Package 'nsgp'

August 29, 2016

Title Non-Stationary Gaussian Process Regression

Version 1.0.5

Date 2014-10-14

Author Markus Heinonen

Maintainer Markus Heinonen <markus.heinonen@gmail.com>

Depends R (>= 3.0.0)

Imports MASS

Description A Gaussian process regression using a Gaussian kernel for both one-sample and two-sample cases. Includes non-stationary Gaussian kernel (exponential decay function) and several likelihood ratio tests for differential testing along target points.

License GPL-2

URL http://amis-group.fr/software/nsgp

NeedsCompilation no

Repository CRAN

Date/Publication 2014-10-15 06:54:09

R topics documented:

`gpr1sample` *Perform one-sample GP regression*

Description

Computes the optimal GP model by optimizing the marginal likelihood

Usage

```
gpr1sample(x, y, x.targets, noise = NULL, nsnoise = TRUE, nskernel = TRUE,
  expectedmll = FALSE, params = NULL, defaultparams = NULL,
  lbounds = NULL, ubounds = NULL, optim.restarts = 3,
  derivatives = FALSE)
```

Arguments

`x`	input points
`y`	output values (same length as `x`)
`x.targets`	target points
`noise`	observational noise (variance), either NULL, a constant scalar or a vector
`nsnoise`	estimate non-stationary noise from replicates, if possible (default)
`nskernel`	use non-stationary kernel
`expectedmll`	use an alternative expected mll optimization criteria
`params`	gaussian kernel parameters: `(sigma.f, sigma.n, l, lmin, c)`
`defaultparams`	initial parameters for optimization (5-length vector)
`lbounds`	lower bounds for parameters (5-length vector)
`ubounds`	upper bounds for parameters (5-length vector)
`optim.restarts`	restarts in the gradient ascent (default=3)
`derivatives`	compute also GP derivatives

Details

Parameter optimization performed through L-BFGS using analytical gradients with restarts. The input points `x` and output values `y` need to be matching length vectors. If replicates are provided, they are used to estimate dynamic observational noise.

The resulting GP model is encapsulated in the return object. The estimated posterior is in `targets$pmean` and `targets$pstd` for target points `x.targets`. Use `plot.gp` to visualize the GP.

Value

A gp-object (list) containing following elements

`targets`	data frame of predictions with points as rows and columns..
`_$x`	points
`_$pmean`	posterior mean of the gp
`_$pstd`	posterior standard deviation of the gp
`_$noisestd`	noises (variance)
`_$mll`	the MLL log likelihood ratio
`_$emll`	the EMLL log likelihood ratio
`_$pc`	the posterior concentration log likelihood ratio
`_$npc`	the noisy posterior concentration log likelihood ratio
`cov`	learned covariance matrix
`mll`	marginal log likelihood value
`emll`	expected marginal log likelihood value
`kernel`	the kernel matrix used
`ekernel`	the EMLL kernel matrix
`params`	the learned parameter vector:
`_$sigma.f`	kernel variance
`_$sigma.n`	kernel noise
`_$l`	maximum lengthscale
`_$lmin`	minimum lengthscale
`_$c`	curvature
`x`	the input points
`y`	the output values

See Also

`gpr2sample` `plot.gp`

Examples

```
# load example data
data(toydata)

## Not run: can take sevaral minutes
 # perform gpr
 res = gpr1sample(toydata$ctrl$x, toydata$ctrl$y, seq(0,22,0.1))
 print(res)
## End(Not run)

# pre-computed toydata model
data(toygps)
print(toygps$ctrlmodel)
```

gpr2sample *Performs two-sample GP regression*

Description

Performs gaussian process regression for two time-series: control and case. A third null GP model is learned that assumes both data coming from same process. Various likelihood ratios between the null and individual models are estimated to distinguish when case and control processes are significantly different. Use `plot.gppack` to visualize the models.

Usage

```
gpr2sample(x.ctrl, y.ctrl, x.case = NULL, y.case, x.targets,
  noise.ctrl = NULL, noise.case = NULL, nsnoise = TRUE, nskernel = TRUE,
  expectedmll = FALSE, params.ctrl = NULL, params.case = NULL,
  defaultparams = NULL, lbounds = NULL, ubounds = NULL,
  lockatzero = FALSE, optim.restarts = 3, derivatives = FALSE)
```

Arguments

`x.ctrl`	input points (control)
`y.ctrl`	output values (control)
`x.case`	input points (case)
`y.case`	output values (case)
`x.targets`	target points
`noise.ctrl`	observational noise
`noise.case`	observational noise
`nsnoise`	estimate non-stationary noise function from replicates, if available
`nskernel`	use non-stationary kernel (default)
`expectedmll`	use expected MLL optimization criteria
`params.ctrl`	kernel parameters (control)
`params.case`	kernel parameters (case)
`defaultparams`	initial parameters for optimization
`lbounds`	lower bounds for parameter optimization
`ubounds`	upper bounds for parameter optimization
`lockatzero`	estimate a pseudo-observation for time 0
`optim.restarts`	restarts in the gradient ascent (default=3)
`derivatives`	compute also GP derivatives

Details

The control and case do not need have same amount of points. The resulting `gppack` object contains the three learned models and the likelihood ratios along `x.targets`.

