Review on Statistical Tools and Samplers

Will Handley
<wh260@cam.ac.uk>

Royal Society University Research Fellow & Turing Fellow
Astrophysics Group, Cavendish Laboratory, University of Cambridge
Kavli Institute for Cosmology, Cambridge
Gonville & Caius College
willhandley.co.uk

26th November 2021

The Alan Turing Institute

The name of the game

- Many considerations in this talk applicable to Bayesian, Frequentist or "Monte Carlo"
- I acknowledge my Bayesian bias!
- In general consider a function $\mathcal{P}(\theta)$ over some d-dimensional space
- This function can be "forward calculated": given any location θ in space, can compute P
- Analytic pen-and-paper results assumed unavailable/impossible
- We wish to:
  - Explore the region(s) of high $\mathcal{P}$
  - Generate representative samples, either in this region, or in the tails
  - find the hypervolume under the curve $\int P(\theta) d\theta$
- Examples include
  - Generating samples from a Bayesian posterior distribution
  - Generating datasets for computing test statistics
  - Generating Monte Carlo events
  - Computing Bayesian evidences for model comparison
  - Computing cross sections

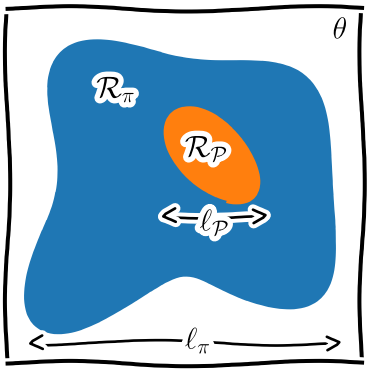
Notation

- Space of parameters θ
- Probability distribution/function $\mathcal{P}(\theta)$
- Region of valid parameters/support of function/prior/measure $\pi(\theta)$
  - Note that for any numerical method even if not specified there is an implicit measure imposed by floating point arithmetic

Why is this hard?

- In general, the region of significantly non-zero $\mathcal{P}$ (the typical set) is very "small"
- This problem gets exponentially worse the higher the dimensionality d becomes
- Quantify this with the Kullback-Liebler divergence between distributions $\mathcal{P}(\theta)$ and $\pi(\theta)$

$$\mathcal{D} = \int \log \mathcal{P}(\theta) \log \frac{\mathcal{P}(\theta)}{\pi(\theta)} d\theta \sim \log \frac{V_\pi}{V_\mathcal{P}}, \quad \frac{V_\pi}{V_\mathcal{P}} \sim \left(\frac{\ell_\pi}{\ell_\mathcal{P}}\right)^d$$



- Note that this is not a distance
- One can verify $\sim$ as being $=$ for top hat distributions over regions $R_\mathcal{P}$ and R_π with volumes $V_\mathcal{P}$ and V_π respectively:

$$\mathcal{P}(\theta) = \begin{cases} 1/V_\mathcal{P} & \theta \in R_\mathcal{P} \\ 0 & \text{otherwise} \end{cases} \qquad \pi(\theta) = \begin{cases} 1/V_\pi & \theta \in R_\pi \\ 0 & \text{otherwise} \end{cases}$$

- The integral smooths this over the distribution $\mathcal{P}$

How not to do it

- The worst way to explore/integrate a probability distribution/generate samples is to randomly sample the space using $\pi(\theta)$ [2012.09874]
- Gridding is also equivalently bad
- Whilst this works in low dimensions, if each parameter is confined within some fraction $f \sim \ell_{\mathcal{P}}/\ell_{\pi}$ of the space, then the volume fraction $\sim \mathcal{O}(f^d)$, or equivalently $\mathcal{D} \sim d \log f$
- Random sampling has an efficiency of $\boxed{\approx e^{-\mathcal{D}} \sim e^{-d \log f} = f^{-d}}$
- If you find that naive tail sampling is performant for e.g. importance sampling and unweighting, then your function likely has an unusual $\mathcal{D}$ scaling with d.
- Turning this around, you can use the inefficiency of random sampling to estimate $\mathcal{D}$
- Paper being released soon:
  "Exploring phase space with Nested Sampling" Handley, **Janßen**, Schumann & **Yallup**
  Also **Carragher** et al [2101.00428]

