Package 'sgmcmc'

October 14, 2022

Type Package

Title Stochastic Gradient Markov Chain Monte Carlo

Version 0.2.5

Description
Provides functions that performs popular stochastic gradient Markov chain Monte Carlo (SGMCMC) methods on user specified models. The required gradients are automatically calculated using 'TensorFlow' <https://www.tensorflow.org/>, an efficient library for numerical computation. This means only the log likelihood and log prior functions need to be specified. The methods implemented include stochastic gradient Langevin dynamics (SGLD), stochastic gradient Hamiltonian Monte Carlo (SGHMC), stochastic gradient Nose-Hoover thermostat (SGNHT) and their respective control variate versions for increased efficiency. References: M. Welling, Y. W. Teh (2011) <http://www.icml-2011.org/papers/398_icmlpaper.pdf>; T. Chen, E. B. Fox, C. E. Guestrin (2014) <arXiv:1402.4102>; N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, H. Neven (2014) <https://papers.nips.cc/paper/5592-bayesian-sampling-using-stochastic-gradient-thermostats>; J. Baker, P. Fearnhead, E. B. Fox, C. Nemeth (2017) <arXiv:1706.05439>. For more details see <doi:10.18637/jss.v091.i03>.

License GPL-3

Encoding UTF-8

Depends R (>= 3.0), tensorflow

Imports utils, reticulate

SystemRequirements TensorFlow (https://www.tensorflow.org/), TensorFlow Probability (https://www.tensorflow.org/probability/)

Suggests testthat, MASS, knitr, ggplot2, rmarkdown

LazyData true

VignetteBuilder knitr

RoxygenNote 6.0.1

URL https://github.com/STOR-i/sgmcmc

BugReports https://github.com/STOR-i/sgmcmc/issues

NeedsCompilation no

Author Jack Baker [aut, cre, cph],
Christopher Nemeth [aut, cph],
Paul Fearnhead [aut, cph],
Emily B. Fox [aut, cph],
STOR-i [cph]

Maintainer Jack Baker <jackbaker92@mail.com>

Repository CRAN

Date/Publication 2019-10-24 18:10:02 UTC

R topics documented:

`getDataset` *Load example datasets*

Description

Download and load one of the example datasets for the package: `covertype` or `mnist`. These datasets are required for the vignettes in the package. The code generating these datasets is available at https://github.com/jbaker92/sgmcmc-data.

Usage

```
getDataset(dataset)
```

Arguments

`dataset`	string which determines the dataset to load: either `"covertype"` or `"mnist"`.

Value

Returns the desired dataset. The next two sections give more details about each dataset.

covertype

The samples in this dataset correspond to 30×30m patches of forest in the US, collected for the task of predicting each patch's cover type, i.e. the dominant species of tree. We use the LIBSVM dataset, which transforms the data to a binary problem rather than multiclass.

format: A matrix with 581012 rows and 55 variables. The first column is the classification labels, the other columns are the 54 explanatory variables.

source: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

mnist

The MNIST dataset is a dataset of handwritten digits from 0-9. Each image is 28x28 pixels. We can interpret this as a large matrix of numbers, representing the value at each pixel. These 28x28 matrices are then flattened to be vectors of length 784. For each image, there is an associated label, which determines which digit the image is of. This image is encoded as a vector of length 10, where element i is 1 if the digit is i-1 and 0 otherwise. The dataset is split into two parts: 55,000 data points of training data and 10,000 points of test data.

format: A list with two elements `train` and `test`.

- The training set mnist$train is a list with two entries: images and labels, located at mnist$train$images, mnist$train$labels respectively.
- The dataset mnist$train$images is a matrix of size 55000x784, the labels mnist$train$labels is a matrix of size 55000x10.
- The test set mnist$test is a list with two entries: images and labels, located at mnist$test$images, mnist$test$labels respectively.
- The dataset mnist$test$images is a matrix of size 10000x784, the labels mnist$test$labels is a matrix of size 10000x10.

source: http://yann.lecun.com/exdb/mnist/

Examples

```
## Not run:
# Download the covertype dataset
covertype = get_dataset("covertype")
# Download the mnist dataset
mnist = get_dataset("mnist")

## End(Not run)
```

getParams *Get current parameter values*

Description

Return the current parameter values as a list of R arrays (converted from TensorFlow tensors).

Usage

```
getParams(sgmcmc, sess)
```

Arguments

sgmcmc	a stochastic gradient MCMC object returned by *Setup such as sgldSetup, sgldcvSetup etc.
sess	a TensorFlow session created using initSess

Value

Returns a list with the same names as params, with R arrays of the current parameter values

Examples

```
## Not run:
# Simulate from a Normal Distribution, unknown location and known scale with uninformative prior
# Run sgmcmc step by step and calculate estimate of location on the fly to reduce storage
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
sgld = sgldSetup(logLik, dataset, params, stepsize)
nIters = 10^4L
# Initialize location estimate
locEstimate = 0
# Initialise TensorFlow session
sess = initSess(sgld)
for ( i in 1:nIters ) {
    sgmcmcStep(sgld, sess)
    locEstimate = locEstimate + 1 / nIters * getParams(sgld, sess)$theta
}
# For more examples see vignettes

## End(Not run)
```

`initSess` *Initialise TensorFlow session and sgmcmc algorithm*

Description

Initialise the TensorFlow session and the sgmcmc algorithm. For algorithms with control variates this will find the MAP estimates of the log posterior and calculate the full log posterior gradient at this point. For algorithms without control variates this will simply initialise a TensorFlow session.