Value

a gppack-object that contains

`ctrlmodel`	the gp-object corresponding to the control data
`casemodel`	the gp-object corresponding to the case data
`nullmodel`	the gp-object corresponding to the shared null data
`ratios`	the log likelihood ratios between the control and case against the null model, contains..
`______$mll`	marginal log likelihood ratio
`______$emll`	expected marginal log likelihood ratio
`______$pc`	log posterior concentration ratio
`______$npc`	log noisy posterior concentration ratio

See Also

`gpr1sample` `plot.gppack`

Examples

```
# read toy data
data(toydata)

## Not run: can take several minutes
 # perform two-sample regression
 res = gpr2sample(toydata$ctrl$x, toydata$ctrl$y, toydata$case$x, toydata$case$y, seq(0,22,0.1))
 print(res)
## End(Not run)

# pre-computed model for toydata
data(toygps)
print(toygps)
```

`nsgp` *Gaussian process regression and statistical testing*

Description

This package implements the non-stationary gaussian processes for one- and two-sample cases, and statistical likelihood ratio tests for distinguishing when two time-series are significantly different. The package offers two main functions: `gpr1sample` and `gpr2sample`.

Details

The function gpr1sample learns a gaussian process model that uses either stationary or non-stationary gaussian kernel, which assumes a perturbation at (time) point 0. The non-stationarity is controlled by a time-dependent lengthscale in the gaussian kernel. The time-dependency $l - (l - l_{min})e^{-ct}$ follows exponential decay, such that it starts at value `l.min` and grows logarithmically to `l` by curvature parameter `c`.

In gpr2sample we compare control and case time-series by building GP models for both of them individually, while also building a third null model for joint data (assume that data come from the same process). The null model and the case/control models are then compared with likelihood ratios for significant different along time. The package includes standard marginal log likelihood (MLL) ratio, and three novel ones: expected marginal log likelihood (EMLL) measures the ratio between the models while discarding data; posterior concentration (PC) ratio measures the difference of variance between null and individual models; and noisy posterior concentration (NPC) ratio also compares observational noises.

`plot.gp` *Plot a gaussian process*

Description

Plots a gaussian process. Several boolean parameters for modifying the plot. By default plots the data, posterior mean and 95% interval.

Usage

```
## S3 method for class 'gp'
plot(x, y = NULL, plotdata = TRUE, plotmean = TRUE,
  plotcov = TRUE, plotnoise = FALSE, samples = 0, sigma = 2,
  title = NULL, legend = FALSE, plotgradient = TRUE, plotls = FALSE,
  ...)
```

Arguments

`x`	the gp-object
`y`	placeholder variable
`plotdata`	plot the data (default)
`plotmean`	plot the GP mean (default)
`plotcov`	plot the GP covariances (default)
`plotnoise`	plot the observational noise (default)
`samples`	plot N samples from the GP
`sigma`	variance level to plot
`title`	plot title
`legend`	plot legend
`plotgradient`	use gradient graphics
`plotls`	plot lengthscale function
`...`	...

Examples

```
# read toy data
data(toydata)

## Not run: can take several minutes
# perform one-sample regression
res = gpr2sample(toydata$ctrl$x, toydata$ctrl$y, seq(0,22,0.1))

# pre-computed model for toydata
data(toygps)
res = toygps$ctrlmodel

# basic plot with data, estimated mean and 95\%
plot(res)

# don't plot the data, plot some samples drawn from the learned gp
plot(res, plotdata=FALSE, samples=3)
## End(Not run)
```

`plot.gppack` *Plots several GP's simultaneously*

Description

Plots the GP's corresponding to the control and case data, as well as the null model. Visualizes the log likelihood ratios between the null and individual models. Several boolean parameters for modifying the plot. By default plots the data, posterior mean and 95% interval for CASE and CONTROL.

Usage

```
## S3 method for class 'gppack'
plot(x, y = NULL, plotdata = TRUE, plotmeans = TRUE,
  plotcovs = TRUE, plotnoises = FALSE, plotnull = FALSE,
  plotratios = "emll", thr = 1, samples = 0, sigma = 2, title = NULL,
  legend = FALSE, plotgradient = TRUE, ...)
```

Arguments

`x`	the gppack-object
`y`	placeholder variable
`plotdata`	plot the data (default)
`plotmeans`	plot the GP mean (default)
`plotcovs`	plot the GP covariances (default)
`plotnoises`	plot the observational noise (default)
`plotnull`	plots also the null model

plotratios	plots the ratios, choices are emll, mll, npc, pc
thr	ratio threshold
samples	plot N samples from the GP
sigma	variance level to plot
title	plot title
legend	plot legend
plotgradient	use gradient graphics
...	...

