; if TRUE, progress is printed at each iteration.

Details

In each iteration, the leave-one-out GEMSS criterion is computed for every point in the training set. The point that yields the smallest criterion value is removed. The predictive R-squared is then calculated on the validation set to evaluate the performance of the reduced dataset.

If there exists a reduced dataset with a predictive R-squared value greater than the specified `threshold`, the algorithm reports the subset that maximizes the R-squared. Otherwise, no removal is suggested.

Value

a list containing:

- `index`: an integer vector of the indices of the retained training data points.
- `remove`: an integer vector of the indices of the removed data points.
- `eval_matrix`: a matrix detailing the removal sequence, including the step `j`, the `removed_id`, and the resulting out-of-sample `R2_pred`.

References

Chang, M. C., Hua, S. Z., & Wu, C. F. J. (2026). GEMSS-Driven Subsampling for Information Extraction and Redundancy Elimination. *Technometrics*, 1–20. doi:10.1080/00401706.2026.2670596

Examples

```
# Generate 1D data with intentionally clustered (redundant) points
set.seed(123)
X_reg <- seq(0, 1, length.out = 20)
X_cluster <- rnorm(10, mean = 0.5, sd = 0.01) # Redundant points tightly packed around x=0.5
X <- as.matrix(c(X_reg, X_cluster))
Y <- sin(2 * pi * X) + rnorm(30, sd = 0.05)

# Generate validation data
X_val <- as.matrix(seq(0.05, 0.95, length.out = 20))
Y_val <- sin(2 * pi * X_val) + rnorm(20, sd = 0.05)

# Run GEMSS removal
res <- gemss_remove(X, Y, X_val, Y_val, n_remove = 10, covtype = "Matern3_2", verbose = TRUE)

# View the indices of the removed points
print(res$remove)
print(res$eval_matrix)

# Plot the removal data points (red)
plot(X, Y, main = "GEMSS Data Removal")
points(X[res$remove], Y[res$remove], pch = 19, col = 2)

# Compare GP predictions before and after removal
x_seq <- as.matrix(seq(0, 1, 0.02))
gp_bf <- hetGP::mleHomGP(X, Y, covtype = "Matern3_2")
lines(x_seq, predict(gp_bf, x_seq)$mean, col = 1, lty = 2)
gp_af <- hetGP::mleHomGP(X[-res$remove, , drop = FALSE], Y[-res$remove], covtype = "Matern3_2")
lines(x_seq, predict(gp_af, x_seq)$mean, col = 3)
```

`gemss_select` *Subdata Selection via GEMSS*

Description

Implements the Generalized Error-Minimizing Subsampling (GEMSS) algorithm to select subdata for Gaussian Process models.

Usage

```
gemss_select(
  X,
  Y,
  ns,
  covtype,
  parameters = NULL,
  X_val = NULL,
  Y_val = NULL,
  c1 = NULL,
  n_srs = NULL,
  n_top = NULL,
  verbose = TRUE
)
```

Arguments

`X`	an `n` x `p` matrix of input design locations.
`Y`	an `n` x `1` numeric vector of observed responses.
`ns`	target size of the final subdata.
`covtype`	covariance kernel type, either "Gaussian", "Matern5_2", or "Matern3_2".
`parameters`	a list of hyperparameters: `beta0` (mean), `theta` (length-scales), and `g` (nugget). If not provided, these are estimated via `mleHomGP` on a pilot sample generated by `twin`.
`X_val`, `Y_val`	optional validation data for calculating the predictive R-squared. If omitted, a random sample of size `min(0.1*n, 5000)` is used.
`c1`	value between 0 and 1 of the first term in GEMSS criterion ($c_2 = 1 - c_1$). The default (`NULL`) uses the adaptive weighting proposed in the paper.
`n_srs`, `n_top`	optional values to determine the candidate set for each iteration: • `n_srs`: number of random data points (default is `ns/2`). • `n_top`: number of top-ranked points from the previous iteration (default is `ns/2`).
`verbose`	logical; if `TRUE`, progress is printed at each iteration.

Details

The GEMSS criterion is defined as:

$$\mathcal{G} = c_1\delta_1^2 + c_2\delta_2$$

where δ_1^2 is the squared prediction error and δ_2 is the posterior variance. The weights c_1 and $c_2 = 1 - c_1$ control the balance between prediction accuracy and space-filling properties.