Why do sampling

- Instead of randomly sampling the space, samples $\theta \sim \mathcal{P}$ drawn from the distribution exponentially concentrate around the significantly non-zero region of $\mathcal{P}$
- This represents an optimal compression of the space
- If you have generated a set of samples drawn from a distribution $S = \{\theta_i : i = 1 \cdots N, \theta_i \sim \mathcal{P}\}$, then one can compute integrals

$$\langle f(\theta) \rangle_{\mathcal{P}} = \int f(\theta)\mathcal{P}(\theta)d\theta \approx \sum_{\theta_i \in S} f(\theta_i)$$

- Typically this is done using weighted samples $S = \{(w_i, \theta_i) : i = 1 \cdots N, \theta_i \sim \mathcal{P}\}$

$$\langle f(\theta) \rangle_{\mathcal{P}} = \int f(\theta)\mathcal{P}(\theta)d\theta \approx \sum_{w_i, \theta_i \in S} w_i f(\theta_i)$$

Metropolis Hastings

- Turn the N-dimensional problem into a one-dimensional one.
- Pick start point θ_0.
- At step i:
  1. Propose a new point θ_{i+1} a small step away from θ_i
  2. If uphill $\mathcal{P}(\theta_{i+1}) > \mathcal{P}(\theta_i)$, make step...
  3. ...otherwise make step with probability $\alpha = \mathcal{P}(\theta_{i+1})/\mathcal{P}(\theta_i)$.
- Requires a proposal distribution $\mathcal{Q}(\theta_{i+1}|\theta_i)$
- In general case where $\mathcal{Q}$ is not symmetric, need acceptance ratio:

$$\alpha = \frac{\mathcal{P}(\theta_{i+1})\mathcal{Q}(\theta_i|\theta_{i+1})}{\mathcal{P}(\theta_i)\mathcal{Q}(\theta_{i+1}|\theta_i)}$$

TOOLS

Whilst many exist: `PyMC3`, `cobaya`, `MontePython`,..., in practice the algo is so simple, and the proposal distribution so problem-specific, it's usually relatively efficient to write your own.

Metropolis Hastings

Where can this go wrong?

- Burn in
  - It can take a while for the chain to equilibriate to the typical set
  - It is hard to diagnose burn in, particularly in high dimensions
- Multimodality
  - If the function has multiple separated peaks, despite mathematical guarantees of convergence it will take a Hubble time to move between modes.
- Correlated distributions
  - In practice the peak(s) of the distribution have nontrivial structure (e.g. narrow ridges)
  - Very hard to create a flexible enough proposal to accomodate all, and not strictly Markovian
- Phase transitions
  - A different kind of multi-modality, which can occur if the function is a "slab and spike"
  - Two regions – one high-volume V lower $\mathcal{P}$, the other low V high $\mathcal{P}$. Difficult to transition
- Poor parallelisation
  - In practice it is not well parallelised, since majority of time is spent in burn-in

Hamiltonian Monte-Carlo

- Key idea: Treat $\log L(\Theta)$ as a potential energy
- Guide walker under "force":

$$F(\Theta) = \nabla \log L(\Theta)$$

- Walker is naturally "guided" uphill
- Conserved quantities mean efficient acceptance ratios.
- Whilst the recieved wisdom is that this is "tuning parameter free", in practice the mass matrix has similar degrees of tuning unless the problem is naturally normalised (which physicists are generally quite good at doing anyway).

TOOLS

- `stan` is a fully fledged, mature programming language with HMC as a default sampler.
- `TensorFlow` and `PyTorch` packages exist, although leader has not emerged.

Ensemble sampling

- Instead of one walker, evolve a set of n walkers.
- Can use information present in ensemble to guide proposals.
- Generally tuning parameter free
- Struggles with multimodal distributions
- Strive to be affine invariant

TOOLS

- `emcee`: The MCMC Hammer [1202.3665]
- `zeus`: Ensemble slice sampling [2002.06212]

The fundamental issue with all of the above

- They can't integrate functions over the space

$$\begin{aligned} Z &= P(D|M) \\ &= \int P(D|\Theta, M)P(\Theta|M)d\Theta \\ &= \langle L \rangle_{\pi} \end{aligned}$$

