Package 'rgw'

October 10, 2016

Type Package

Title Goodman-Weare Affine-Invariant Sampling

Version 0.1.0

Date 2016-10-10

Author Adam Mantz

Maintainer Adam Mantz <amantz@slac.stanford.edu>

Description Implementation of the affine-invariant method of Goodman & Weare (2010) <DOI:10.2140/camcos.2010.5.65>, a method of producing Monte-Carlo samples from a target distribution.

License MIT + file LICENSE

LazyLoad yes

Imports parallel

URL https://github.com/abmantz/rgw

NeedsCompilation no

Repository CRAN

Date/Publication 2016-10-10 19:31:59

R topics documented:

rgw-package *Goodman-Weare Affine-Invariant Sampling*

Description

This package implements the affine-invariant method of Goodman & Weare (2010) <DOI:10.2140/camcos.2010.5.65>, a method of producing Monte-Carlo samples from a target distribution. The implementation is based on the description of the 'emcee' python package (implementing the same method) written by Forman-Mackey et al. (2012) <DOI:10.1086/670067>. See 'References' in the documentation of the GoodmanWeare function for full citation details.

Details

Package:	rgw
Type:	Package
Version:	0.1.0
Date:	2016-10-10
License:	MIT
LazyLoad:	yes

Author(s)

Adam Mantz <amantz@slac.stanford.edu>

GoodmanWeare *Goodman-Weare Affine-Invariant Sampling*

Description

Produces a Monte-Carlo Markov ensemble using the affine-invariant method of Goodman & Weare.

Usage

```
GoodmanWeare(ensemble, lnpost, Nsteps, current.lnP=NULL,
 mc.cores=getOption("mc.cores", 1L), ...)
```

Arguments

ensemble	an Nparam*Nwalkers array holding the initial state of the sampler. Nparam is the dimensionality of the parameter space and Nwalkers is the number of positions in the parameter space comprising the ensemble. Nwalkers must be even, and in practice should be *at minimum* twice Nparam.

lnpost	function taking a vector of parameter values as input, and returning the log-posterior density.
Nsteps	number of iterations to run the sampler.
current.lnP	vector holding the log-posterior value corresponding to the initial position of each walker. If not provided, this will be calculated internally.
mc.cores	number of cores to use for parallelism.
...	additional arguments to pass to lnpost.

Value

A list containing $ensemble: an array of the same dimensionality as ensemble, containing the position of the walkers after Nsteps iterations of the sampler; and $current.lnP: the log-posterior density for each walker.

Note

By default, the code will attempt to run in parallel (see the 'parallel' package). To prevent this, pass mc.cores=1.

Author(s)

Adam Mantz

References

Goodman, J. & Weare, J. (2010, Comm. App. Math. Comp. Sci., 5:6) <DOI:10.2140/camcos.2010.5.65>. This implementation is based on the description given by Foreman-Mackey et al. (2012, arXiv:1202.3665) <DOI:10.1086/670067>.

Examples

```
# In this example, we'll sample from a simple 2D Gaussian

# Define the log-posterior function
lnP = function(x) sum( dnorm(x, c(0,1), c(pi, exp(0.5)), log=TRUE) )

# Initialize an ensemble of 100 walkers
nwalk = 100
ensemble = array(dim=c(2, nwalk))
ensemble[1,] = rnorm(nwalk, 0, 0.1)
ensemble[2,] = rnorm(nwalk, 1, 0.1)

# Run for a bit
ens2 = GoodmanWeare(ensemble, lnP, 100, mc.cores=1)

# Plot the resulting ensemble
plot(t(ens2$ensemble))
# Compare to a direct draw from the posterior distribution
points(rnorm(nwalk, 0, pi), rnorm(nwalk, 1, exp(0.5)), col=2, pch=3)
```

GoodmanWeare.rem *Goodman-Weare Affine-Invariant Sampling*

Description

Produces a Monte-Carlo Markov ensemble using the affine-invariant method of Goodman & Weare, saving progress periodically.

Usage

```
GoodmanWeare.rem(post, lnpost, thin=1, mention.every=NA,
 save.every=NA, save.file=NA, ...)
```

Arguments

post	an Nparam*Nwalkers*Nsteps array. post[,,1] should hold the initial state of the sampler (see help for GoodmanWeare). Checkpoints and the return value will have the same shape, with subsequent layers post[,,i] holding the ensemble state at later iterations.
lnpost	function taking a vector of parameter values as input, and returning the log-posterior density.
thin	thinning factor for saving the results.
mention.every	print a message to the console every time this many iterations are completed.
save.every	save the accumulated Markov ensemble to disk every time this many iterations are completed.
save.file	filename for saving progress.
...	additional arguments to pass to GoodmanWeare or lnpost.

Value

An array of the same dimensionality as post, storing the position of the walkers in post[,,i] every thin iterations.

Note

By default, the code will attempt to run in parallel (see the 'parallel' package). To prevent this, pass mc.cores=1.

Author(s)

Adam Mantz

References

See help for GoodmanWeare.

Examples

```
# In this example, we'll sample from a simple 2D Gaussian.
# (This is the same example as used in GoodmanWeare.)

# Define the log-posterior function
lnP = function(x) sum( dnorm(x, c(0,1), c(pi, exp(0.5)), log=TRUE) )

# Initialize an ensemble of 100 walkers. We'll take 100 steps, saving the ensemble after each.
nwalk = 100
post = array(NA, dim=c(2, nwalk, 101))
post[1,,1] = rnorm(nwalk, 0, 0.1)
post[2,,1] = rnorm(nwalk, 1, 0.1)

# Run
post = GoodmanWeare.rem(post, lnP, mc.cores=1)

# Plot the final ensemble
plot(post[1,,101], post[2,,101])
# Look at the trace of each parameter for one of the walkers.
plot(post[1,1,])
plot(post[2,1,])
```

Index