The default value for c_1 is $3\delta_2^2/(2\delta_1^4 + 3\delta_2^2)$, which is derived by minimizing the variance of the GEMSS criterion. Setting $c_1 = 0$ results in space-filling subdata.

The algorithm initializes with a small random sample of size 2 and sequentially adds the point from the candidate set that maximizes $\mathcal{G}$ until the subdata reaches size ns.

The candidate set in each iteration is composed of two parts: a random sample of size n_srs, and the n_top points that yielded the highest $\mathcal{G}$ values in the previous iteration.

All GP hyperparameters except σ^2 are estimated from an initial pilot sample of size ns and remain fixed throughout the selection process.

If the response is non-stationary, one can first use a simpler model (such as a first-order linear model or a regression tree) to detrend the non-stationary part, and then apply `gemss_select` on the residuals. See Example 2 below for a demonstration of this two-stage approach.

Value

a list containing:

- `index`: the indices of the selected subdata.
- `r_sq`: a data.frame containing the subdata size and the corresponding predictive R-square.
- `parameters`: a list of hyperparameters containing `beta0`, `theta`, `g`, and `sigma2` (the plug-in estimator of the variance).

References

Chang, M. C., Hua, S. Z., & Wu, C. F. J. (2026). GEMSS-Driven Subsampling for Information Extraction and Redundancy Elimination. *Technometrics*, 1–20. doi:10.1080/00401706.2026.2670596

