The scaling frontier of nested sampling

Will Handley
<wh260@cam.ac.uk>

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

1st July 2024

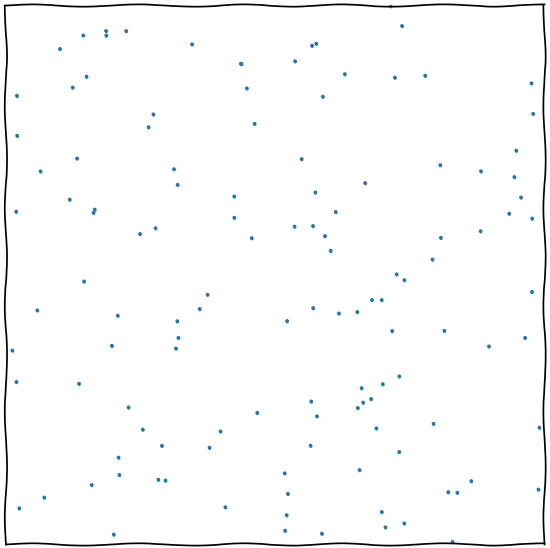

UNIVERSITY OF CAMBRIDGE



Recap: the nested sampling meta-algorithm

- Start with n_{live} random prior samples $\theta \sim \pi$.
- Delete lowest likelihood sample, and replace with a new one at higher value.
- The "live points" steadily contract around the peak(s) of the function.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n_{\text{live}}}$ of their volume.
- This is an exponential contraction, so

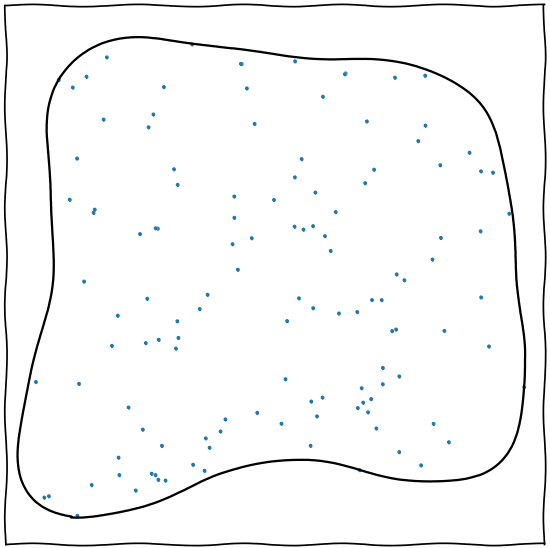
$$\mathcal{Z} = \int \mathcal{L}\pi d\theta \approx \sum_{i \in \text{dead}} \mathcal{L}_i \Delta X_i, \quad X_i = e^{-i/n_{\text{live}}}$$



Recap: the nested sampling meta-algorithm

- Start with n_{live} random prior samples $\theta \sim \pi$.
- Delete lowest likelihood sample, and replace with a new one at higher value.
- The "live points" steadily contract around the peak(s) of the function.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n_{\text{live}}}$ of their volume.
- This is an exponential contraction, so

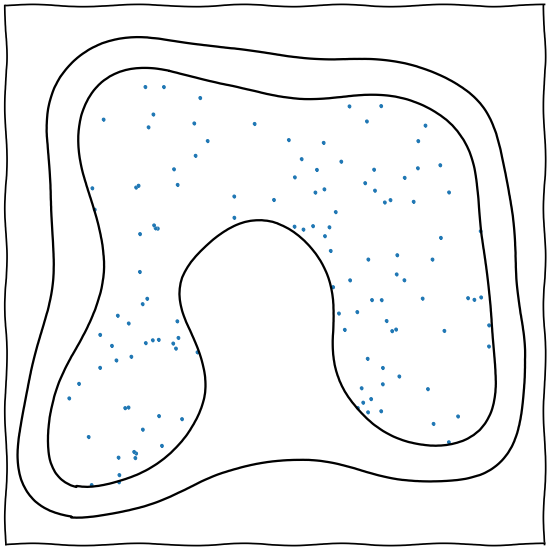
$$\mathcal{Z} = \int \mathcal{L}\pi \, d\theta \approx \sum_{i \in \text{dead}} \mathcal{L}_i \Delta X_i, \quad X_i = e^{-i/n_{\text{live}}}$$



Recap: the nested sampling meta-algorithm

- Start with n_{live} random prior samples $\theta \sim \pi$.
- Delete lowest likelihood sample, and replace with a new one at higher value.
- The "live points" steadily contract around the peak(s) of the function.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n_{\text{live}}}$ of their volume.
- This is an exponential contraction, so

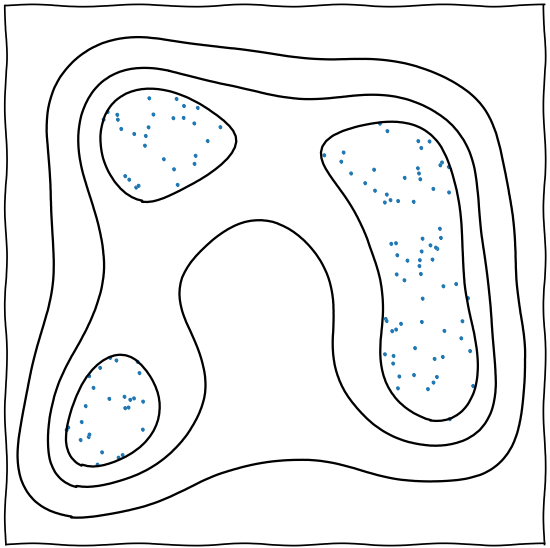
$$\mathcal{Z} = \int \mathcal{L}\pi \, d\theta \approx \sum_{i \in \text{dead}} \mathcal{L}_i \Delta X_i, \quad X_i = e^{-i/n_{\text{live}}}$$



Recap: the nested sampling meta-algorithm

- Start with n_{live} random prior samples $\theta \sim \pi$.
- Delete lowest likelihood sample, and replace with a new one at higher value.
- The "live points" steadily contract around the peak(s) of the function.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n_{\text{live}}}$ of their volume.
- This is an exponential contraction, so

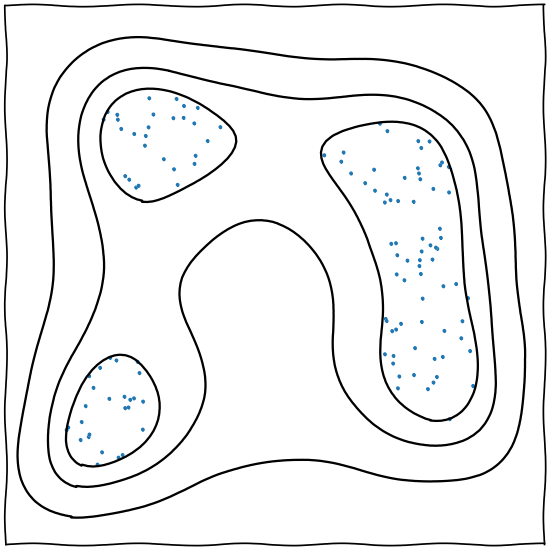
$$\mathcal{Z} = \int \mathcal{L}\pi d\theta \approx \sum_{i \in \text{dead}} \mathcal{L}_i \Delta X_i, \quad X_i = e^{-i/n_{\text{live}}}$$