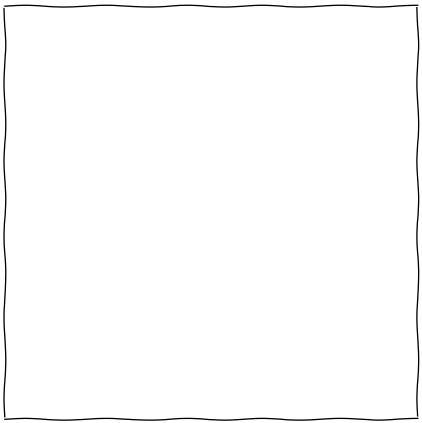
- MCMC fundamentally explores the posterior, and cannot average over the prior.
- Simulated annealing gives one possibility for "tricking" MCMC into computing evidences.
  - Inspired by thermodynamics.
  - Suffers from similar issues to MCMC.
  - Unclear how to choose correct annealing schedule

Nested sampling

- Nested sampling is a completely different way of sampling.
- Uses ensemble sampling to compress prior to posterior.
- Maintain a set S of n samples, which are sequentially updated:

  S_0: Generate n samples uniformly over the space (from the prior π).

  S_{n+1}: Delete the lowest likelihood sample in S_n, and replace it with a new uniform sample with higher likelihood
- Requires one to be able to sample uniformly within a region, subject to a *hard likelihood constraint*.

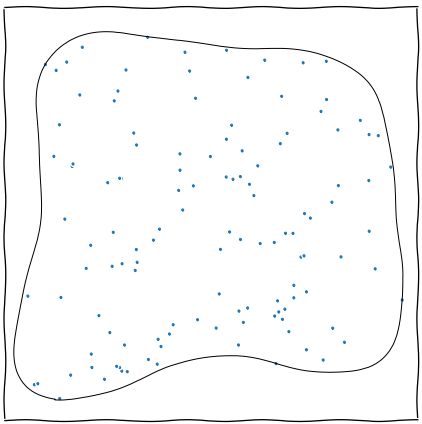


Nested sampling

- Nested sampling is a completely different way of sampling.
- Uses ensemble sampling to compress prior to posterior.
- Maintain a set S of n samples, which are sequentially updated:

  S_0: Generate n samples uniformly over the space (from the prior π).

  S_{n+1}: Delete the lowest likelihood sample in S_n, and replace it with a new uniform sample with higher likelihood
- Requires one to be able to sample uniformly within a region, subject to a *hard likelihood constraint*.

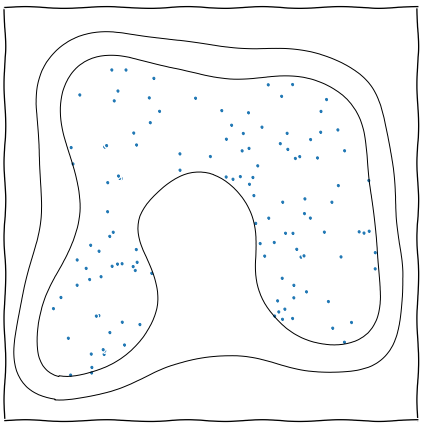


Nested sampling

- Nested sampling is a completely different way of sampling.
- Uses ensemble sampling to compress prior to posterior.
- Maintain a set S of n samples, which are sequentially updated:

  S_0: Generate n samples uniformly over the space (from the prior π).

  S_{n+1}: Delete the lowest likelihood sample in S_n, and replace it with a new uniform sample with higher likelihood
- Requires one to be able to sample uniformly within a region, subject to a *hard likelihood constraint*.

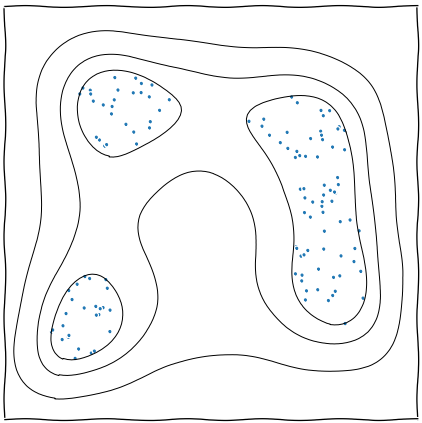


Nested sampling

- Nested sampling is a completely different way of sampling.
- Uses ensemble sampling to compress prior to posterior.
- Maintain a set S of n samples, which are sequentially updated:

  S_0: Generate n samples uniformly over the space (from the prior π).

