Package ‘twingp’

September 20, 2024

Type Package

Title A Fast Global-Local Gaussian Process Approximation

Version 1.0.0

Date 2024-08-24

Copyright See the file COPYRIGHTS for copyright details

Description A global-local approximation framework for large-scale Gaussian process modeling. Please see Vakayil and Joseph (2024) <doi:10.1080/00401706.2023.2296451> for details. This work is supported by U.S. NSF grants CMMI-1921646 and DMREF-1921873.

License Apache License (== 2.0)

Depends R (>= 3.0.2)

Imports Rcpp, nloptr (>= 1.2.0)

LinkingTo Rcpp, RcppEigen, nloptr (>= 1.2.0)

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation yes

Author Akhil Vakayil [aut, cre] (<https://orcid.org/0000-0003-3684-6617>), V. Roshan Joseph [aut, ths] (<https://orcid.org/0000-0002-9430-5301>), Jose L. Blanco [ctb] (nanoflann author)

Maintainer Akhil Vakayil <akhilv@gatech.edu>

Repository CRAN

Date/Publication 2024-09-20 13:40:02 UTC

Contents

twingp-package 2
twingp 2

Index **5**

twingp-package *A Global-Local Approximation Framework for Large-Scale Gaussian Process Modeling*

Description

For further details on the methodology, please refer to Vakayil and Joseph (2024). The package uses `nlopt` (Johnson, 2007) C++ library for hyperparameter optimization, `nanoflann` (Blanco and Rai, 2014) C++ library for nearest neighbor queries, and `Eigen` (Guennebaud and Jacob, 2010) C++ library for matrix operations.

References

Vakayil, A., & Joseph, V. R. (2024). A Global-Local Approximation Framework for Large-Scale Gaussian Process Modeling. Technometrics, 66(2), 295-305.

Johnson, S. G. (2007), The NLopt nonlinear-optimization package. http://github.com/stevengj/nlopt

Guennebaud, G., Jacob, B., & Others. (2010). Eigen v3. http://eigen.tuxfamily.org

Blanco, J. L. & Rai, P. K. (2014). nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. https://github.com/jlblancoc/nanoflann

twingp *A Fast Global-Local Gaussian Process Approximation*

Description

A Fast Global-Local Gaussian Process Approximation

Usage

```
twingp(
  x,
  y,
  x_test,
  nugget = TRUE,
  twins = 5,
  g_num = NULL,
  l_num = NULL,
  v_num = NULL,
  num_threads = NULL
)
```

Arguments

x n * d numeric matrix representing the training features

y n * 1 response vector corresponding to x

x_test t * d numeric matrix representing the t testing locations

nugget Boolean indicating if a nugget to model observation noise is included in the model, the default is True

twins Number of twinning samples computed to identify the best set of global points, the default is 5

g_num Number of global points included in the model, the default is min(50 * d, max(sqrt(n), 10 * d))

l_num Number of local points included in the model, the default is max(25, 3 * d)

v_num Number of validation points, the default is 2 * g_num

num_threads Number of threads for parallel computation, the default is all available threads

Details

We employ a combined global-local approach in building the Gaussian process approximation. Our framework uses a subset-of-data approach where the subset is a union of a set of global points designed to capture the global trend in the data, and a set of local points specific to a given testing location. We use Twinning (Vakayil and Joseph, 2022) to identify the set of global points. The local points are identified as the nearest neighbors to the testing location. The correlation function is also modeled as a combination of a global, and a local kernel. For further details on the methodology, please refer to Vakayil and Joseph (2024).

Value

A list of two t * 1 vectors mu, and sigma representing the mean prediction and associated standard error corresponding to x_test

References

Vakayil, A., & Joseph, V. R. (2024). A Global-Local Approximation Framework for Large-Scale Gaussian Process Modeling. Technometrics, 66(2), 295-305.

Vakayil, A., & Joseph, V. R. (2022). Data Twinning. Statistical Analysis and Data Mining: The ASA Data Science Journal, 15(5), 598-610.

Examples

```
grlee12 = function(x) {
  term1 = sin(10 * pi * x) / (2 * x)
  term2 = (x - 1)^4
  y = term1 + term2
  return(y)
}

x = matrix(seq(0.5, 2.5, length=500), ncol=1)
y = apply(x, 1, grlee12) + rnorm(nrow(x)) * 0.1
```

```
x_test = matrix(seq(0.5, 2.5, length=2000), ncol=1)
y_test = apply(x_test, 1, grlee12)

result = twingp(x, y, x_test, num_threads=2)
rmse = sqrt(mean((y_test - result$mu)^2))
```

Index

twingp, 2
twingp-package, 2