Examples

```
# --- Example 1: 1D Regression ---
fx <- function(x) cos((x - 0.8) * 2 * pi)^7 * sin(x) + 5 * sin(x) * (sin(x^2))^10
X <- matrix(runif(200), 200, 1)
Y <- apply(X, 1, fx) + rnorm(nrow(X), 0, 0.05)

# Select 20 points using GEMSS
res <- gemss_select(X, Y, ns = 20, covtype = "Matern5_2")

# Predict on a grid
x.grid <- as.matrix(seq(0, 1, 0.01))
y.pred <- gp_predict(x.grid, X[res$index, ], Y[res$index], "Matern5_2", res$parameters)

# Visualize 1D Results
plot(X, Y, col = "grey", main = "Selected Subdata and GP Prediction")
```

```
points(X[res$index, ], Y[res$index], col = "red", pch = 19)
lines(x.grid, y.pred$mean, col = "red", lwd = 2)

# --- Example 2: 2D Non-Stationary Surface (Modified Michalewicz Function) ---
# For highly non-stationary data, directly applying GEMSS can yield suboptimal
# results. A highly effective strategy is a two-stage approach:
# 1. Fit a simple, fast global model to detrend the data.
# 2. Apply GEMSS on the residuals to capture the complex, local variations.
# This example demonstrates how detrending improves the final prediction accuracy.

if (requireNamespace("ContourFunctions", quietly = TRUE)) {
  michalewicz_2d <- function(x) {
    x1 <- x[1] * pi
    x2 <- x[2] * pi
    m <- 10
   -(sin(x1) * (sin(x1^2 / pi))^(2 * m) + sin(x2) * (sin(2 * x2^2 / pi))^(2 * m)) + (x1 + x2)
  }

  set.seed(1)
  # Generate 1,000 data points
  X2D <- matrix(runif(2000), 1000, 2)
  Y2D <- apply(X2D, 1, michalewicz_2d) + rnorm(nrow(X2D), 0, 0.005)
  df2D <- data.frame(Y = Y2D, X1 = X2D[, 1], X2 = X2D[, 2])

  # --- Fit Models ---

  # A. Detrend the data using a first-order linear model (global trend)
  trend_model <- lm(Y ~ X1 + X2, data = df2D)
  Y2D_resid <- df2D$Y - predict(trend_model, newdata = df2D)

  # Apply GEMSS on Residuals (Two-Stage Method)
  res_twostage <- GEMSS::gemss_select(X2D, Y2D_resid, ns = 80, covtype = "Matern5_2")

  # B. Apply GEMSS Directly on Response (Direct Method for comparison)
  res_direct <- GEMSS::gemss_select(X2D, Y2D, ns = 80, covtype = "Matern5_2")


  # --- Set up the Visualization Grid ---
  ngrid <- 50
  gx <- seq(0, 1, len = ngrid)
  grid_matrix <- as.matrix(expand.grid(X1 = gx, X2 = gx))

  # Calculate true response and true residuals on the grid
  true_y_grid <- apply(grid_matrix, 1, michalewicz_2d)
  resid_grid <- true_y_grid - predict(trend_model, newdata = as.data.frame(grid_matrix))

  # Plot True Functions
  # ContourFunctions::cf_grid(gx, gx, matrix(true_y_grid, ngrid, ngrid),
  #                           bar = TRUE, main = "True Response Function")
  # ContourFunctions::cf_grid(gx, gx, matrix(resid_grid, ngrid, ngrid),
  #                           bar = TRUE, main = "Residual Function")
```

```
  # --- Predict on the Grid ---
  # A. Two-Stage Subdata Prediction (Global Trend + GP on Residuals)
  trend_pred_grid <- predict(trend_model, newdata = as.data.frame(grid_matrix))
  gp_resid_pred <- GEMSS::gp_predict(grid_matrix,
                                     X2D[res_twostage$index, ],
                                     Y2D_resid[res_twostage$index],
                                     "Matern5_2",
                                     res_twostage$parameters)
  pred_twostage_final <- trend_pred_grid + gp_resid_pred$mean

  # B. Direct Subdata Prediction (GP directly on original Y)
  gp_direct_pred <- GEMSS::gp_predict(grid_matrix,
                                      X2D[res_direct$index, ],
                                      Y2D[res_direct$index],
                                      "Matern5_2",
                                      res_direct$parameters)
  pred_direct_final <- gp_direct_pred$mean


  # --- Evaluate Prediction Accuracy (Normalized RMSPE) ---
  rmse_baseline <- mean((mean(Y2D) - true_y_grid)^2)^0.5
  nrmspe_direct <- mean((pred_direct_final - true_y_grid)^2)^0.5 / rmse_baseline
  nrmspe_twostage <- mean((pred_twostage_final - true_y_grid)^2)^0.5 / rmse_baseline

  # --- Plot the Final Predictions ---
  # Plot Direct Method
  ContourFunctions::cf_grid(gx, gx, matrix(pred_direct_final, ngrid, ngrid), bar = TRUE,
                            main = paste0("Direct GP Prediction | nRMSPE: ",
                                          round(nrmspe_direct, 4)),
                            afterplotfunc = function() {
                              points(X2D[res_direct$index, ], col = 'blue',
                                     pch = 1, cex = 1.5, lwd = 2)
                            })

  # Plot Two-Stage Method: The GP Residual Component
  # ContourFunctions::cf_grid(gx, gx, matrix(gp_resid_pred$mean, ngrid, ngrid),
  #                           bar = TRUE, main = "GP Prediction on Residuals Only",
  #                           afterplotfunc = function() {
  #                             points(X2D[res_twostage$index, ], col = 'green',
  #                                    pch = 1, cex = 1.5, lwd = 2)
  #                           })

  # Plot Two-Stage Method: The Final Combined Prediction
  ContourFunctions::cf_grid(gx, gx, matrix(pred_twostage_final, ngrid, ngrid), bar = TRUE,
                            main = paste0("Two-Stage Prediction | nRMSPE: ",
                                          round(nrmspe_twostage, 4)),
                            afterplotfunc = function() {
                              points(X2D[res_twostage$index, ], col = 'green',
                                     pch = 1, cex = 1.5, lwd = 2)
                            })
}
```

gp_predict *Gaussian Process Prediction*

Description

Performs Gaussian Process prediction for a given set of new design locations, using hyperparameters provided in the parameters list.

Usage

```
gp_predict(X_new, X, Y, covtype, parameters)
```

Arguments

X_new	a m x p matrix of new design locations to predict.
X	a n x p matrix of training design locations.
Y	a n x 1 numeric vector of observed responses at training locations.
covtype	a character string specifying the covariance kernel type. Must be one of "Gaussian", "Matern5_2", or "Matern3_2".
parameters	a list of hyperparameters containing theta, g, sigma2, and beta0.

Details

The function uses the standard GP prediction formulas:

$$\mu(x_{new}) = \beta_0 + k(x_{new}, X)K^{-1}(Y - \beta_0)$$

$$s^2(x_{new}) = \sigma^2[(1 + g) - k(x_{new}, X)(K + gI)^{-1}k(X, x_{new})]$$

Value

a list containing:

- mean: a m x 1 numeric vector of predicted means at X_new.
- sd2: a m x 1 numeric vector of predicted variances at X_new.

Index

compute_kernel, 2

gemss_remove, 3
gemss_select, 5
gp_predict, 9

mleHomGP, *5*

twin, *5*