Recap: the nested sampling meta-algorithm

- Start with n_{live} random prior samples $\theta \sim \pi$.
- Delete lowest likelihood sample, and replace with a new one at higher value.
- The "live points" steadily contract around the peak(s) of the function.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n_{\text{live}}} \pm \frac{1}{n_{\text{live}}}$ of their volume.
- This is an exponential contraction, so

$$\mathcal{Z} = \int \mathcal{L}\pi \, d\theta \approx \sum_{i \in \text{dead}} \mathcal{L}_i \Delta X_i, \quad X_i = e^{-(i \pm \sqrt{i})/n_{\text{live}}}$$



Kullback Leibler divergence (or entropy H or, information gain)

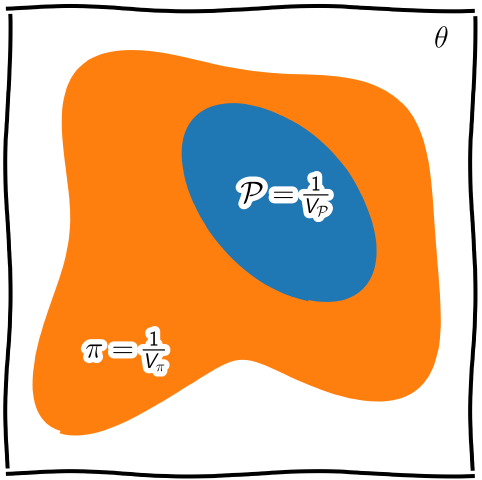
- The KL divergence between prior π and posterior $\mathcal{P}$ is is defined as:

$$\mathcal{D}_{\mathrm{KL}} = \left\langle \log \frac{\mathcal{P}}{\pi} \right\rangle_{\mathcal{P}} = \int \mathcal{P}(\theta) \log \frac{\mathcal{P}(\theta)}{\pi(\theta)} d\theta.$$

- Whilst not a distance, $\mathcal{D} = 0$ when $\mathcal{P} = \pi$.
- Occurs in the context of machine learning as an objective function for training functions.
- In Bayesian inference it can be understood as a log-ratio of "volumes":

$$\mathcal{D}_{\mathrm{KL}} \approx \log \frac{V_\pi}{V_\mathcal{P}}.$$

(this is exact for top-hat distributions).

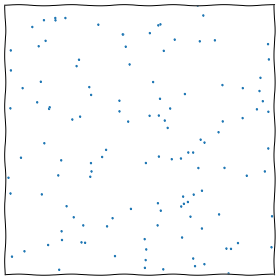


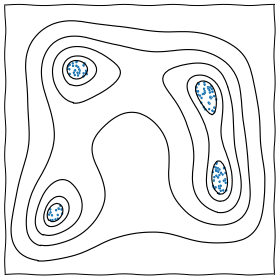
How fast in nested sampling?

Zixiao Hu
MSci 2023

$$T = T_{\mathcal{L}} \times \overbrace{\underbrace{n_{\text{live}} \times \mathcal{D}_{\text{KL}}}_{n_{\mathcal{L}}} \times f_{\text{sampler}}}^{n_{\text{dead}}},$$

- NS compresses the volume X of the live points from prior $X = 1$ to posterior $X = e^{-\mathcal{D}_{\text{KL}}}$.
- It does this exponentially so at iteration i, $X_i = e^{-i/n_{\text{live}}}$
- Need $e^{-n_{\text{dead}}/n_{\text{live}}} \sim e^{-\mathcal{D}_{\text{KL}}} \Rightarrow \boxed{n_{\text{dead}} \sim n_{\text{live}} \times \mathcal{D}_{\text{KL}}}$.
- It takes a f_{sampler} likelihood evaluations to generate a new live point, so $n_{\mathcal{L}} = n_{\text{dead}} \times f_{\text{sampler}}$
- It takes $T_{\mathcal{L}}$ seconds to call the likelihood, so $T = T_{\mathcal{L}} \times n_{\mathcal{L}}$





Further reading [2312.00294]

`aeons`: approximating the end of nested sampling, (Zixiao Hu et al)

How accurate is nested sampling?

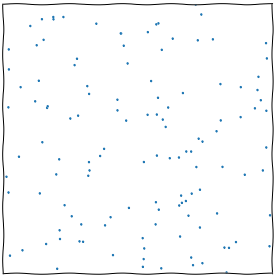
Lukas Hergt
PhD 2020

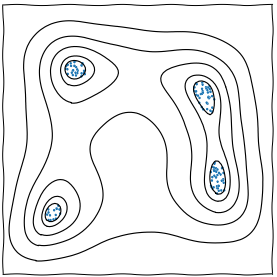
$$\boxed{\sigma(\log \mathcal{Z}) \approx \sqrt{\mathcal{D}_{\mathrm{KL}}/n_{\mathrm{live}}}}$$

- From the Occam's razor equation [2102.11511]:

$$\log \mathcal{Z} = \langle \log \mathcal{L} \rangle_{\mathcal{P}} - \mathcal{D}_{\mathrm{KL}}$$

- The dominant error is in the estimation of compression $\mathcal{D}_{\mathrm{KL}}$
- At each iteration, compress by $\exp(-\frac{1}{n_{\mathrm{live}}} \pm \frac{1}{n_{\mathrm{live}}})$
- After n_{dead} iterations, compressed by $e^{-\frac{n_{\mathrm{dead}}}{n_{\mathrm{live}}} \pm \frac{\sqrt{n_{\mathrm{dead}}}}{n_{\mathrm{live}}}} = e^{-\mathcal{D}_{\mathrm{KL}}}$
- As before $n_{\mathrm{dead}} \sim n_{\mathrm{live}} \times \mathcal{D}_{\mathrm{KL}}$
- $\sigma(\log \mathcal{Z}) = \sigma(\mathcal{D}_{\mathrm{KL}}) = \frac{\sqrt{n_{\mathrm{dead}}}}{n_{\mathrm{live}}} = \sqrt{\frac{\mathcal{D}_{\mathrm{KL}}}{n_{\mathrm{live}}}}$





The scaling frontier of nested sampling

How fast in nested sampling?

$$T = T_{\mathcal{L}} \times n_{\text{live}} \times \mathcal{D}_{\text{KL}} \times f_{\text{sampler}}$$

How accurate is nested sampling?

$$\sigma(\log \mathcal{Z}) \approx \sqrt{\mathcal{D}_{\text{KL}}/n_{\text{live}}}$$

in d dimensional parameter space:

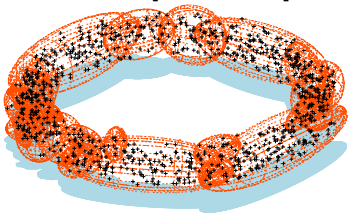
- $T_{\mathcal{L}}$: likelihood eval time $\sim \mathcal{O}(d)$
- n_{live}: number of live points $\sim \mathcal{O}(d)$
- $\mathcal{D}_{\text{KL}}$: KL divergence from prior to posterior $\approx \log V_\pi / V_{\mathcal{P}}$ $\sim \mathcal{O}(d)$
- f_{sampler}: efficiency of point generation region$\sim \mathcal{O}(e^{d/d_0})$ or path$\sim \mathcal{O}(d)$

- Algorithmically improving f_{sampler} is only a fraction of the story!
- $\mathcal{D}_{\text{KL}}$ appears twice, so improvements here are quadratically important.
- Gradients give you d more information.