  S_{n+1}: Delete the lowest likelihood sample in S_n, and replace it with a new uniform sample with higher likelihood
- Requires one to be able to sample uniformly within a region, subject to a *hard likelihood constraint*.

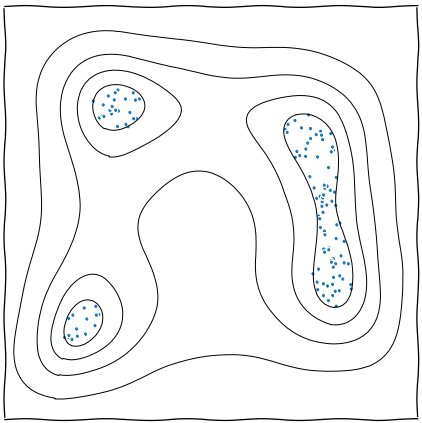


Nested sampling

- Nested sampling is a completely different way of sampling.
- Uses ensemble sampling to compress prior to posterior.
- Maintain a set S of n samples, which are sequentially updated:

  S_0: Generate n samples uniformly over the space (from the prior π).

  S_{n+1}: Delete the lowest likelihood sample in S_n, and replace it with a new uniform sample with higher likelihood
- Requires one to be able to sample uniformly within a region, subject to a *hard likelihood constraint*.

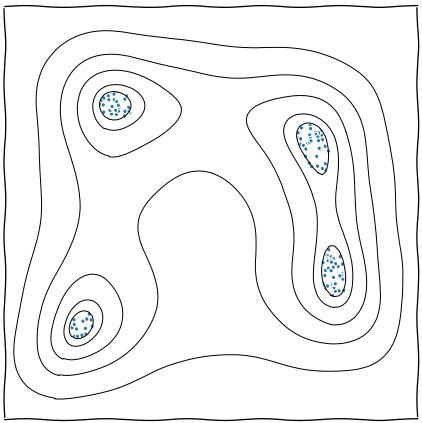


Nested sampling

- Nested sampling is a completely different way of sampling.
- Uses ensemble sampling to compress prior to posterior.
- Maintain a set S of n samples, which are sequentially updated:

  S_0: Generate n samples uniformly over the space (from the prior π).

  S_{n+1}: Delete the lowest likelihood sample in S_n, and replace it with a new uniform sample with higher likelihood
- Requires one to be able to sample uniformly within a region, subject to a *hard likelihood constraint*.

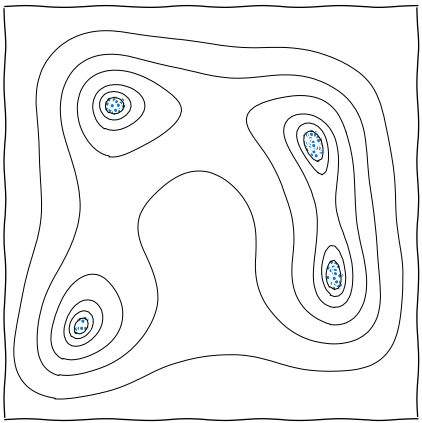


Nested sampling

- Nested sampling is a completely different way of sampling.
- Uses ensemble sampling to compress prior to posterior.
- Maintain a set S of n samples, which are sequentially updated:

  S_0: Generate n samples uniformly over the space (from the prior π).

  S_{n+1}: Delete the lowest likelihood sample in S_n, and replace it with a new uniform sample with higher likelihood
- Requires one to be able to sample uniformly within a region, subject to a *hard likelihood constraint*.

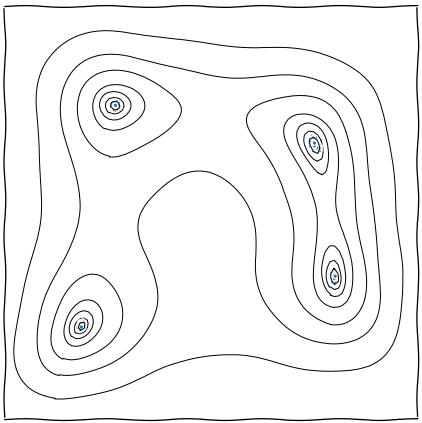


Nested sampling

- Nested sampling is a completely different way of sampling.
- Uses ensemble sampling to compress prior to posterior.
- Maintain a set S of n samples, which are sequentially updated:

  S_0: Generate n samples uniformly over the space (from the prior π).