Usage

```
initSess(sgmcmc, verbose = TRUE)
```

Arguments

`sgmcmc`	an sgmcmc object created using *Setup e.g. `sgldSetup`, `sgldcvSetup`
`verbose`	optional. Default TRUE. Boolean specifying whether to print progress.

Value

sess a TensorFlow session, see the TensorFlow for R website for more details.

Examples

```
## Not run:
# Simulate from a Normal Distribution, unknown location and known scale with uninformative prior
# Run sgmcmc step by step and calculate estimate of location on the fly to reduce storage
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
sgld = sgldSetup(logLik, dataset, params, stepsize)
nIters = 10^4L
# Initialize location estimate
locEstimate = 0
# Initialise TensorFlow session
sess = initSess(sgld)
for ( i in 1:nIters ) {
    sgmcmcStep(sgld, sess)
    locEstimate = locEstimate + 1 / nIters * getParams(sgld, sess)$theta
}
# For more examples see vignettes

## End(Not run)
```

installTF *Install TensorFlow and TensorFlow Probability*

Description

Install the python packages required by sgmcmc, including TensorFlow and TensorFlow probability. Uses the tensorflow::install_tensorflow function.

Usage

```
installTF()
```

sghmc *Stochastic Gradient Hamiltonian Monte Carlo*

Description

Simulates from the posterior defined by the functions logLik and logPrior using stochastic gradient Hamiltonian Monte Carlo. The function uses TensorFlow, so needs TensorFlow for python installed. Currently we use the approximation $\hat{\beta} = 0$, as used in the simulations by the original reference. This will be changed in future implementations.

Usage

```
sghmc(logLik, dataset, params, stepsize, logPrior = NULL,
  minibatchSize = 0.01, alpha = 0.01, L = 5L, nIters = 10^4L,
  verbose = TRUE, seed = NULL)
```

Arguments

logLik	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
dataset	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
params	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
stepsize	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
logPrior	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.

minibatchSize	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
alpha	optional. Default 0.01. List of numeric values corresponding to the SGHMC momentum tuning constants (α in the original paper). One value should be given for each parameter in params, the names should correspond to those in params. Alternatively specify a single float to specify that value for all parameters.
L	optional. Default 5L. Integer specifying the trajectory parameter of the simulation, as defined in the main reference.
nIters	optional. Default 10^4L. Integer specifying number of iterations to perform.
verbose	optional. Default TRUE. Boolean specifying whether to print algorithm progress
seed	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

Returns list of arrays for each parameter containing the MCMC chain. Dimension of the form (nIters,paramDim1,paramDim2,...)

References

- Chen, T., Fox, E. B., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In ICML (pp. 1683-1691).

Examples

```
## Not run:
# Simulate from a Normal Distribution with uninformative, improper prior
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-5)
output = sghmc(logLik, dataset, params, stepsize)
# For more examples see vignettes

## End(Not run)
```

---

sghmccv *Stochastic Gradient Hamiltonian Monte Carlo with Control Variates*

Description

Simulates from the posterior defined by the functions logLik and logPrior using stochastic gradient Hamiltonian Monte Carlo with an improved gradient estimate that is calculated using control variates. Currently we use the approximation $\hat{\beta} = 0$, as used in the simulations by the original reference. This will be changed in future implementations.

Usage

```
sghmccv(logLik, dataset, params, stepsize, optStepsize, logPrior = NULL,
  minibatchSize = 0.01, alpha = 0.01, L = 5L, nIters = 10^4L,
  nItersOpt = 10^4L, verbose = TRUE, seed = NULL)
```

Arguments

logLik	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
dataset	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
params	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
stepsize	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
optStepsize	numeric value specifying the stepsize for the optimization to find MAP estimates of parameters. The TensorFlow GradientDescentOptimizer is used.
logPrior	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.
minibatchSize	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
alpha	optional. Default 0.01. List of numeric values corresponding to the SGHMC momentum tuning constants (α in the original paper). One value should be given for each parameter in params, the names should correspond to those in params. Alternatively specify a single float to specify that value for all parameters.
L	optional. Default 5L. Integer specifying the trajectory parameter of the simulation, as defined in the main reference.
nIters	optional. Default 10^4L. Integer specifying number of iterations to perform.
nItersOpt	optional. Default 10^4L. Integer specifying number of iterations of initial optimization to perform.
verbose	optional. Default TRUE. Boolean specifying whether to print algorithm progress
seed	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

Returns list of arrays for each parameter containing the MCMC chain. Dimension of the form (nIters,paramDim1,paramDim2,...)

