Package 'PFoptim'

October 15, 2021

Type Package

Title Global Stochastic Optimization using a Particle Filter Algorithm

Version 1.0

Date 2021-07-27

Author Mathieu Gerber

Maintainer Mathieu Gerber <mathieu.gerber@bristol.ac.uk>

Description This package implements the G-PFSO (Global Particle Filter Stochastic Optimization) algorithm of Gerber and Douc (2021) for finding the global minimizer of a function defined through an expectation. Informally speaking, G-PFSO can be seen as a particle and derivative-free version of stochastic gradient methods.

License GPL (>= 2)

Imports Rcpp (>= 1.0.7), Rdpack

LinkingTo Rcpp

RdMacros Rdpack

RoxygenNote 7.1.1.9001

R topics documented:

PFoptim-package *The 'PFoptim' package: summary information*

Description

The package provides an implementation of the G-PFSO (Global Particle Filter Stochastic Optimization) algorithm of Gerber and Douc (2021) for finding the global minimizer of a function defined through an expectation. In addition, a function for implementing the SSP resampling algorithm(Gerber et al. 2019) and a function for implementating the Stratified resampling algorithm are also provided.

Author(s)

Mathieu Gerber

Maintainer: Mathieu Gerber <mathieu.gerber@bristol.ac.uk>

References

Gerber M, Chopin N, Whiteley N (2019). "Negative association, ordering and convergence of resampling methods." *The Annals of Statistics*, **47**(4), 2236–2260.

Gerber M, Douc R (2021). "A global stochastic optimization particle filter algorithm." *arXiv preprint arXiv:2007.04803*.

gpfso *Global Particle filter Stochastic Optimization*

Description

This function implements the G-PFSO (Global Particle Filter Stochastic Optimization) algorithm of Gerber and Douc (2021) for minimzing either the function $\theta \mapsto E[\mathrm{fn}(\theta, Y)]$ from i.i.d. realizations $y_1, ..., y_n$ of Y or the function $\theta \mapsto \sum_{i=1}^{n} \mathrm{fn}(\theta, y_i)$, where θ is a vector of dimension d.

Usage

```
gpfso(y, N, fn, init, numit, ..., resampling=c("SSP", "STRAT", "MULTI"), control= list())
```

Arguments

y	Either a vector of observations or a matrix of observations (the number of rows being the sample size).
N	Number of particles. The parameter N must be greater or equal to 2.

fn	function for a single observation. If theta is an N by d matrix and y is a matrix then fn(theta,y[i,]) must be a vector of length N. Similarly, if theta is an N by d matrix and y is a vector then fn(theta,y[i]) must be a vector of length N. If some rows of theta are outside the search space then the corresponding entries of the vector fn(theta,y[i,]) must be equal to Inf.
init	Function used to sample the initial particles such that init(N) is an N by d matrix (or alternatively a vector of length N if d=1).
...	Further arguments to be passed to fn.
numit	Number of iterations of the algorithm. If numit is not specified then G-PFSO estimates the minimizer of the function $E[\mathrm{fn}(\theta, Y)]$ (in which case the observations are processed sequentially and numit is equal to the sample size). If numit is specified then G-PFSO computes the minimizer of the function $\sum_{i=1}^{n} \mathrm{fn}(\theta, y_i)$.
resampling	Resampling algorithm to be used. Resamping should be either "SSP" (SSP resampling), "STRAT" (stratified resampling) or "MULTI" (multinomial resampling).
control	A list of control parameters. See details.

Value

A list with the following components:

B_par	Value of $\bar{\theta}^N_{\mathrm{numit}}$
T_par	Value of $\tilde{\theta}^N_{\mathrm{numit}}$
T_hist	Value of $\tilde{\theta}^N_t$ for $t = 1, ..., \mathrm{numit}$ (only if trace=TRUE)
ESS	Value of the effective sample for $t = 1, ..., \mathrm{numit}$ (only if trace=TRUE)

References

Gerber M, Douc R (2021). "A global stochastic optimization particle filter algorithm." *arXiv preprint arXiv:2007.04803*.