  S_{n+1}: Delete the lowest likelihood sample in S_n, and replace it with a new uniform sample with higher likelihood
- Requires one to be able to sample uniformly within a region, subject to a *hard likelihood constraint*.

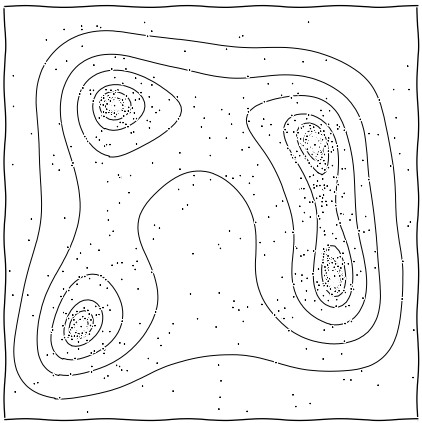


Nested sampling

- Nested sampling is a completely different way of sampling.
- Uses ensemble sampling to compress prior to posterior.
- Maintain a set S of n samples, which are sequentially updated:

  S_0: Generate n samples uniformly over the space (from the prior π).

  S_{n+1}: Delete the lowest likelihood sample in S_n, and replace it with a new uniform sample with higher likelihood
- Requires one to be able to sample uniformly within a region, subject to a *hard likelihood constraint*.

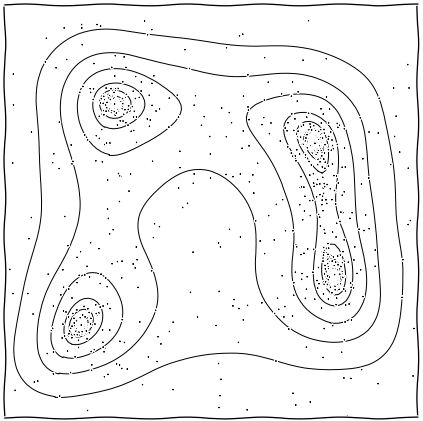


Nested sampling

- At the end, one is left with a set of discarded points
- These may be weighted to form posterior samples
- They can also be used to calculate the normalising constant
  - Critically, this is because nested sampling probabilistically estimates the volume of the parameter space

$$X_i \approx \left(\frac{n}{n+1}\right) X_{i-1} \quad \Rightarrow \quad X_i \approx \left(\frac{n}{n+1}\right)^i \approx e^{-i/n}$$

  - Whilst these are only statistical estimates, we know the error bar
  - Nested sampling is therefore a partition function calculator
- The evolving ensemble of live points allows algorithms

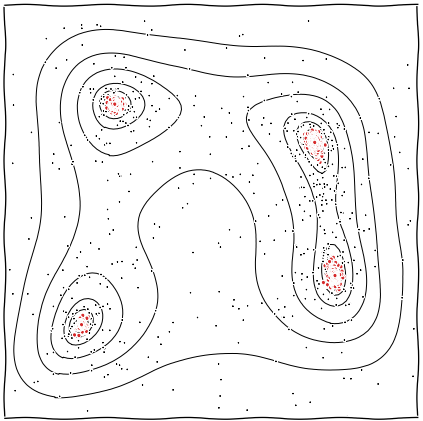


Nested sampling

- At the end, one is left with a set of discarded points
- These may be weighted to form posterior samples
- They can also be used to calculate the normalising constant
  - Critically, this is because nested sampling probabilistically estimates the volume of the parameter space

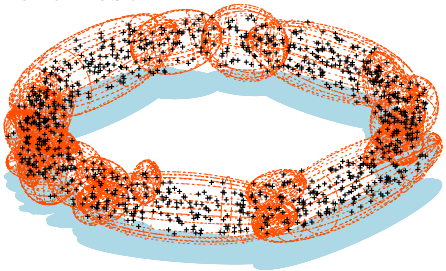
$$X_i \approx \left(\frac{n}{n+1}\right) X_{i-1} \quad \Rightarrow \quad X_i \approx \left(\frac{n}{n+1}\right)^i \approx e^{-i/n}$$