References

- Baker, J., Fearnhead, P., Fox, E. B., and Nemeth, C. (2017). Control variates for stochastic gradient MCMC. ArXiv preprint arXiv:1706.05439.
- Chen, T., Fox, E. B., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In ICML (pp. 1683-1691).

Examples

```
## Not run:
# Simulate from a Normal Distribution with uninformative prior
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-5)
optStepsize = 1e-1
output = sghmccv(logLik, dataset, params, stepsize, optStepsize)

## End(Not run)
```

`sghmccvSetup` *Create an sghmccv object*

Description

Creates an sghmccv (stochastic gradient Hamiltonian Monte Carlo with Control Variates) object which can be passed to `sgmcmcStep` to simulate from 1 step of sghmc, using a gradient estimate with control variates for the posterior defined by logLik and logPrior. This allows the user to code the loop themselves, as in many standard TensorFlow procedures (such as optimization). Which means they do not need to store the chain at each iteration. This is useful when the full chain needs a lot of memory.

Usage

```
sghmccvSetup(logLik, dataset, params, stepsize, optStepsize, logPrior = NULL,
  minibatchSize = 0.01, alpha = 0.01, L = 5L, nItersOpt = 10^4L,
  verbose = TRUE, seed = NULL)
```

Arguments

`logLik`	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
`dataset`	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions

`params`	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
`stepsize`	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
`optStepsize`	numeric value specifying the stepsize for the optimization to find MAP estimates of parameters. The TensorFlow GradientDescentOptimizer is used.
`logPrior`	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.
`minibatchSize`	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
`alpha`	optional. Default 0.01. List of numeric values corresponding to the SGHMC momentum tuning constants (α in the original paper). One value should be given for each parameter in params, the names should correspond to those in params. Alternatively specify a single float to specify that value for all parameters.
`L`	optional. Default 5L. Integer specifying the trajectory parameter of the simulation, as defined in the main reference.
`nItersOpt`	optional. Default 10^4L. Integer specifying number of iterations of initial optimization to perform.
`verbose`	optional. Default TRUE. Boolean specifying whether to print algorithm progress
`seed`	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

The function returns an 'sghmccv' object, a type of sgmcmc object. Which is used to pass the required information about the current model to the `sgmcmcStep` function. The function `sgmcmcStep` runs one step of sghmc with a gradient estimate that uses control variates. Attributes of the sghmccv object you'll probably find most useful are:

params list of tf$Variables with the same names as the params list passed to `sghmccvSetup`. This is the object passed to the logLik and logPrior functions you declared to calculate the log posterior gradient estimate.

paramsOpt list of tf$Variables with the same names as the `params` list passed to `sghmccvSetup`. These variables are used to initially find MAP estimates and then store these optimal parameter estimates.

estLogPost a tensor that estimates the log posterior given the current placeholders and params.

logPostOptGrad list of `tf$Variables` with same names as `params`, this stores the full log posterior gradient at each MAP estimate after the initial optimization step.

Other attributes of the object are as follows:

N dataset size.

data dataset as passed to sghmccvSetup.

n minibatchSize as passed to sghmccvSetup.

placeholders list of tf$placeholder objects with the same names as dataset used to feed minibatches of data to sgmcmcStep. These are also the objects that gets fed to the dataset argument of the logLik and logPrior functions you declared.

stepsize list of stepsizes as passed to sghmccvSetup

alpha list of alpha tuning parameters as passed to sghmcSetup.

L integer trajectory parameter as passed to sghmcSetup.

dynamics a list of TensorFlow steps that are evaluated by sgmcmcStep.

estLogPostOpt a TensorFlow tensor relying on `paramsOpt` and `placeholders` which estimates the log posterior at the optimal parameters. Used in the initial optimization step.

fullLogPostOpt a TensorFlow tensor used in the calculation of the full log posterior gradient at the MAP estimates.

optimizer a TensorFlow optimizer object used to find the initial MAP estimates.

Examples

```
## Not run:
# Simulate from a Normal Distribution, unknown location and known scale with uninformative prior
# Run sgmcmc step by step and calculate estimate of location on the fly to reduce storage
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
optStepsize = 1e-1
sghmccv = sghmccvSetup(logLik, dataset, params, stepsize, optStepsize)
nIters = 10^4L
# Initialize location estimate
locEstimate = 0
# Initialise TensorFlow session
sess = initSess(sghmccv)
for ( i in 1:nIters ) {
    sgmcmcStep(sghmccv, sess)
    locEstimate = locEstimate + 1 / nIters * getParams(sghmccv, sess)$theta
}
# For more examples see vignettes

## End(Not run)
```

sghmcSetup *Create an sghmc object*

Description

Creates an sghmc (stochastic gradient Hamiltonian Monte Carlo) object which can be passed to sgmcmcStep to simulate from 1 step of SGLD for the posterior defined by logLik and logPrior. This allows the user to code the loop themselves, as in many standard TensorFlow procedures (such as optimization). Which means they do not need to store the chain at each iteration. This is useful when the full chain needs a lot of memory.

