juliacon 2021

Algorithmic Variants of Nested Sampling

Saranjeet Kaur Bhogal

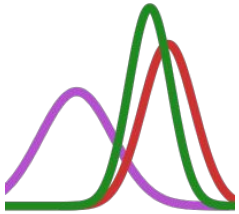
SaranjeetKaur

saranjeet-kaur-48ab769b

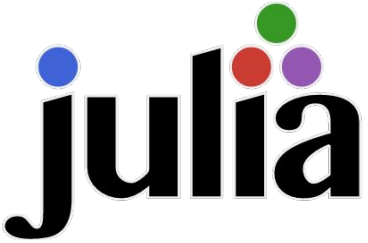
@qwertyquesting

Google Summer of Code 2020

Turing team



The Julia Language Organization



GSoC 2020 Mentors

Hong Ge, Cameron Pfiffer

Package Maintainer

Miles Lucas

Overview

- Nested sampling algorithm allows the user to generate samples from posterior distributions and estimate the model evidence
- NestedSamplers.jl package is inspired from the python package dynesty and its modular approach to nested sampling
- Julia's multiple dispatch makes it even more effective
- Three proposal algorithms: Random staggering, Slicing and Random slicing
- Code repository: NestedSamplers.jl
- My work product: GitHub Gist

Random Staggering

- Alternative to random walk proposal algorithm
- 10 ≤ Number of parameters ≤ 20
- Step size is exponentially adjusted to reach a target acceptance rate during each proposal, in addition to between proposals
- A new live point is proposed by randomly staggering away from an existing live point

```
"""
    Proposals.RStagger(;ratio=0.5, walks=25, scale=1)

Propose a new live point by random staggering away from an existing live point.
This differs from the random walk proposal in that the step size here is exponentially adjusted
to reach a target acceptance rate _during_ each proposal, in addition to _between_
proposals.

## Parameters
- `ratio` is the target acceptance ratio
- `walks` is the minimum number of steps to take
- `scale` is the proposal distribution scale, which will update _between_ proposals.
"""
@with_kw mutable struct RStagger <: AbstractProposal
    ratio = 0.5
    walks = 25
    scale = 1.0

    @assert 1 / walks ≤ ratio ≤ 1 "Target acceptance ratio must be between 1/`walks` and 1"
    @assert walks > 1 "Number of steps must be greater than 1"
    @assert scale ≥ 0 "Proposal scale must be non-negative"
end
```

Source code of Proposals.RStagger

Slicing

- Number of parameters > 20
- A standard Gibbs-like implementation where a single multivariate slice is a combination of univariate slices through each axis
- A new live point is proposed by a series of slices away from an existing live point

```
"""
    Proposals.Slice(;slices=5, scale=1)

Propose a new live point by a series of random slices away from an existing live point.
This is a standard _Gibbs-like_ implementation where a single multivariate slice is a c

## Parameters
- `slices` is the minimum number of slices
- `scale` is the proposal distribution scale, which will update _between_ proposals.
"""
@with_kw mutable struct Slice <: AbstractProposal
    slices = 5
    scale = 1.0

    @assert slices ≥ 1 "Number of slices must be greater than or equal to 1"
    @assert scale ≥ 0 "Proposal scale must be non-negative"
end

function (prop::Slice)(rng::AbstractRNG,
                       point::AbstractVector,
                       logl_star,
                       bounds::AbstractBoundingSpace,
                       loglike,
                       prior_transform;
                       kwargs...)
```

Source code of Proposals.Slice

Random Slicing

- Alternative to slicing proposal algorithm
- Number of parameters > 20
- Polychord nested sampling algorithm is roughly equivalent to this algorithm
- A standard random implementation where each slice is along a random direction based on the provided axes
- A new live point is proposed by a series of random slices away from an existing live point

```
"""
    Proposals.RSlice(;slices=5, scale=1)
Propose a new live point by a series of random slices away from an existing live point.
This is a standard _random_ implementation where each slice is along a random direction
## Parameters
- `slices` is the minimum number of slices
- `scale` is the proposal distribution scale, which will update _between_ proposals.
"""
@with_kw mutable struct RSlice <: AbstractProposal
    slices = 5
    scale = 1.0

    @assert slices ≥ 1 "Number of slices must be greater than or equal to 1"
    @assert scale ≥ 0 "Proposal scale must be non-negative"
end

function (prop::RSlice)(rng::AbstractRNG,
                        point::AbstractVector,
                        logl_star,
                        bounds::AbstractBoundingSpace,
                        loglike,
                        prior_transform;
                        kwargs...)
```

Source code of Proposals.RSlice

Further Work

- Providing more advanced proposals and bounds alternatives in the NestedSamplers.jl package
- Creating documents which illustrate the use of this sampler in various scenarios

Thank you!