Details

The threshold thr is the logarithmic likelihood ratio between null and control+case models. The default value 1 hence corresponds to a likelihood ratio of 2.72.

Examples

```
# read toy data
data(toydata)

## Not run: can take several minutes
# perform two-sample regression
res = gpr2sample(toydata$ctrl$x, toydata$ctrl$y, toydata$case$x, toydata$case$y, seq(0,22,0.1))

# pre-computed model for toydata
data(toygps)
res = toygps

# basic plot
plot(res)

# plot also the null model, don't plot data, means or noise
plot(res, plotnull=TRUE, plotdata=FALSE, plotmeans=FALSE, plotnoise=FALSE)
## End(Not run)
```

---

print.gp *prints the one-sample GP summary*

Description

prints the one-sample GP summary

Usage

```
## S3 method for class 'gp'
print(x, ...)
```

Arguments

x	the estimated GP-object
...	for compatibility

`print.gppack` *prints the two-sample GP summary*

Description

prints the two-sample GP summary

Usage

```
## S3 method for class 'gppack'
print(x, ...)
```

Arguments

x	the estimated gppack-object containing ctrlmodel, casemodel and nullmodel
...	for compability

`simulategp` *Generate simulated GP models*

Description

Generate simulated GP models

Usage

```
simulategp(N = 100, reps = 3, obs = c(0, 5, 10, 15, 20), xs = seq(0, 20,
  0.2), filename = NULL, l.noise = 12, l.shape = 4, l.scale = 4,
  sigmaf.noise = 0.25, sigmaf.shape = 1.5, sigmaf.scale = 0.3)
```

Arguments

N	number of gp's
reps	replicate observations
obs	observation timepoints
xs	target timepoints
filename	file to save the results
l.noise	Noise model lengthscale
l.shape	shape of lengthscale Gamma distribution

l.scale	scale of lengthscale Gamma distribution
sigmaf.noise	Noise model sigma.f
sigmaf.shape	shape of sigma.f Gamma distribution
sigmaf.scale	scale of sigma.f Gamma distribution

Value

List with

simdata	Simulated datamatrix
gps	Simulated GPs

See Also

simulategp.perturbed

---

simulategp.perturbed *Generate simulated perturbed GP models*

Description

Takes pregenerated simulated GP models as input, and models both the perturbation and additional perturbation variance as GP models, that are all combined into a perturbed GP model. We sample reps data points from this at timepoints obs

Usage

```
simulategp.perturbed(N = 100, reps = 3, obs = c(0, 5, 10, 15, 20),
  gps.ctrl = NULL, filename = NULL, xs = seq(0, 20, 0.2), l.noise = 12,
  l.shape = 2, l.scale = 2.5, sigmaf.noise = 0.25, sigmaf.shape = 4,
  sigmaf.scale = 0.5)
```

Arguments

N	number of gp's
reps	replicate observations
obs	observation timepoints
gps.ctrl	the GPS models to be perturbed
filename	file to save the results
xs	target timepoints
l.noise	Noise model lengthscale
l.shape	shape of lengthscale Gamma distribution
l.scale	scale of lengthscale Gamma distribution
sigmaf.noise	Noise model sigma.f
sigmaf.shape	shape of sigma.f Gamma distribution
sigmaf.scale	scale of sigma.f Gamma distribution

Value

List with

simdata	Simulated datamatrix
gps	Simulated GPs

See Also

simulategp

---

summary.gp *prints the one-sample GP summary*

Description

identical to print.gp

Usage

```
## S3 method for class 'gp'
summary(object, ...)
```

Arguments

object	the estimated GP-object
...	for compatibility

---

summary.gppack *prints the two-sample GP summary*

Description

identical to print.gppack

Usage

```
## S3 method for class 'gppack'
summary(object, ...)
```

Arguments

object	the estimated gppack-object containing ctrlmodel, casemodel and nullmodel
...	for compability

toydata — *Toy time-series data for testing, contains two time-series, case and control*

Description

Toy time-series data for testing, contains two time-series, case and control

Usage

```
data(toydata)
```

Format

A list of two dataframes containing 24 (x,y) pairs both, i.e. toydata$ctrl and toydata$case

Source

randomly generated data

toygps — *Toy time-series model for testing*

Description

contains the GP models for the 'toydata'

Usage

```
data(toygps)
```

Format

A gppack object

Source

prelearned model with `gpr2sample`

Index