  - Whilst these are only statistical estimates, we know the error bar
  - Nested sampling is therefore a partition function calculator
- The evolving ensemble of live points allows algorithms

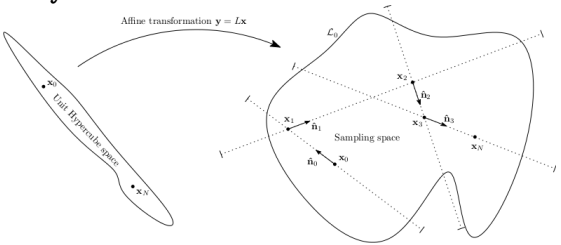


Implementations of Nested Sampling

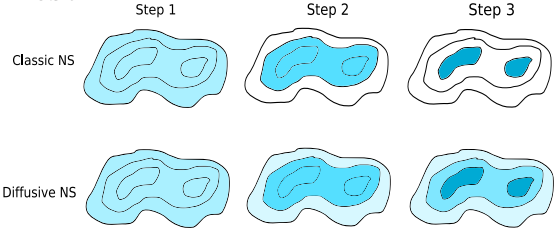
MultiNest



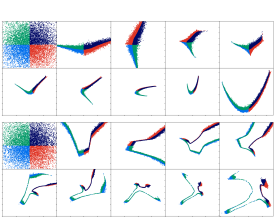
PolyChord

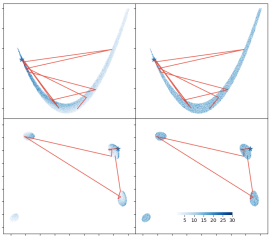


DNest



NeuralNest





Nested Sampling: Benefits and drawbacks

Relative to traditional numerical posterior samples (Metropolis Hastings, HMC, emcee), nested sampling:

- \+ Can calculate evidence (and therefore perform model comparison).
- \+ Can handle multi-modal distributions.
- \+ Requires little tuning for an a-priori unseen problem.
- \+ Highly parallelisable ($n_{\text{cores}} \sim n_{\text{live}} \gg 4$).
- − Slower than a well-tuned posterior sampler.
- − Run time is dependent on prior choice, and priors must be proper (some people view this as a feature rather than a bug).

Nested Sampling: a user's guide

1. Nested sampling is a likelihood scanner, rather than posterior explorer.
   - This means typically most of its time is spent on burn-in rather than posterior sampling
   - Changing the stopping criterion from 10^{-3} to 0.5 does little to speed up the run, but can make results very unreliable
2. The number of live points n_{live} is a resolution parameter.
   - Run time is linear in n_{live}, posterior and evidence accuracy goes as $\frac{1}{\sqrt{n_{\mathrm{live}}}}$.
   - Set low for exploratory runs $\sim \mathcal{O}(10)$ and increased to $\sim \mathcal{O}(1000)$ for production standard.
3. Most algorithms come with additional reliability parameter(s).
   - e.g. `MultiNest`: eff, `PolyChord`: n_{repeats}
   - These are parameters which have no gain if set too conservatively, but increase the reliability
   - Check that results do not degrade if you reduce them from defaults, otherwise increase.

Key tools for Nested Sampling

`anesthetic`	Nested sampling post processing [1905.04768]
`insertion`	cross-checks using order statistics [2006.03371] `github.com/williamjameshandley/anesthetic`
`nestcheck`	cross-checks using unthreaded runs [1804.06406] `github.com/ejhigson/nestcheck`
`MultiNest`	Ellipsoidal rejection sampling [0809.3437] `github.com/farhanferoz/MultiNest`
`PolyChord`	Python/C++/Fortran state of the art [1506.00171] `github.com/PolyChord/PolyChordLite`
`dynesty`	Python re-implementation of several codes [1904.02180] `github.com/joshspeagle/dynesty`

FAQs

- What was that awesome website?

  Full credit to Chi-feng for this incredible online demonstration tool
  chi-feng.github.io/mcmc-demo/

- How do you make your plots look hand-drawn?

  `import matplotlib.pyplot as plt; plt.xkcd()`