Usage

```
sghmcSetup(logLik, dataset, params, stepsize, logPrior = NULL,
  minibatchSize = 0.01, alpha = 0.01, L = 5L, seed = NULL)
```

Arguments

logLik	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
dataset	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
params	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
stepsize	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
logPrior	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.
minibatchSize	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
alpha	optional. Default 0.01. List of numeric values corresponding to the SGHMC momentum tuning constants (α in the original paper). One value should be given for each parameter in params, the names should correspond to those in params. Alternatively specify a single float to specify that value for all parameters.
L	optional. Default 5L. Integer specifying the trajectory parameter of the simulation, as defined in the main reference.
seed	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

The function returns an 'sghmc' object, which is used to pass the required information about the current model to the sgmcmcStep function. The function sgmcmcStep runs one step of sghmc. The sghmc object has the following attributes:

params list of tf$Variables with the same names as the params list passed to sghmcSetup. This is the object passed to the logLik and logPrior functions you declared to calculate the log posterior gradient estimate.

estLogPost a tensor that estimates the log posterior given the current placeholders and params.

N dataset size.

data dataset as passed to sghmcSetup.

n minibatchSize as passed to sghmcSetup.

placeholders list of tf$placeholder objects with the same names as dataset used to feed minibatches of data to sgmcmcStep. These objects get fed to the dataset argument of the logLik and logPrior functions you declared.

stepsize list of stepsizes as passed to sghmcSetup.

alpha list of alpha tuning parameters as passed to sghmcSetup.

L integer trajectory parameter as passed to sghmcSetup.

dynamics a list of TensorFlow steps that are evaluated by sgmcmcStep.

Examples

```
## Not run:
# Simulate from a Normal Distribution, unknown location and known scale with uninformative prior
# Run sgmcmc step by step and calculate estimate of location on the fly to reduce storage
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
sghmc = sghmcSetup(logLik, dataset, params, stepsize)
nIters = 10^4L
# Initialize location estimate
locEstimate = 0
# Initialise TensorFlow session
sess = initSess(sghmc)
for ( i in 1:nIters ) {
    sgmcmcStep(sghmc, sess)
    locEstimate = locEstimate + 1 / nIters * getParams(sghmc, sess)$theta
}
# For more examples see vignettes

## End(Not run)
```

sgld *Stochastic Gradient Langevin Dynamics*

Description

Simulates from the posterior defined by the functions logLik and logPrior using stochastic gradient Langevin Dynamics. The function uses TensorFlow, so needs TensorFlow for python installed.

Usage

```
sgld(logLik, dataset, params, stepsize, logPrior = NULL,
  minibatchSize = 0.01, nIters = 10^4L, verbose = TRUE, seed = NULL)
```

Arguments

logLik	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
dataset	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
params	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
stepsize	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
logPrior	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.
minibatchSize	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
nIters	optional. Default 10^4L. Integer specifying number of iterations to perform.
verbose	optional. Default TRUE. Boolean specifying whether to print algorithm progress
seed	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

Returns list of arrays for each parameter containing the MCMC chain. Dimension of the form (nIters,paramDim1,paramDim2,...)

References

- Welling, M., and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. ICML (pp. 681-688).

Examples

```
## Not run:
# Simulate from a Normal Distribution with uninformative prior
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
output = sgld(logLik, dataset, params, stepsize)
# For more examples see vignettes

## End(Not run)
```

`sgldcv` *Stochastic Gradient Langevin Dynamics with Control Variates*

Description

Simulates from the posterior defined by the functions logLik and logPrior using stochastic gradient Langevin Dynamics with an improved gradient estimate using Control Variates. The function uses TensorFlow, so needs TensorFlow for python installed.

Usage

```
sgldcv(logLik, dataset, params, stepsize, optStepsize, logPrior = NULL,
  minibatchSize = 0.01, nIters = 10^4L, nItersOpt = 10^4L,
  verbose = TRUE, seed = NULL)
```

Arguments

`logLik`	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
`dataset`	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
`params`	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions

stepsize	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
optStepsize	numeric value specifying the stepsize for the optimization to find MAP estimates of parameters. The TensorFlow GradientDescentOptimizer is used.
logPrior	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.
minibatchSize	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
nIters	optional. Default 10^4L. Integer specifying number of iterations to perform.
nItersOpt	optional. Default 10^4L. Integer specifying number of iterations of initial optimization to perform.
verbose	optional. Default TRUE. Boolean specifying whether to print algorithm progress
seed	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

Returns list of arrays for each parameter containing the MCMC chain. Dimension of the form (nIters,paramDim1,paramDim2,...)

References

- Baker, J., Fearnhead, P., Fox, E. B., and Nemeth, C. (2017). Control variates for stochastic gradient MCMC. ArXiv preprint arXiv:1706.05439.
- Welling, M., and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. ICML (pp. 681-688).

