Package 'deGradInfer'

January 20, 2020

Title Parameter Inference for Systems of Differential Equation

Version 1.0.1

Description Efficient Bayesian parameter inference for systems of ordinary differential equations. The inference is based on adaptive gradient matching (AGM, Dondelinger et al. 2013 <http://proceedings.mlr.press/v31/dondelinger13a.pdf>, Macdonald 2017 <http://theses.gla.ac.uk/7987/1/2017macdonaldphd.pdf>), which offers orders-of-magnitude improvements in computational efficiency over standard methods that require solving the differential equation system. Features of the package include flexible specification of custom ODE systems as R functions, support for missing variables, Bayesian inference via population MCMC.

Depends R (>= 3.3.1)

License GPL-3

Encoding UTF-8

LazyData true

Imports deSolve, gdata, gptk, graphics, stats

RoxygenNote 7.0.2

Suggests testthat, knitr, rmarkdown, ggplot2

VignetteBuilder knitr

NeedsCompilation no

Author Benn Macdonald [aut],
Frank Dondelinger [aut, cre]

Maintainer Frank Dondelinger <fdondelinger.work@gmail.com>

Repository CRAN

Date/Publication 2020-01-20 19:30:25 UTC

R topics documented:

---

agm *Main function for adaptive gradient matching*

Description

Function agm uses adaptive gradient matching to infer the parameters of a user-defined ODE system from data. For details on AGM, see e.g. Dondelinger et al. (2013), Macdonald (2017).

Usage

```
agm(
  data,
  time,
  ode.system,
  numberOfParameters,
  noise.sd = 0.001,
  observedVariables = 1:ncol(data),
  temperMismatchParameter = FALSE,
  initialisedParameters = NULL,
  chainNum = 20,
  gpCovType = "rbf",
  saveFile = NULL,
  defaultTemperingScheme = NULL,
  maxIterations = 3e+05,
  showPlot = FALSE,
  showProgress = FALSE,
  mismatchParameterValues = NULL,
  originalSignalOnlyPositive = FALSE,
  logPrior = "Uniform",
  explicit = FALSE,
  explicitNoiseInfer = TRUE
)
```

Arguments

data	A matrix of observations of the ODE system over time. The number of rows is equal to the number time points and the number of columns is equal to the number of variables in the system.
time	A vector containing the time points at which the observations were made.
ode.system	A function describing the ODE system. See Details for more information.
numberOfParameters	A scalar specifying the number of parameters in the ODE system. If explicitly solving the ODE system, the number of parameters will (usually) be equal to the number of ODE parameters plus the number of initial conditions of the system.
noise.sd	A scalar specifying the value at which to fix the standard deviation of the observational noise. Default noise.sd=1e-3.
observedVariables	A vector specifying which variables are observed in the system. Default is observedVariables=1:ncol(data) (fully observed system).
temperMismatchParameter	Logical: whether tempering of the gradient mismatch parameter be carried out? Default is temperMismatchParameter=FALSE.
initialisedParameters	A vector containing ODE parameters at which to intialise the MCMC. Can be set as NULL to initialise with a random draw from the prior distribution. Default is initialisedParameters=NULL.
chainNum	A scalar specifying the number of parallel temperature chains. Default is\ chainNum=20.
gpCovType	A string specifying the choice of kernel for the Gaussian process. Currently, there are two: gpCovType="rbf" and gpCovType="sigmoidVar".
saveFile	A string specifying the path and name of the file containing the result output. Can be set as NULL to save as "AGM Results.R" to the current working directory. Default is saveFile=NULL.
defaultTemperingScheme	A string indicating which of the two default gradient mismatch parameter value ladders to use. Choices are defaultTemperingScheme="LB2" or\ defaultTemperingScheme="LB10". Should only be used when temperMismatchParameter=TRUE. Default is\ defaultTemperingScheme=NU
maxIterations	A scalar specifying the number of total MCMC iterations. Default is\ maxIterations=300000.
showPlot	Logical: whether plots of the MCMC progress should be displayed. Default is showPlot=FALSE.
showProgress	Logical: whether % completion and various parameter values should be printed to the workspace. Default is showProgress=FALSE.
mismatchParameterValues	A matrix containing user specified values for the gradient mismatch parameter. The number of rows should be equal to chainNum and the number of columns should be equal to the number of variables in the system. A typical ladder should have the largest value in the first row and the smallest value in the last row. Should only be used when defaultTemperingScheme=NULL. Default is mismatchParameterValues=NULL.