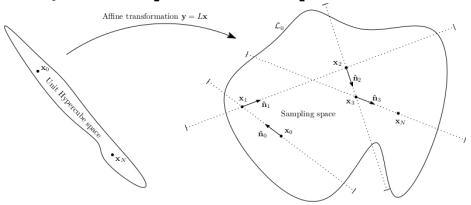
- $T \sim \mathcal{O}(d^4)$ whilst polynomial is far from ideal! – Let's unpack the caveats.

f_{sampler}: live-point generation efficiency

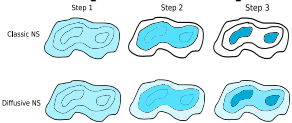
MultiNest [0809.3437]



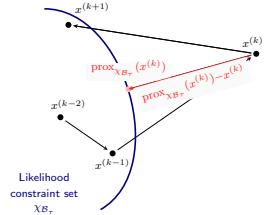
PolyChord [1506.00171]



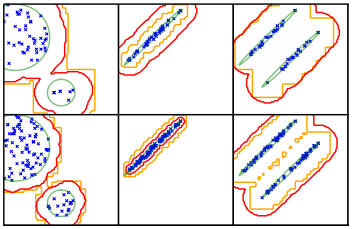
DNest [1606.03757]



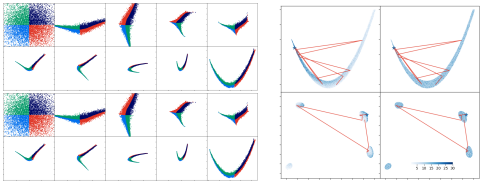
ProxNest [2106.03646]



UltraNest [2101.09604]



NeuralNest [1903.10860]



nautilus [2306.16923]

nessai [2102.11056]

nora [2305.19267]

jaxns [2012.15286],

dynesty [1904.02180]

nested_fit [1611.10189]

pymatnext [0906.3544]

f_{sampler}: live-point generation efficiency

- Broadly, most nested samplers can be split into how they create new live points.
- i.e. how they sample from the hard likelihood constraint $\{\theta \sim \pi : \mathcal{L}(\theta) > \mathcal{L}_*\}$.

Rejection samplers

- e.g. `MultiNest`, `UltraNest`.
- Constructs bounding region and draws many invalid points until $\mathcal{L}(\theta) > \mathcal{L}_*$.
- Efficient in low dimensions, exponentially inefficient $\sim \mathcal{O}(e^{d/d_0})$ in high $d > d_0 \sim 10$.

Chain-based samplers

- e.g. `PolyChord`, `ProxNest`.
- Run Markov chain starting at a live point, generating many valid (correlated) points.
- Linear $\sim \mathcal{O}(d)$ penalty in decorrelating new live point from the original seed point.

- for high dimensions, state of the art is linear
- Recent innovations in `UltraNest` allow self-tuning of efficiency for rejection sampling

Frontier

Can chain-based NS self-tune chain length to be sub-linear at run time?

n_{live}: the number of live points

There are two reasons one might increase the number of live points:

Multimodality

At each iteration, if a mode has a basin of attraction $< 1/n_{\text{live}}$ of the space you are more likely than not to miss it

- No reason to expect multimodality to have anything so simple as linear scaling with dimension.

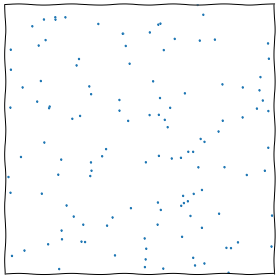
Accuracy

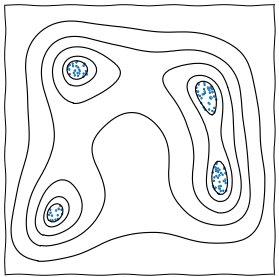
Evidence error $\sigma(\log \mathcal{Z}) = \sqrt{\mathcal{D}_{\text{KL}}/n_{\text{live}}}$

- So if $\mathcal{D}_{\text{KL}} \sim \mathcal{O}(d)$, then to maintain the same quality of inference have to scale n_{live} at the same rate.

Frontier

Can we make this "mode missing" argument more quantitative?





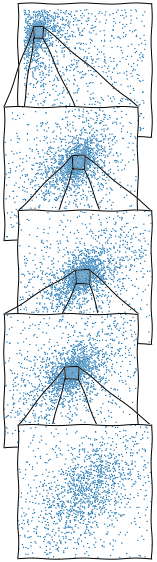
$T_{\mathcal{L}}$: the time to evaluate the likelihood

Acceleration/emulation

- Increasing emphasis of using machine learning/AI emulators to accelerate slow components of pipelines
- Nested sampling dead points are good at training emulators of likelihoods, due to their nose-to-tail sampling (Brewer's "Dead measure")
- `BAMBI` [1110.2997] does this at run-time to accelerate nested sampling

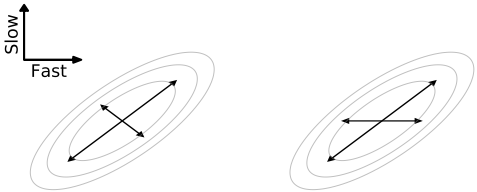
Frontier

Is BAMBI the best we can do?



Fast-slow hierarchies

- Success in cosmology where some parameters take longer than others
- reduces $T_{\mathcal{L}} \sim \mathcal{O}(d_{\text{slow}})$.



Frontier

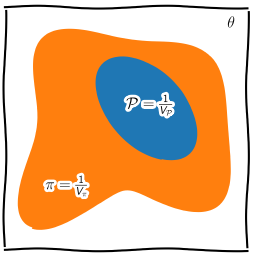
Can fast-slow hierarchies be better automated within algorithms?