Examples

```
## Not run:
# Simulate from a Normal Distribution with uninformative prior
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
optStepsize = 1e-1
output = sgldcv(logLik, dataset, params, stepsize, optStepsize)

## End(Not run)
```

sgldcvSetup *Create an sgldcv object*

Description

Creates an sgldcv (stochastic gradient Langevin Dynamics with Control Variates) object which can be passed to sgmcmcStep to simulate from 1 step of sgld, using a gradient estimate with control variates for the posterior defined by logLik and logPrior. This allows the user to code the loop themselves, as in many standard TensorFlow procedures (such as optimization). Which means they do not need to store the chain at each iteration. This is useful when the full chain needs a lot of memory.

Usage

```
sgldcvSetup(logLik, dataset, params, stepsize, optStepsize, logPrior = NULL,
  minibatchSize = 0.01, nItersOpt = 10^4L, verbose = TRUE, seed = NULL)
```

Arguments

logLik	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
dataset	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
params	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
stepsize	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
optStepsize	numeric value specifying the stepsize for the optimization to find MAP estimates of parameters. The TensorFlow GradientDescentOptimizer is used.
logPrior	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.
minibatchSize	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
nItersOpt	optional. Default 10^4L. Integer specifying number of iterations of initial optimization to perform.
verbose	optional. Default TRUE. Boolean specifying whether to print algorithm progress
seed	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

The function returns an 'sgldcv' object, a type of sgmcmc object. Which is used to pass the required information about the current model to the sgmcmcStep function. The function sgmcmcStep runs one step of sgld with a gradient estimate that uses control variates. Attributes of the sgldcv object you'll probably find most useful are:

params list of tf$Variables with the same names as the params list passed to sgldcvSetup. This is the object passed to the logLik and logPrior functions you declared to calculate the log posterior gradient estimate.

paramsOpt list of tf$Variables with the same names as the params list passed to sgldcvSetup. These variables are used to initially find MAP estimates and then store these optimal parameter estimates.

estLogPost a tensor that estimates the log posterior given the current placeholders and params.

logPostOptGrad list of tf$Variables with same names as params, this stores the full log posterior gradient at each MAP estimate after the initial optimization step.

Other attributes of the object are as follows:

N dataset size.

data dataset as passed to sgldcvSetup.

n minibatchSize as passed to sgldcvSetup.

placeholders list of tf$placeholder objects with the same names as dataset used to feed minibatches of data to sgmcmcStep. These are also the objects that gets fed to the dataset argument of the logLik and logPrior functions you declared.

stepsize list of stepsizes as passed to sgldcvSetup

dynamics a list of TensorFlow steps that are evaluated by sgmcmcStep.

estLogPostOpt a TensorFlow tensor relying on paramsOpt and placeholders which estimates the log posterior at the optimal parameters. Used in the initial optimization step.

fullLogPostOpt a TensorFlow tensor used in the calculation of the full log posterior gradient at the MAP estimates.

optimizer a TensorFlow optimizer object used to find the initial MAP estimates.

Examples

```
## Not run:
# Simulate from a Normal Distribution, unknown location and known scale with uninformative prior
# Run sgmcmc step by step and calculate estimate of location on the fly to reduce storage
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
optStepsize = 1e-1
sgldcv = sgldcvSetup(logLik, dataset, params, stepsize, optStepsize)
nIters = 10^4L
```

```
# Initialize location estimate
locEstimate = 0
# Initialise TensorFlow session
sess = initSess(sgldcv)
for ( i in 1:nIters ) {
    sgmcmcStep(sgldcv, sess)
    locEstimate = locEstimate + 1 / nIters * getParams(sgldcv, sess)$theta
}
# For more examples see vignettes

## End(Not run)
```

`sgldSetup` *Create an sgld object*

Description

Creates an sgld (stochastic gradient Langevin dynamics) object which can be passed to `sgmcmcStep` to simulate from 1 step of SGLD for the posterior defined by logLik and logPrior. This allows the user to code the loop themselves, as in many standard TensorFlow procedures (such as optimization). Which means they do not need to store the chain at each iteration. This is useful when the full chain needs a lot of memory.

Usage

```
sgldSetup(logLik, dataset, params, stepsize, logPrior = NULL,
  minibatchSize = 0.01, seed = NULL)
```

Arguments

`logLik`	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
`dataset`	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
`params`	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
`stepsize`	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
`logPrior`	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.

`minibatchSize` optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).

`seed` optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

The function returns an 'sgld' object, which is used to pass the required information about the current model to the `sgmcmcStep` function. The function `sgmcmcStep` runs one step of sgld. The sgld object has the following attributes:

params list of tf$Variables with the same names as the params list passed to `sgldSetup`. This is the object passed to the logLik and logPrior functions you declared to calculate the log posterior gradient estimate.

estLogPost a tensor that estimates the log posterior given the current placeholders and params (the placeholders holds the minibatches of data).