originalSignalOnlyPositive
: Logical: whether all signals observed should be non-negative. When originalSignalOnlyPositive=TR any negative values of the sampled interpolant will be set to zero. Default is originalSignalOnlyPositive=FALSE.

logPrior
: A string specifying whether one of the default log priors for the ODE parameters should be used, or a user-specified function. Current choices for the default prior are "Uniform", "Gamma" (shape=4, rate=2) and "Mixed" (3 ODE parameters; N(mean=0, sd=0.4), N(mean=0, sd=0.4). Alternatively the user may specify a function for calculating the prior, see Details below. Default is logPrior='Uniform'.

explicit
: Logical: whether the ODE system should be explicitly solved, rather than doing gradient matching. This means that the Gaussian process model is ignored, and the ODE system is directly fitted to the observed data. Default is explicit=FALSE.

explicitNoiseInfer
: Logical: whether the standard deviation of the observational noise should be inferred when using the method that explicitly solves the ODEs. Only considered when explicit=TRUE. Default is explicitNoiseInfer=TRUE.

Details

The parameters ode.system should be a function of the form f(t,X,params) where t is the time point vector for which the derivatives should be calculated, X is a T by p matrix containing the values of the variables in the system at time t, and params is a vector with the current estimated parameter values. The function should return a matrix with the derivatives of x with respect to time (in the same order as in x). Note that in order to be consistent with the ode in package deSolve, we require that the function also works for input at a single time point.

For specifying a custom prior on the parameters, the user should write their function to take as input a vector of parameters, and return a vector of log densities for a given parameter set. For example, logPrior = function(params) c(dgamma(params,1,1,log=TRUE) defines a Gamma parameter prior with shape and scale 1.

Value

Function returns a list with elements posterior.mean, the mean of the parameter samples from the posterior (after burning of 1/4 of the number of samples taken), posterior.sd, the standard deviation, posterior.samples, the parameter samples, ll, the log likelihood, x.samples, the samples of the latent variables, gp.samples, the samples of the GP hyperparameters, noise.samples, the samples of sigma (currently fixed), swappedChains, the number of times the chains have been swapped, ll.all.chain, the log likelihood for all chains and tuning, the inferred tuning parameters for acceptance of the MCMC moves.

Examples

```
dataTest <- LV_example_dataset$data
timeTest <- LV_example_dataset$time
noiseTest <- LV_example_dataset$noise
```

```
LV_func = function(t, X, params) {
dxdt = cbind(
  X[,1]*(params[1] - params[2]*X[,2]),
  - X[,2]*(params[3] - params[4]*X[,1])
)
return(dxdt)
}

# Example run only; to achieve convergence the number of iterations and
# chains must be increased.
param.result = agm(data=dataTest,time=timeTest,noise.sd=0.31,ode.system=LV_func,
    numberOfParameters=4,temperMismatchParameter=TRUE,
    chainNum=4, maxIterations=150,originalSignalOnlyPositive=TRUE,
    logPrior="Gamma",defaultTemperingScheme="LB10")

print(param.result$posterior.mean)
```

---

doMCMC *Main MCMC function Runs the MCMC for the specified number of iterations and returns the sampled parameter values*

Description

Main MCMC function Runs the MCMC for the specified number of iterations and returns the sampled parameter values

Usage

```
doMCMC(timePoints, data, auxVars, options)
```

Arguments

timePoints	Measured time points for the ODE system.
data	Observed data.
auxVars	Auxiliary variables.
options	Options for MCMC run.

Value

Function returns a list with elements parameters, the sampled ODE parameters for the current MCMC iteration, tuning, the inferred tuning parameters for acceptance of the MCMC moves, paramsRec, the sampled ODE parameters recorded over all MCMC iterations, lLRec, the log likelihood recorded over all MCMC iterations, xRec, the samples from the Gaussian process of the latent variables recorded over all MCMC iterations, gpRec, the samples of the hyperparameters for the Gaussian process recorded over all MCMC iterations, timePoints, the time points, noiseRec, the standard deviation of the observational noise recorded over all MCMC iterations (currently fixed), swappedChains, the number of times the chains have been swapped, chainNums, the number of

chains, maxIterations, the total number of MCMC iterations and lLAllChains, the log likelihood for all chains recorded over all MCMC iterations. If the user specifies temperMismatchParameter=FALSE, the function additionally returns gradientMismatchParameterRec, the sampled gradient mismatch parameters recorded over all MCMC iterations.