$\mathcal{D}_{\mathrm{KL}}$: The Kullback Leibler divergence

- Naively $\mathcal{D}_{\mathrm{KL}} \sim \mathcal{O}(d)$, since if you have d *useful* parameters, you would expect to need to compress each of them, so each contribute linearly to the volume contribution
- From the Occam's razor equation however:

$$\log \mathcal{Z} = \langle \log \mathcal{L} \rangle_{\mathcal{P}} - \mathcal{D}_{\mathrm{KL}}$$

  a large $\mathcal{D}_{\mathrm{KL}}$ means a "bad" model
- Methods which explore models weighted by their evidence won't spend as much time in high $\mathcal{D}_{\mathrm{KL}}$ regions [1506.09024].
- There's also an intuition of the KL being "information", so increasing the number of parameters shouldn't extract more content.
- The KL divergence is a property of the model, so one would initially consider this to be something that the user has no control over...



$$\mathcal{D}_{\mathrm{KL}} = \log \frac{V_\pi}{V_{\mathcal{P}}}$$

$$\mathcal{D}_{\mathrm{KL}} = \left\langle \log \frac{\mathcal{P}}{\pi} \right\rangle_{\mathcal{P}}$$

Posterior repartitioning

- Almost all sampling algorithms (MH, HMC, SMC) are only sensitive to the product of likelihood and prior $\boxed{\mathcal{L} \times \pi}$.
- In practice this manifests as many codes implementing "prior terms" in the likelihood (or visa-versa) with no ill effect
- Nested sampling is unusual, in that one "samples from the prior π, subject to a hard constraint $\mathcal{L} > \mathcal{L}_*$.":

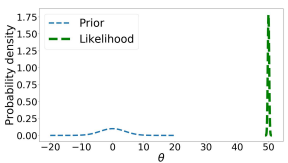
$$\{\theta \sim \pi : \mathcal{L}(\theta) > \mathcal{L}_*\}$$

- This separates the prior from the likelihood at an algorithmic level.

- One can therefore repartition the likelihood and prior $(\mathcal{L}, \pi) \to (\tilde{\mathcal{L}}, \tilde{\pi})$, providing

$$\boxed{\mathcal{L} \times \pi = \tilde{\mathcal{L}} \times \tilde{\pi}}$$

- This moves pieces between likelihood and prior that you have analytic control over.



(a) unrepresentative prior, $\theta_* = 50$ (b) estimated posterior by MultiNest, $\theta_* = 50$

- Chen, Feroz & Hobson [1908.04655] invented this to deal with misspecified priors

Posterior repartitioning for acceleration

Metha Prathaban
PhD

Key idea

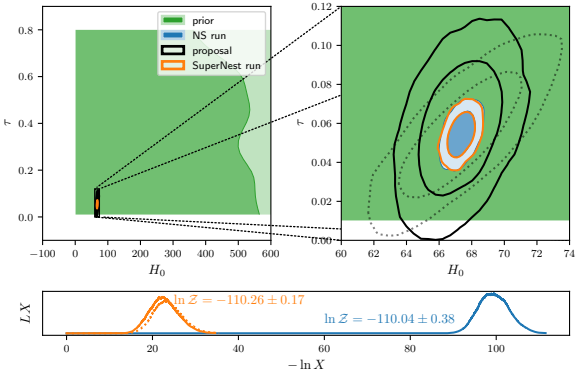
Whilst posteriors and evidences are invariant to repartitioning $\mathcal{L} \to \tilde{\mathcal{L}}$, the KL divergence is not.

- For constant "run quality" σ,

$$T = T_{\mathcal{L}} \times n_{\text{live}} \times \mathcal{D}_{\text{KL}} \times f_{\text{sampler}}, \qquad \sigma \approx \sqrt{\mathcal{D}_{\text{KL}}/n_{\text{live}}}$$

$$\Rightarrow \boxed{T = T_{\mathcal{L}} \times \sigma \times \mathcal{D}_{\text{KL}}^2 \times f_{\text{sampler}}}$$

  so if you can reduce the KL divergence, then quadratic gains to be made
- `SuperNest` [2212.01760]
- Ongoing work with Metha Prathaban & Harry Bevins.

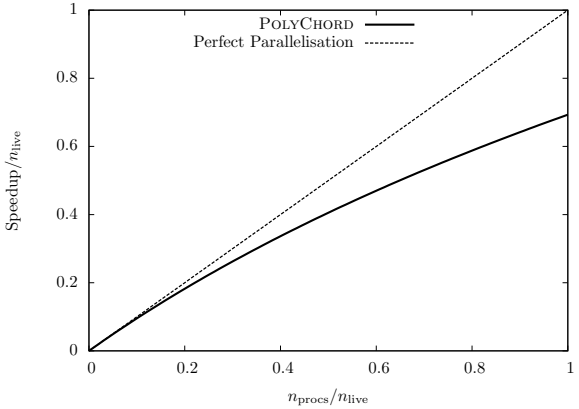


Frontier

Can we use posterior repartitioning to quadratically reduce the run time of nested sampling?

Parallelisation

- Nested sampling is well-suited to parallelisation, up to the number of live points, in contrast with MH/HMC
- I've known people run nested sampling on 10,000 CPUs
- This is in addition to any existing parallelisation in the likelihood
- Dynamic nested sampling very useful here adding machinery for throwing away multiple points [1704.03459]



Frontier

Are there any other parallelisation strategies, e.g. clustering/divide & conquer?

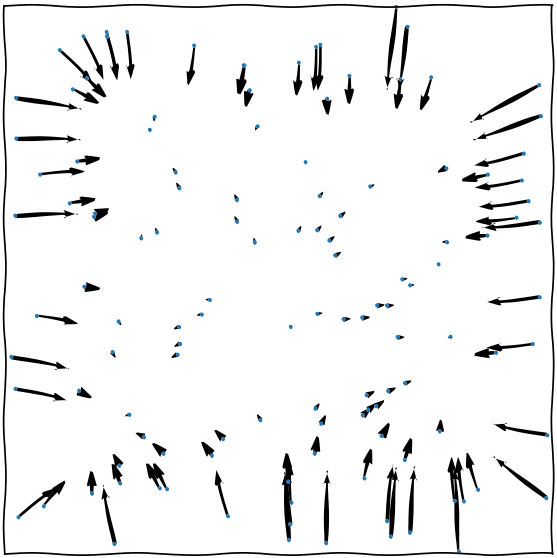
Nested sampling with gradients

Namu Kroupa
PhD (with Gábor Csányi)

- In principle, gradients give you $\sim \mathcal{O}(d)$ pieces of information to help with exploration
- With modern automatic differentiation codes (`JAX`, `PyTorch`, `TensorFlow`, `Enzyme` ...) these now come "for free".
- However nested sampling has unique challenges, so plugging in default (e.g. HMC/Langevin) algorithms does not work

Frontier

Can you come up with a nested sampling strategy that uses gradients and scales pass $d \sim \mathcal{O}(100)$?



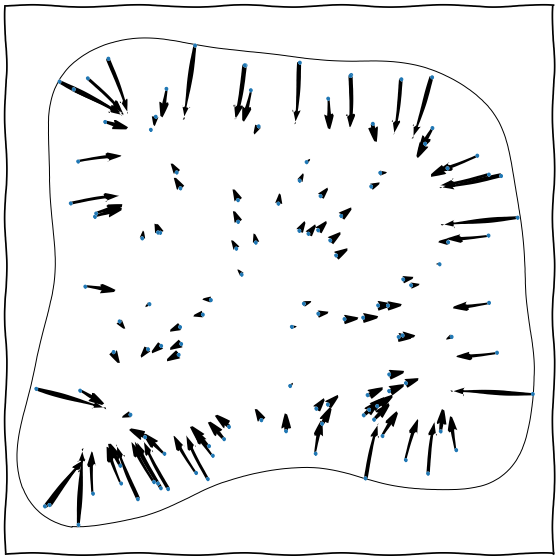
Nested sampling with gradients

Namu Kroupa
PhD (with Gábor Csányi)

- In principle, gradients give you $\sim \mathcal{O}(d)$ pieces of information to help with exploration
- With modern automatic differentiation codes (`JAX`, `PyTorch`, `TensorFlow`, `Enzyme` ...) these now come "for free".
- However nested sampling has unique challenges, so plugging in default (e.g. HMC/Langevin) algorithms does not work

Frontier

Can you come up with a nested sampling strategy that uses gradients and scales pass $d \sim \mathcal{O}(100)$?