N dataset size.

data dataset as passed to `sgldSetup`.

n minibatchSize as passed to `sgldSetup`.

placeholders list of tf$placeholder objects with the same names as dataset used to feed minibatches of data to `sgmcmcStep`. These are the objects that get fed to the dataset argument of the logLik and logPrior functions you declared.

stepsize list of stepsizes as passed to `sgldSetup`.

dynamics a list of TensorFlow steps that are evaluated by `sgmcmcStep`.

Examples

```
## Not run:
# Simulate from a Normal Distribution, unknown location and known scale with uninformative prior
# Run sgmcmc step by step and calculate estimate of location on the fly to reduce storage
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
sgld = sgldSetup(logLik, dataset, params, stepsize)
nIters = 10^4L
# Initialize location estimate
locEstimate = 0
# Initialise TensorFlow session
sess = initSess(sgld)
for ( i in 1:nIters ) {
    sgmcmcStep(sgld, sess)
    locEstimate = locEstimate + 1 / nIters * getParams(sgld, sess)$theta
}
# For more examples see vignettes
```

```
## End(Not run)
```

`sgmcmc` *sgmcmc: A package for stochastic gradient MCMC*

Description

The sgmcmc package implements some of the most popular stochastic gradient MCMC methods including SGLD, SGHMC, SGNHT. It also implements control variates as a way to increase the efficiency of these methods. The algorithms are implemented using TensorFlow which means no gradients need to be specified by the user as these are calculated automatically. It also means the algorithms are efficient.

sgmcmc functions

The main functions of the package are sgld, sghmc and sgnht which implement the methods stochastic gradient Langevin dynamics, stochastic gradient Hamiltonian Monte Carlo and stochastic gradient Nose-Hoover Thermostat respectively. Also included are control variate versions of these algorithms, which uses control variates to increase their efficiency. These are the functions sgldcv, sghmccv and sgnhtcv.

References

Baker, J., Fearnhead, P., Fox, E. B., & Nemeth, C. (2017) control variates for stochastic gradient Langevin dynamics. Preprint.

Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. ICML (pp. 681-688).

Chen, T., Fox, E. B., & Guestrin, C. (2014). stochastic gradient Hamiltonian Monte Carlo. In ICML (pp. 1683-1691).

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., & Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. NIPS (pp. 3203-3211).

`sgmcmcStep` *Single step of sgmcmc*

Description

Update parameters by performing a single sgmcmc step with dynamics as defined in the sgmcmc object. This can be used to perform sgmcmc steps inside a loop as in standard TensorFlow optimization procedures. This is useful when high dimensional chains cannot fit into memory.

Usage

```
sgmcmcStep(sgmcmc, sess)
```

Arguments

sgmcmc	a stochastic gradient MCMC object returned by *Setup such as sgldSetup, sgldcvSetup etc.
sess	a TensorFlow session created using initSess

Examples

```
## Not run:
# Simulate from a Normal Distribution, unknown location and known scale with uninformative prior
# Run sgmcmc step by step and calculate estimate of location on the fly to reduce storage
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
sgld = sgldSetup(logLik, dataset, params, stepsize)
nIters = 10^4L
# Initialize location estimate
locEstimate = 0
# Initialise TensorFlow session
sess = initSess(sgld)
for ( i in 1:nIters ) {
    sgmcmcStep(sgld, sess)
    locEstimate = locEstimate + 1 / nIters * getParams(sgld, sess)$theta
}
# For more examples see vignettes

## End(Not run)
```

---

sgnht *Stochastic Gradient Nose Hoover Thermostat*

Description

Simulates from the posterior defined by the functions logLik and logPrior using stochastic gradient Nose Hoover Thermostat. The thermostat step needs a dot product to be calculated between two vectors. So when the algorithm uses parameters that are higher order than vectors (e.g. matrices and tensors), the thermostat step uses a tensor contraction. Tensor contraction is otherwise known as the inner product between two tensors.

Usage

```
sgnht(logLik, dataset, params, stepsize, logPrior = NULL,
  minibatchSize = 0.01, a = 0.01, nIters = 10^4L, verbose = TRUE,
  seed = NULL)
```

Arguments

logLik	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
dataset	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
params	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
stepsize	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
logPrior	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.
minibatchSize	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
a	optional. Default 0.01. List of numeric values corresponding to SGNHT diffusion factors (see Algorithm 2 of the original paper). One value should be given for each parameter in params, the names should correspond to those in params. Alternatively specify a single float to specify that value for all parameters.
nIters	optional. Default 10^4L. Integer specifying number of iterations to perform.
verbose	optional. Default TRUE. Boolean specifying whether to print algorithm progress
seed	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

Returns list of arrays for each parameter containing the MCMC chain. Dimension of the form (nIters,paramDim1,paramDim2,...)

References

- Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. NIPS (pp. 3203-3211).

Examples

```
## Not run:
# Simulate from a Normal Distribution with uninformative, improper prior
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
```

```
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 5e-6)
output = sgnht(logLik, dataset, params, stepsize)
# For more examples see vignettes

## End(Not run)
```

---

sgnhtcv *Stochastic Gradient Nose Hoover Thermostat with Control Variates*

Description

Simulates from the posterior defined by the functions logLik and logPrior using stochastic gradient Nose Hoover Thermostat with an improved gradient estimate that is calculated using control variates. The thermostat step needs a dot product to be calculated between two vectors. So when the algorithm uses parameters that are higher order than vectors (e.g. matrices and tensors), the thermostat step uses a tensor contraction. Tensor contraction is otherwise known as the inner product between two tensors.