Examples

```
#Definition of fn
fn_toy<-function(theta, obs){
  test<-rep(0,nrow(theta))
  test[theta[,2]>0]<-1
  ll<-rep(-Inf,nrow(theta))
  ll[test==1]<-dnorm(obs,mean=theta[test==1,1], sd=theta[test==1,2],log=TRUE)
  return(-ll)
}
#Generate data y_1,...,y_n
n<-10000             #sample size
theta_star<-c(0,1)  #true parameter value
y<-rnorm(n,mean=theta_star[1], sd=theta_star[2])
d<-length(theta_star)
#Define init funciton to be used
```

```
pi0<-function(N){
    return(cbind(rnorm(N,0,5), rexp(N)))
}
##Example 1: Maximum likelihood estimation in the Gaussian model
##true value of the MLE
mle<-c(mean(y),sd(y))
## use gpfso to compute the MLE
Est<-gpfso(y, N=100, fn=fn_toy, init=pi0, numit=20000, control=list(trace=TRUE))
## print \bar{\theta}^N_{numit} and \tilde{\theta}^N_{numit}
print(Est$B_par)
print(Est$T_par)
##assess convergence
par(mfrow=c(1,2))
for(k in 1:2){
  plot(Est$T_hist[,k],type='l', xlab="iteration", ylab="approximation error")
  lines(cumsum(Est$T_hist[,k])/1:length(Est$T_hist[,k]),type='l', col='red')
  abline(h=mle[k])
}
##Example 2: Expected log-likelihood estimation in the Gaussian model
## Estimation of theta_star using gpfso
Est<-gpfso(y, N=100, fn=fn_toy, init=pi0, control=list(trace=TRUE))
## print \bar{\theta}^N_{numit} and \tilde{\theta}^N_{numit}
print(Est$B_par)
print(Est$T_par)
##assess convergence
par(mfrow=c(1,2))
for(k in 1:2){
  plot(Est$T_hist[,k],type='l', xlab="iteration", ylab="approximation error")
  lines(cumsum(Est$T_hist[,k])/1:length(Est$T_hist[,k]),type='l', col='red')
  abline(h=theta_star[k])
}
```

`SSP_Resampler` *SSP resampling*

Description

This function implements the SSP resampling algorithm (Gerber et al. 2019).

Usage

```
SSP_Resampler(U,W)
```

Arguments

`W`	A vector of normalized weights.
`U`	A vector of points in (0,1) such that `length(U)=length(W)`.

Value

A vector of length N with elements in the set $\{1, ..., N\}$, with `N=length(U)=length(W)`.

References

Gerber M, Chopin N, Whiteley N (2019). "Negative association, ordering and convergence of resampling methods." *The Annals of Statistics*, **47**(4), 2236–2260.

Examples

```
N<-100
W<-rbeta(N,0.5,2)
W<-W/sum(W)
J<-SSP_Resampler(runif(N),W)
```

Stratified_Resampler *Stratified resampling*

Description

This function implements the stratified resampling algorithm descibed see e.g. in Section 9.6 of Chopin and Papaspiliopoulos (2020)

Usage

```
Stratified_Resampler(U,W)
```

Arguments

`W`	A vector of normalized weights.
`U`	A vector of points in (0,1) such that `length(U)=length(W)`.

Value

A vector of length N with elements in the set $\{1, ..., N\}$, with `N=length(U)=length(W)`.

References

Chopin N, Papaspiliopoulos O (2020). *An introduction to sequential Monte Carlo*, volume 4. Springer.

Examples

```
N<-100
W<-rbeta(N,0.5,2)
W<-W/sum(W)
J<-Stratified_Resampler(runif(N),W)
```

Index