Package ‘rhmc’

October 14, 2022

Type Package

Title Hamiltonian Monte Carlo

Version 1.0.0

Description Implements simple Hamiltonian Monte Carlo routines in R for sampling from any desired target distribution which is continuous and smooth. See Neal (2017) <arXiv:1701.02434> for further details on Hamiltonian Monte Carlo. Automatic parameter selection is not supported.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

NeedsCompilation no

Author Victhor Sartório [aut, cre, cph]

Maintainer Victhor Sartório <victhor@dme.ufrj.br>

Repository CRAN

Date/Publication 2018-10-28 22:20:03 UTC

R topics documented:

hamiltonian_dynamics *Hamiltonian Dynamics*

Description

Approximates Hamiltonian dynamics for some potential function and a L2-norm kinectic funcion, assuming H(q,p) = U(q) + K(p).

Usage

```
hamiltonian_dynamics(U, q, p, L, eps, m)
```

Arguments

U	Potential function of the system.
q	Initial position vector.
p	Initial momentum vector.
L	Number of steps.
eps	Size of each step.
m	Mass vector.

Value

A list with the position ‘q‘ and momentum ‘p‘ at the end of the trajectory.

Examples

```
U = function(x) exp(-0.5 * x^2) / sqrt(2 * pi)
hamiltonian_dynamics(U, -2, 0.8, 100, 0.1, 1)
hamiltonian_dynamics(U, -2, 0.85, 100, 0.1, 1)
```

hmc *Hamiltonian Monte Carlo*

Description

Performs Hamiltonian Monte Carlo for a desired target function.

Usage

```
hmc(f, init, numit, L, eps, mass)
```

Arguments

f	Minus log-density function of interest.
init	Initial point for the algorithm.
numit	Number of iterations.
L	Leapfrog parameter: number of steps.
eps	Leapfrog parameter: size of each step.
mass	Mass vector.

Value

A list with the chain with the samples of interest, the values of the log-density calculated at each step and the acceptance rate.

Examples

```
f = function(x) -dnorm(x, 20, 10, log = TRUE)
hmc(f, 19, 1000, 16, 0.3, 0.1)
```

---

num_grad *Numerical Gradient*

Description

Performs numerical differentiation of a function at a specific point. Uses some numerical tricks to always achieve a reliable, though not necessarily optimal, error.

Usage

```
num_grad(f, x)
```

Arguments

f	The function for which the gradient is desired.
x	The point at which the gradient should be approximated.

Value

The gradient of the function ‘f‘ at ‘x‘.

Examples

```
func = function(x) exp(-0.5 * x ^ 2) / sqrt(2 * pi)
grad = function(x) -x * exp(-0.5 * x ^ 2) / sqrt(2 * pi)
num_grad(func, -2)
abs(num_grad(func, -2) - grad(-2))
```

Index