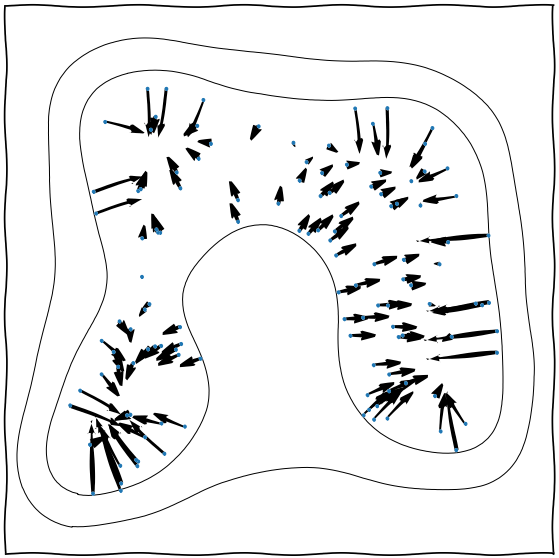
Nested sampling with gradients

Namu Kroupa
PhD (with Gábor Csányi)

- In principle, gradients give you $\sim \mathcal{O}(d)$ pieces of information to help with exploration
- With modern automatic differentiation codes (`JAX`, `PyTorch`, `TensorFlow`, `Enzyme` ...) these now come "for free".
- However nested sampling has unique challenges, so plugging in default (e.g. HMC/Langevin) algorithms does not work

Frontier

Can you come up with a nested sampling strategy that uses gradients and scales pass $d \sim \mathcal{O}(100)$?



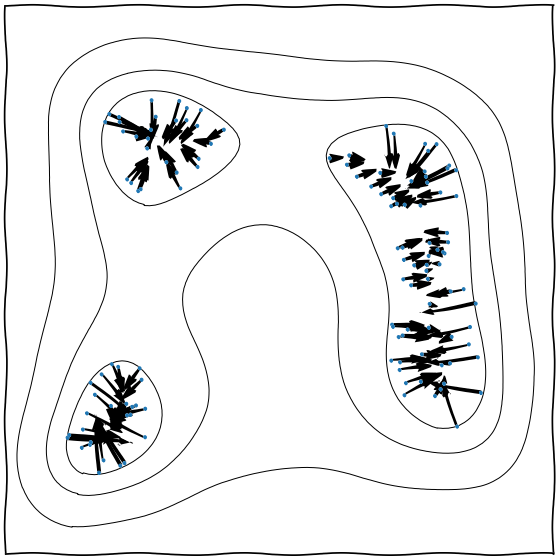
Nested sampling with gradients

Namu Kroupa
PhD (with Gábor Csányi)

- In principle, gradients give you $\sim \mathcal{O}(d)$ pieces of information to help with exploration
- With modern automatic differentiation codes (`JAX`, `PyTorch`, `TensorFlow`, `Enzyme` ...) these now come "for free".
- However nested sampling has unique challenges, so plugging in default (e.g. HMC/Langevin) algorithms does not work

Frontier

Can you come up with a nested sampling strategy that uses gradients and scales pass $d \sim \mathcal{O}(100)$?



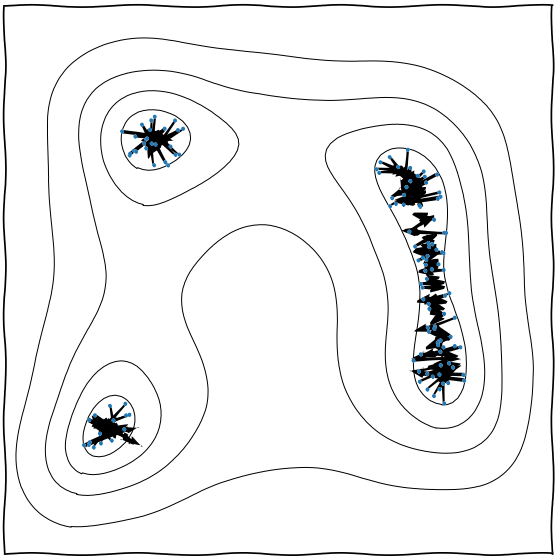
Nested sampling with gradients

Namu Kroupa
PhD (with Gábor Csányi)

- In principle, gradients give you $\sim \mathcal{O}(d)$ pieces of information to help with exploration
- With modern automatic differentiation codes (`JAX`, `PyTorch`, `TensorFlow`, `Enzyme` ...) these now come "for free".
- However nested sampling has unique challenges, so plugging in default (e.g. HMC/Langevin) algorithms does not work

Frontier

Can you come up with a nested sampling strategy that uses gradients and scales pass $d \sim \mathcal{O}(100)$?



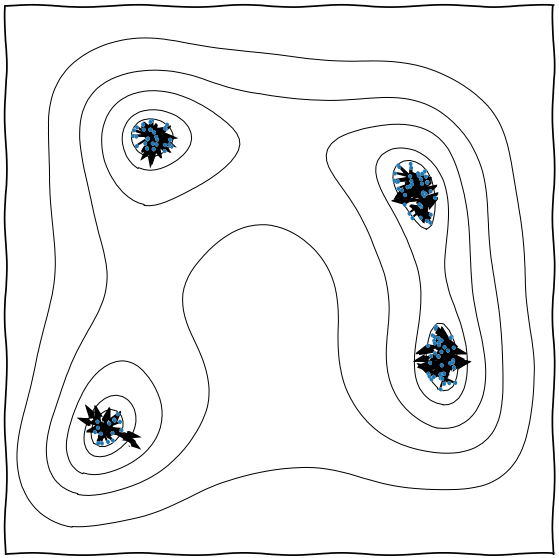
Nested sampling with gradients

Namu Kroupa
PhD (with Gábor Csányi)

- In principle, gradients give you $\sim \mathcal{O}(d)$ pieces of information to help with exploration
- With modern automatic differentiation codes (`JAX`, `PyTorch`, `TensorFlow`, `Enzyme` ...) these now come "for free".
- However nested sampling has unique challenges, so plugging in default (e.g. HMC/Langevin) algorithms does not work

Frontier

Can you come up with a nested sampling strategy that uses gradients and scales pass $d \sim \mathcal{O}(100)$?