Usage

```
sgnhtcv(logLik, dataset, params, stepsize, optStepsize, logPrior = NULL,
  minibatchSize = 0.01, a = 0.01, nIters = 10^4L, nItersOpt = 10^4L,
  verbose = TRUE, seed = NULL)
```

Arguments

logLik	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
dataset	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
params	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
stepsize	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
optStepsize	numeric value specifying the stepsize for the optimization to find MAP estimates of parameters. The TensorFlow GradientDescentOptimizer is used.
logPrior	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.

`minibatchSize`	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
`a`	optional. Default 0.01. List of numeric values corresponding to SGNHT diffusion factors (see Algorithm 2 of the original paper). One value should be given for each parameter in params, the names should correspond to those in params. Alternatively specify a single float to specify that value for all parameters.
`nIters`	optional. Default 10^4L. Integer specifying number of iterations to perform.
`nItersOpt`	optional. Default 10^4L. Integer specifying number of iterations of initial optimization to perform.
`verbose`	optional. Default TRUE. Boolean specifying whether to print algorithm progress
`seed`	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

Returns list of arrays for each parameter containing the MCMC chain. Dimension of the form (nIters,paramDim1,paramDim2,...). Names are the same as the params list.

References

- Baker, J., Fearnhead, P., Fox, E. B., and Nemeth, C. (2017). Control variates for stochastic gradient MCMC. ArXiv preprint arXiv:1706.05439.
- Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. NIPS (pp. 3203-3211).

Examples

```
## Not run:
# Simulate from a Normal Distribution with uninformative prior
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
optStepsize = 1e-1
output = sgnhtcv(logLik, dataset, params, stepsize, optStepsize)

## End(Not run)
```

sgnhtcvSetup *Create an sgnhtcv object*

Description

Creates an sgnhtcv (stochastic gradient Nose Hoover thermostat with Control Variates) object which can be passed to sgmcmcStep to simulate from 1 step of sgnht, using a gradient estimate with control variates for the posterior defined by logLik and logPrior. This allows the user to code the loop themselves, as in many standard TensorFlow procedures (such as optimization). Which means they do not need to store the chain at each iteration. This is useful when the full chain needs a lot of memory.

Usage

```
sgnhtcvSetup(logLik, dataset, params, stepsize, optStepsize, logPrior = NULL,
  minibatchSize = 0.01, a = 0.01, nItersOpt = 10^4L, verbose = TRUE,
  seed = NULL)
```

Arguments

logLik	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.
dataset	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
params	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
stepsize	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
optStepsize	numeric value specifying the stepsize for the optimization to find MAP estimates of parameters. The TensorFlow GradientDescentOptimizer is used.
logPrior	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.
minibatchSize	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
a	optional. Default 0.01. List of numeric values corresponding to SGNHT diffusion factors (see Algorithm 2 of the original paper). One value should be given for each parameter in params, the names should correspond to those in params. Alternatively specify a single float to specify that value for all parameters.

`nItersOpt`	optional. Default 10^4L. Integer specifying number of iterations of initial optimization to perform.
`verbose`	optional. Default TRUE. Boolean specifying whether to print algorithm progress
`seed`	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

The function returns an 'sgnhtcv' object, a type of sgmcmc object. Which is used to pass the required information about the current model to the `sgmcmcStep` function. The function `sgmcmcStep` runs one step of sgnht with a gradient estimate that uses control variates. Attributes of the sgnhtcv object you'll probably find most useful are:

params list of tf$Variables with the same names as the params list passed to `sgnhtcvSetup`. This is the object passed to the logLik and logPrior functions you declared to calculate the log posterior gradient estimate.

paramsOpt list of tf$Variables with the same names as the `params` list passed to `sgnhtcvSetup`. These variables are used to initially find MAP estimates and then store these optimal parameter estimates.

estLogPost a tensor relying on `params` and `placeholders`. This tensor estimates the log posterior given the current placeholders and params.

logPostOptGrad list of `tf$Variables` with same names as `params`, this stores the full log posterior gradient at each MAP estimate after the initial optimization step.

Other attributes of the object are as follows:

N dataset size.

data dataset as passed to `sgnhtcvSetup`.

n minibatchSize as passed to `sgnhtcvSetup`.

placeholders list of tf$placeholder objects with the same names as dataset used to feed minibatches of data to `sgmcmcStep`. These are also the objects that gets fed to the dataset argument of the logLik and logPrior functions you declared.

stepsize list of stepsizes as passed to `sgnhtcvSetup`

alpha list of alpha tuning parameters as passed to `sgnhtSetup`.