`getODEGradient` *Calculate gradients from ODE system*

Description

Calculate gradients from ODE system

Usage

```
getODEGradient(X, timePoints, params, auxVars, species = 1:dim(X)[2])
```

Arguments

X	Latent values for the species
timePoints	Times at which to calculate the ODE gradients
params	Current parameter estimates
auxVars	Auxiliary variables (including function for ODE gradients)
species	Which species to return (default=all)

Value

A T by length(species) matrix with the gradients calculated at each time point for the specified species.

`LV_example_dataset` *Data from a Lotka-Volterra ODE system with 2 species and 4 parameters. Species in order are: 1. Sheep (Prey) 2. Wolves (Predators)*

Description

Data from a Lotka-Volterra ODE system with 2 species and 4 parameters. Species in order are: 1. Sheep (Prey) 2. Wolves (Predators)

Usage

```
LV_example_dataset
```

Format

A list with 5 components

data Observed species with observation noise.

data.true Observed species without observation noise.

time Observed time points.

params Observed parameters.

noise Variance for the Gaussian observation noise.

Source

Generated by function simulateLotkaVolterraModel.

`proposeParamsMCMC` *Sample from proposal distribution for MCMC*

Description

Sample from proposal distribution for MCMC

Usage

```
proposeParamsMCMC(oldParams, inferredParams, width)
```

Arguments

`oldParams`	Previous parameter values
`inferredParams`	Proposed parameter values
`width`	Width of random walk proposal

Value

List with proposed parameters, indicator variable of which parameters have changed, old and new proposal probabilities (if

`sigmoidVarKernCompute` *Compute K(x, x2) for sigmoid kernel, used by gptk*

Description

Compute K(x, x2) for sigmoid kernel, used by gptk

Usage

```
sigmoidVarKernCompute(kern, x, x2 = NULL)
```

Arguments

`kern`	kernel
`x`	input 1
`x2`	input 2

Value

K(x, x2)

`sigmoidVarKernDiagCompute`
Compute diagonal of sigmoid kernel (used by gptk).

Description

Compute diagonal of sigmoid kernel (used by gptk).

Usage

```
sigmoidVarKernDiagCompute(kern, x)
```

Arguments

`kern`	Kernel
`x`	Input

Value

Diagonal of the kernel

sigmoidVarKernExpandParam *Insert parameters into sigmoid kernel (used by gptk)*

Description

Insert parameters into sigmoid kernel (used by gptk)

Usage

```
sigmoidVarKernExpandParam(kern, params)
```

Arguments

kern	kernel
params	parameters

Value

kernel

sigmoidVarKernExtractParam *Auxiliary function for sigmoid kernel (used by gptk)*

Description

Auxiliary function for sigmoid kernel (used by gptk)

Usage

```
sigmoidVarKernExtractParam(
  kern,
  only.values = TRUE,
  untransformed.values = TRUE
)
```

Arguments

kern	- kernel
only.values	- return values only
untransformed.values	- transform values

Value

hyperparameters of the GP kernel

`sigmoidVarKernGradient` *Compute gradient of sigmoid kernel with respect to each parameter (used by gptk)*

Description

Compute gradient of sigmoid kernel with respect to each parameter (used by gptk)

Usage

```
sigmoidVarKernGradient(kern, x, x2, covGrad)
```

Arguments

`kern`	kernel
`x`	input 1
`x2`	input 2
`covGrad`	gradient of covariance function

Value

d k(x, x2) / d theta

`sigmoidVarKernParamInit` *Auxiliary function for sigmoid kernel (used by gptk)*

Description

Auxiliary function for sigmoid kernel (used by gptk)

Usage

```
sigmoidVarKernParamInit(kern)
```

Arguments

`kern`	- GP kernel

Value

initialized kernel

`solveODE` *Solve ODE system explicitly.*

Description

Solve ODE system explicitly.

Usage

```
solveODE(num.species, timePoints, ode.system, params)
```

Arguments

`num.species`	Number of variables (species) in the system.
`timePoints`	Time points at which to evaluate the ODE system.
`ode.system`	Function for calculating the derivatives of the ODE system.
`params`	Current values for the ODE parameter estimates.

Value

A list with two elements: `x` contains the results of integrating the ODE at the given time points, and `error` flags if there has been an error while invoking `deSolve`.

Index