Nested sampling with gradients

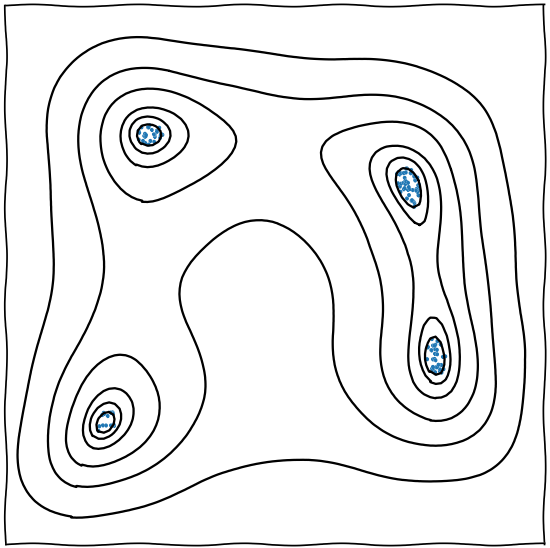
Namu Kroupa
PhD (with Gábor Csányi)

Betancourt	Hamiltonian constrained nested sampling [1005.0157].
Feroz	Galilean Nested Sampling [1312.5638].
Skilling	Galilean and Hamiltonian Monte Carlo [doi:10.3390/proceedings2019033019]
Speagle	Incorporated into `dynesty` [1904.02180].
Habeck	Habeck Demonic nested sampling – uses thermodynamic analogy to soften the hard boundary with a Maxwell daemon [doi:10.1063/1.4905971].
Baldock	Total Enthalpy HMC, incorporating momenta in a more HMC like way, but specialised to materials science [1710.11085].
Cai	ProxNest for high-dimensional convex imaging problems [2106.03646].
Lemos	Updated existing HMC/Galilean nested sampling to use differentiable programming (jax/torch) ICML 2024 [2312.03911]

- See also 2023 MIAPbP talk "Gradients and nested sampling: The present state of the art"
- Come talk to me afterwards for further detail on what I'm working on with Namu Kroupa.

Reversible nested sampling

- One could in principle reverse the direction of travel, and move outward from a peak.
- Need to know the initial volume $X_{\text{start}} \neq 1$
- e.g. if a Laplace estimation were good enough at the peak.
- The same trick in normal nested sampling would reduce volume estimation error and hence improve scaling.

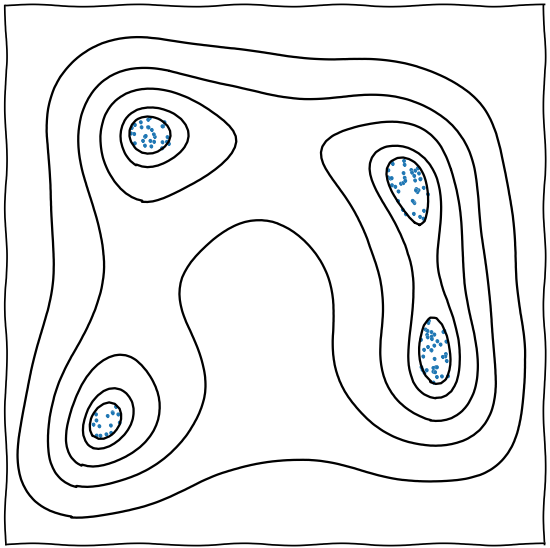


Frontier

- Can you come up with a reversible nested sampling algorithm?
- Can you reduce the volume estimation error by clamping the end point?

Reversible nested sampling

- One could in principle reverse the direction of travel, and move outward from a peak.
- Need to know the initial volume $X_{\text{start}} \neq 1$
- e.g. if a Laplace estimation were good enough at the peak.
- The same trick in normal nested sampling would reduce volume estimation error and hence improve scaling.

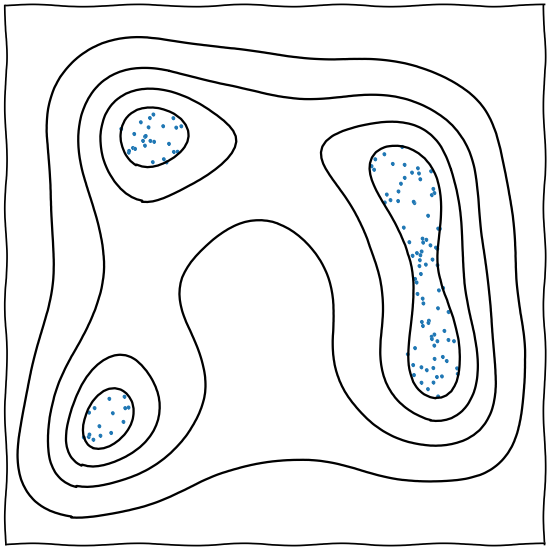


Frontier

- Can you come up with a reversible nested sampling algorithm?
- Can you reduce the volume estimation error by clamping the end point?

Reversible nested sampling

- One could in principle reverse the direction of travel, and move outward from a peak.
- Need to know the initial volume $X_{\text{start}} \neq 1$
- e.g. if a Laplace estimation were good enough at the peak.
- The same trick in normal nested sampling would reduce volume estimation error and hence improve scaling.

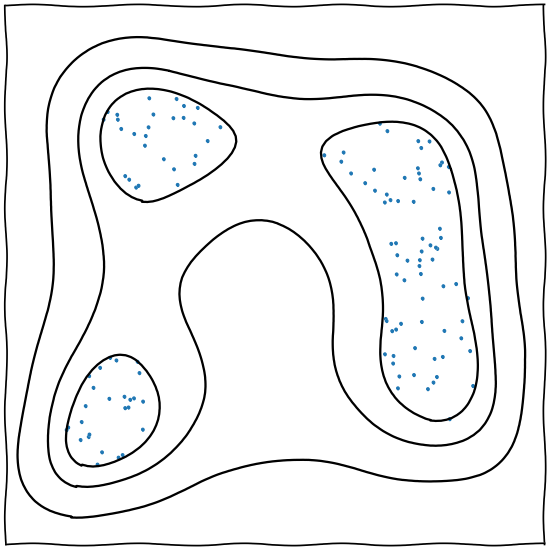


Frontier

- Can you come up with a reversible nested sampling algorithm?
- Can you reduce the volume estimation error by clamping the end point?

Reversible nested sampling

- One could in principle reverse the direction of travel, and move outward from a peak.
- Need to know the initial volume $X_{\text{start}} \neq 1$
- e.g. if a Laplace estimation were good enough at the peak.
- The same trick in normal nested sampling would reduce volume estimation error and hence improve scaling.

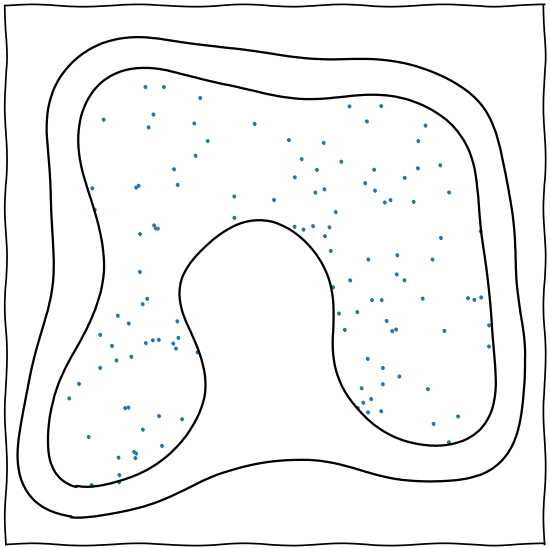


Frontier

- Can you come up with a reversible nested sampling algorithm?
- Can you reduce the volume estimation error by clamping the end point?