L integer trajectory parameter as passed to `sgnhtSetup`.

dynamics a list of TensorFlow steps that are evaluated by `sgmcmcStep`.

estLogPostOpt a TensorFlow tensor relying on `paramsOpt` and `placeholders` which estimates the log posterior at the optimal parameters. Used in the initial optimization step.

fullLogPostOpt a TensorFlow tensor used in the calculation of the full log posterior gradient at the MAP estimates.

optimizer a TensorFlow optimizer object used to find the initial MAP estimates.

Examples

```
## Not run:
# Simulate from a Normal Distribution, unknown location and known scale with uninformative prior
# Run sgmcmc step by step and calculate estimate of location on the fly to reduce storage
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
optStepsize = 1e-1
sgnhtcv = sgnhtcvSetup(logLik, dataset, params, stepsize, optStepsize)
nIters = 10^4L
# Initialize location estimate
locEstimate = 0
# Initialise TensorFlow session
sess = initSess(sgnhtcv)
for ( i in 1:nIters ) {
    sgmcmcStep(sgnhtcv, sess)
    locEstimate = locEstimate + 1 / nIters * getParams(sgnhtcv, sess)$theta
}
# For more examples see vignettes

## End(Not run)
```

---

sgnhtSetup *Create an sgnht object*

Description

Creates an sgnht (stochastic gradient Nose Hoover Thermostat) object which can be passed to sgmcmcStep to simulate from 1 step of SGNHT for the posterior defined by logLik and logPrior. This allows the user to code the loop themselves, as in many standard TensorFlow procedures (such as optimization). Which means they do not need to store the chain at each iteration. This is useful when the full chain needs a lot of memory.

Usage

```
sgnhtSetup(logLik, dataset, params, stepsize, logPrior = NULL,
  minibatchSize = 0.01, a = 0.01, seed = NULL)
```

Arguments

logLik	function which takes parameters and dataset (list of TensorFlow variables and placeholders respectively) as input. It should return a TensorFlow expression which defines the log likelihood of the model.

`dataset`	list of numeric R arrays which defines the datasets for the problem. The names in the list should correspond to those referred to in the logLik and logPrior functions
`params`	list of numeric R arrays which define the starting point of each parameter. The names in the list should correspond to those referred to in the logLik and logPrior functions
`stepsize`	list of numeric values corresponding to the SGLD stepsizes for each parameter The names in the list should correspond to those in params. Alternatively specify a single numeric value to use that stepsize for all parameters.
`logPrior`	optional. Default uninformative improper prior. Function which takes parameters (list of TensorFlow variables) as input. The function should return a TensorFlow tensor which defines the log prior of the model.
`minibatchSize`	optional. Default 0.01. Numeric or integer value that specifies amount of dataset to use at each iteration either as proportion of dataset size (if between 0 and 1) or actual magnitude (if an integer).
`a`	optional. Default 0.01. List of numeric values corresponding to SGNHT diffusion factors (see Algorithm 2 of the original paper). One value should be given for each parameter in params, the names should correspond to those in params. Alternatively specify a single float to specify that value for all parameters.
`seed`	optional. Default NULL. Numeric seed for random number generation. The default does not declare a seed for the TensorFlow session.

Value

The function returns an 'sgnht' object, which is used to pass the required information about the current model to the `sgmcmcStep` function. The function `sgmcmcStep` runs one step of sgnht. The sgnht object has the following attributes:

params list of tf$Variables with the same names as the params list passed to `sgnhtSetup`. This is the object passed to the logLik and logPrior functions you declared to calculate the log posterior gradient estimate.

estLogPost a tensor that estimates the log posterior given the current placeholders and params.

N dataset size.

data dataset as passed to `sgnhtSetup`.

n minibatchSize as passed to `sgnhtSetup`.

placeholders list of tf$placeholder objects with the same names as dataset used to feed minibatches of data to `sgmcmcStep`. This object gets fed to the dataset argument of the logLik and logPrior functions you declared.

stepsize list of stepsizes as passed to `sgnhtSetup`.

a list of a tuning parameters as passed to `sgnhtSetup`.

dynamics a list of TensorFlow steps that are evaluated by `sgmcmcStep`.

Examples

```
## Not run:
# Simulate from a Normal Distribution, unknown location and known scale with uninformative prior
# Run sgmcmc step by step and calculate estimate of location on the fly to reduce storage
dataset = list("x" = rnorm(1000))
params = list("theta" = 0)
logLik = function(params, dataset) {
    distn = tf$distributions$Normal(params$theta, 1)
    return(tf$reduce_sum(distn$log_prob(dataset$x)))
}
stepsize = list("theta" = 1e-4)
sgnht = sgnhtSetup(logLik, dataset, params, stepsize)
nIters = 10^4L
# Initialize location estimate
locEstimate = 0
# Initialise TensorFlow session
sess = initSess(sgnht)
for ( i in 1:nIters ) {
    sgmcmcStep(sgnht, sess)
    locEstimate = locEstimate + 1 / nIters * getParams(sgnht, sess)$theta
}
# For more examples see vignettes

## End(Not run)
```

Index