Reversible nested sampling

- One could in principle reverse the direction of travel, and move outward from a peak.
- Need to know the initial volume $X_{\text{start}} \neq 1$
- e.g. if a Laplace estimation were good enough at the peak.
- The same trick in normal nested sampling would reduce volume estimation error and hence improve scaling.

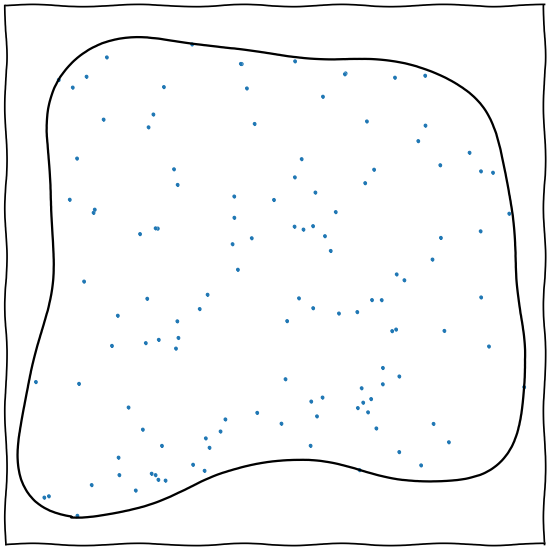


Frontier

- Can you come up with a reversible nested sampling algorithm?
- Can you reduce the volume estimation error by clamping the end point?

Reversible nested sampling

- One could in principle reverse the direction of travel, and move outward from a peak.
- Need to know the initial volume $X_{\text{start}} \neq 1$
- e.g. if a Laplace estimation were good enough at the peak.
- The same trick in normal nested sampling would reduce volume estimation error and hence improve scaling.

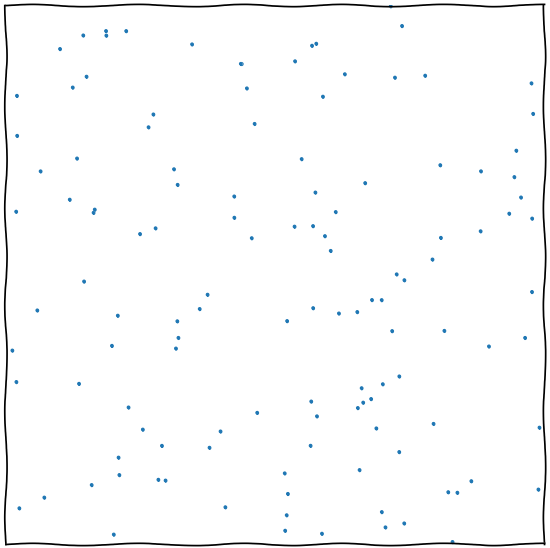


Frontier

- Can you come up with a reversible nested sampling algorithm?
- Can you reduce the volume estimation error by clamping the end point?

Reversible nested sampling

- One could in principle reverse the direction of travel, and move outward from a peak.
- Need to know the initial volume $X_{\text{start}} \neq 1$
- e.g. if a Laplace estimation were good enough at the peak.
- The same trick in normal nested sampling would reduce volume estimation error and hence improve scaling.



Frontier

- Can you come up with a reversible nested sampling algorithm?
- Can you reduce the volume estimation error by clamping the end point?

Memory scaling

Adam Ormondroyd
PhD

- Memory is driven by needing to store:
  - dead points
    $n_{\text{dead}} = n_{\text{live}} \times \mathcal{D}_{\text{KL}} \sim \mathcal{O}(d^2)$
  - optionally their coordinates $\sim \mathcal{O}(d)$
  - optionally further machinery for generating new live points $\sim \mathcal{O}(d)$
- `anesthetic` [1905.04768] as a performant tool for post-processing nested sampling runs

weighted `pandas.DataFrame`

```
[4]: ns = read_chains(gaussian3d_path + "gaussian3d_ns_100nlive_polychord_raw/gaussian3d_ns_100nlive")

[5]: ns
```

[5]:

		x1	x2	x3	logprior__0	loglike__gaussian3d	logL	logL_birth	nlive
	labels	x_1	x_2	x_3	$\log \pi_0$	$\log \mathcal{L}_{\text{gaussian3d}}$	$\ln \mathcal{L}$	$\ln \mathcal{L}_{\text{birth}}$	n_{live}
	weights								
0	3.120715e-17	-5.492322	5.443531	5.329458	-7.45472	-46.857193	-46.857193	-inf	114
1	1.455935e-16	-5.407397	-5.153382	-5.415083	-7.45472	-45.317022	-45.317022	-inf	113
2	1.704418e-14	-3.600835	5.347057	-5.834200	-7.45472	-40.554276	-40.554276	-inf	112
3	1.778227e-14	4.457432	4.539513	-5.918974	-7.45472	-40.511883	-40.511883	-inf	111
4	2.113157e-14	5.046166	4.235301	-5.635906	-7.45472	-40.339317	-40.339317	-inf	110
...	...	...	...	...	...	...	...	...	...
1377	3.269885e-03	-0.009055	-0.031159	0.002386	-7.45472	-2.757345	-2.757345	-2.781492	5
1378	3.269975e-03	-0.028005	-0.014035	0.004734	-7.45472	-2.757317	-2.757317	-2.767012	4
1379	3.270127e-03	-0.028708	0.003180	-0.008758	-7.45472	-2.757271	-2.757271	-2.770868	3
1380	3.270155e-03	-0.021166	-0.003618	0.020802	-7.45472	-2.757263	-2.757263	-2.789061	2
1381	3.271606e-03	0.001819	0.001527	-0.000824	-7.45472	-2.756819	-2.756819	-2.770868	1

1382 rows × 8 columns

Conclusions

github.com/handley-lab

How fast in nested sampling?

$$T = T_{\mathcal{L}} \times n_{\text{live}} \times \mathcal{D}_{\text{KL}} \times f_{\text{sampler}}$$

- Scaling on the face of it $T \sim \mathcal{O}(d^4)$
- In reality can be anywhere between $T \sim \mathcal{O}(d_{\text{slow}})$ and $T \sim \mathcal{O}(d^4)$
- Memory scaling $\sim \mathcal{O}(d^{2-4})$
- Most people focus on improving f_{sampler}, but this is only a fraction of the story.
- Large accelerations to be made in the other terms

How accurate is nested sampling?

$$\sigma(\log \mathcal{Z}) \approx \sqrt{\mathcal{D}_{\text{KL}}/n_{\text{live}}}$$

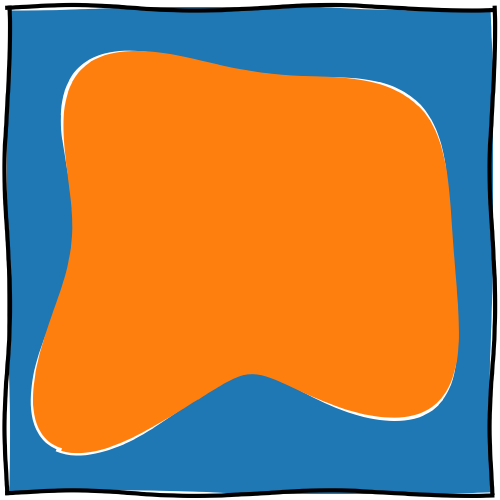
- Technical Review paper: Buchner [2101.09675]
- Nature Review paper: [2205.15570]
- `aeons` [2312.00294]
- `SuperNest` [2212.01760]

Probabalistic volume estimation

- Key idea in NS: estimating volumes probabilistically

$$\frac{V_{\text{after}}}{V_{\text{before}}} \approx \frac{n_{\text{in}}}{n_{\text{out}} + n_{\text{in}}}$$

- This is the **only** way to calculate volume in high dimensions $d > 3$.
  - Geometry is exponentially inefficient.
- This estimation process does not depend on geometry, topology or dimensionality
- Basis of all Monte-Carlo integration
- Nested Sampling uniquely uses a nested framework to couple together MC integrals in a robust, scalable manner.

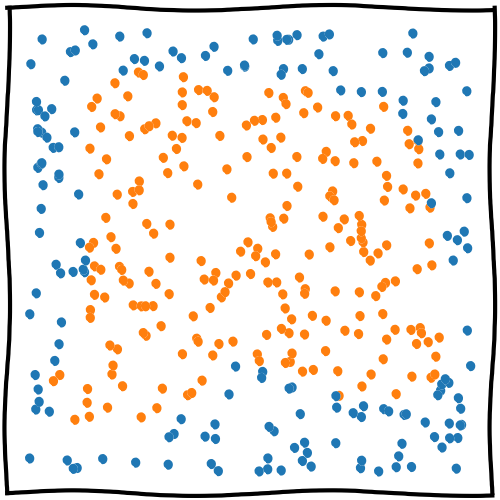


Probabalistic volume estimation

- Key idea in NS: estimating volumes probabilistically

$$\frac{V_{\text{after}}}{V_{\text{before}}} \approx \frac{n_{\text{in}}}{n_{\text{out}} + n_{\text{in}}}$$

- This is the **only** way to calculate volume in high dimensions $d > 3$.
  - Geometry is exponentially inefficient.
- This estimation process does not depend on geometry, topology or dimensionality
- Basis of all Monte-Carlo integration
- Nested Sampling uniquely uses a nested framework to couple together MC integrals in a robust, scalable manner.

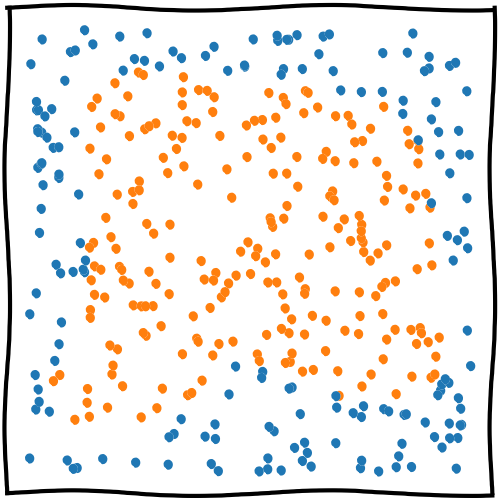


Probabalistic volume estimation

- Key idea in NS: estimating volumes probabilistically

$$\frac{V_{\text{after}}}{V_{\text{before}}} \approx \frac{n_{\text{in}} + 1}{n_{\text{out}} + n_{\text{in}} + 2}$$

- This is the **only** way to calculate volume in high dimensions $d > 3$.
  - Geometry is exponentially inefficient.
- This estimation process does not depend on geometry, topology or dimensionality
- Basis of all Monte-Carlo integration
- Nested Sampling uniquely uses a nested framework to couple together MC integrals in a robust, scalable manner.

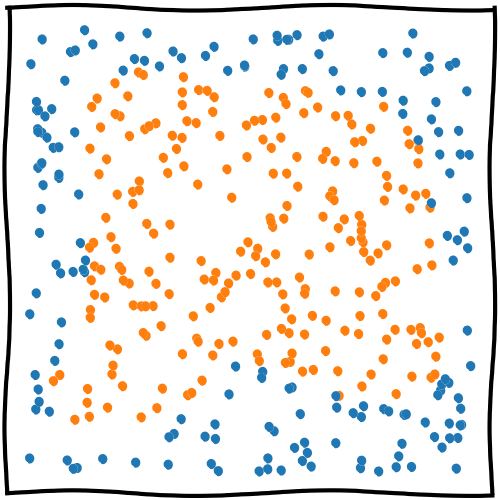


Probabalistic volume estimation

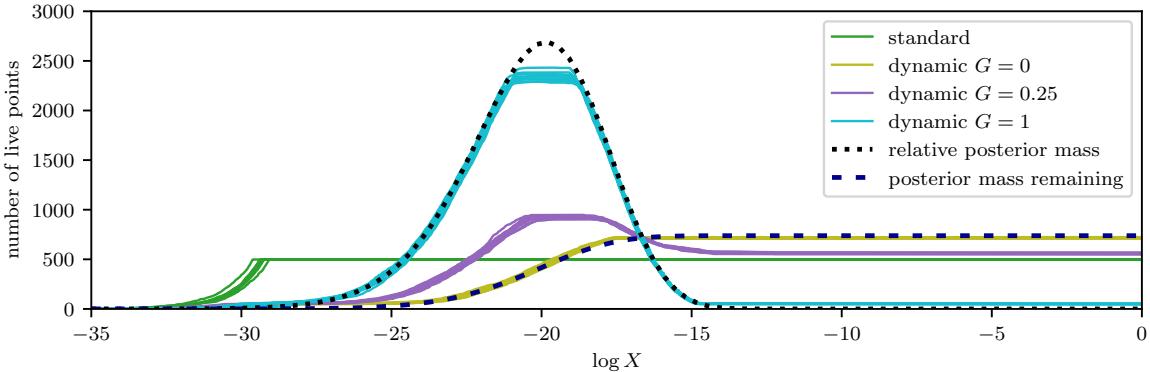
- Key idea in NS: estimating volumes probabilistically

$$\frac{V_{\text{after}}}{V_{\text{before}}} \sim \frac{n_{\text{in}}+1}{n_{\text{out}}+n_{\text{in}}+2} \pm \sqrt{\frac{(n_{\text{in}}+1)(n_{\text{out}}+1)}{(n_{\text{out}}+n_{\text{in}}+2)^2(n_{\text{out}}+n_{\text{in}}+3)}}$$

- This is the **only** way to calculate volume in high dimensions $d > 3$.
  - Geometry is exponentially inefficient.
- This estimation process does not depend on geometry, topology or dimensionality
- Basis of all Monte-Carlo integration
- Nested Sampling uniquely uses a nested framework to couple together MC integrals in a robust, scalable manner.



Why dynamic nested sampling doesn't help



- In dynamic nested sampling [1704.03459], live points are varied by choosing to optionally create/delete at each iteration.
- Can use this to "focus" computational power on posterior.
- However, this is at the cost of reducing evidence accuracy.
- Equivalent to saying "We could just set n_{live} very low" as a solution to scaling.