PolySwyft

a sequential simulation-based nested sampler

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

16th May 2024


UNIVERSITY OF CAMBRIDGE

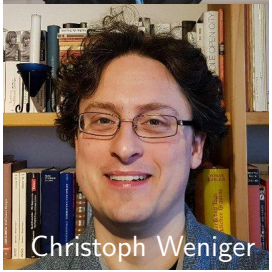


Contents

1. Notation
2. Neural Ratio estimation (NRE)
3. Nested sampling (NS)
4. NS+NRE
5. Future prospects

Stems from over a year of discussion, with the majority of the work done by Kilian Scheutwinkel (PhD student).


Kilian Scheutwinkel


Christoph Weniger

Notation

- A "generative" model M, with tunable parameters θ, describing (compressed) data D.
  - e.g. $M = \Lambda\text{CDM}$, $\theta = \{\Omega_b, \Omega_c, \tau, H_0, A_s, n_s\}$, $D = \{C_\ell\}$.
- Described by simulation process $\theta \to D$, or likelihood $P(D|\theta, M)$.
- Quantifying uncertainty with probability using Bayes theorem:

$$P(\theta|D, M) = \frac{P(D|\theta, M)P(\theta|M)}{P(D|M)}, \quad \text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}, \quad \mathcal{P}(\theta|D) = \frac{\mathcal{L}(D|\theta)\pi(\theta)}{\mathcal{Z}(D)}.$$

- Follows from the oft-forgotten Joint (the probability of everything):

$$\mathcal{P} \times \mathcal{Z} = \mathcal{J} = \mathcal{L} \times \pi, \qquad P(D, \theta|M) = \text{Joint} = \mathcal{J}$$

- Highly relevant (in many overlapping contexts) is the dimensionless ratio

$$r = \frac{\mathcal{P}}{\pi} = \frac{\mathcal{J}}{\pi \times \mathcal{Z}} = \frac{\mathcal{L}}{\mathcal{Z}}$$

- e.g. r occurs in neural ratio estimation, and has the properties that

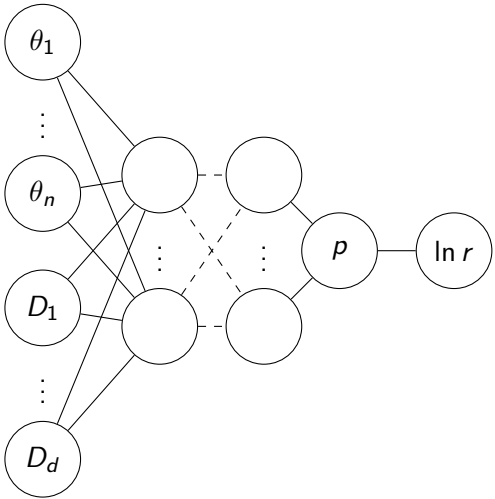
$$\mathcal{D}_{\text{KL}}(\mathcal{P}||\pi) = \langle \log r \rangle_{\mathcal{P}}, \qquad \mathcal{I}(\theta, D)) = \langle \mathcal{D}_{\text{KL}} \rangle_{\mathcal{Z}} = \langle \log r \rangle_{\mathcal{J}}$$

Neural Ratio Estimation

- SBI flavours:

  NPE Neural posterior estimation
  NLE Neural likelihood estimation
  NJE Neural joint estimation
  NRE Neural ratio estimation

- NRE recap:
  1. Generate joint samples $(\theta, D) \sim \mathcal{J}$
     - *straightforward if you have a simulator:* $\theta \sim \pi(\cdot)$, $D \sim \mathcal{L}(\cdot|\theta)$
  2. Generate separated samples $\theta \sim \pi$, $D \sim \mathcal{Z}$
     - *aside: can shortcut step 2 by scrambling the* (θ, D) *pairings from step 1*
  3. Train probabilistic classifier p to distinguish whether (θ, D) came from $\mathcal{J}$ or $\pi \times \mathcal{Z}$.
  4. $\log r = p/(1-p)$.



Neural Ratio Estimation

- SBI flavours:
  - NPE Neural posterior estimation
  - NLE Neural likelihood estimation
  - NJE Neural joint estimation
  - NRE Neural ratio estimation
- NRE recap:
  1. Generate joint samples $(\theta, D) \sim \mathcal{J}$
     - *straightforward if you have a simulator:* $\theta \sim \pi(\cdot)$, $D \sim \mathcal{L}(\cdot|\theta)$
  2. Generate separated samples $\theta \sim \pi$, $D \sim \mathcal{Z}$
     - *aside: can shortcut step 2 by scrambling the* (θ, D) *pairings from step 1*
  3. Train probabilistic classifier p to distinguish whether (θ, D) came from $\mathcal{J}$ or $\pi \times \mathcal{Z}$.
  4. $\log r = p/(1-p)$.

Why I like NRE

- Link between classification and inference is profound.
- Density estimation is hard – Dimensionless r divides out the hard-to-calculate bits.

Why I don't like NRE

- Practical implementations require marginalisation [2107.01214], or autoregression [2308.08597].
- Model comparison and parameter estimation are separate [2305.11241].

TMNRE: Truncated Marginal Neural Ratio Estimation

swyft: github.com/undark-lab/swyft

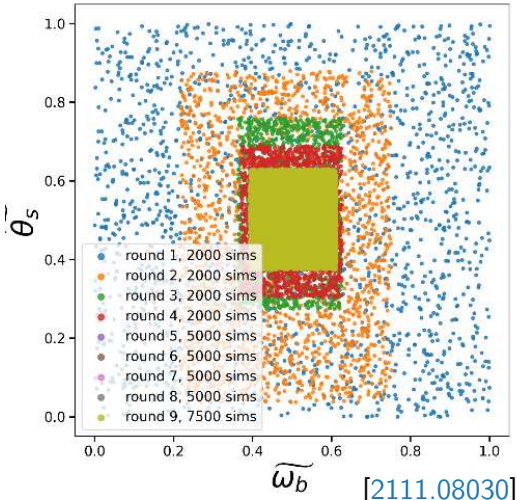
- Two tricks for practical NRE:

Marginalisation

- Only consider one or two parameters at a time.
- Fine if your goal is to produce triangle plots.
- Problematic if information is contained jointly in more than two parameters.

Truncation

- focus parameters θ on a subset of the prior which reproduces observed data $D_{\rm obs}$
- region is somewhat arbitrary (usually a box)
- not amortised, sounds a bit like ABC

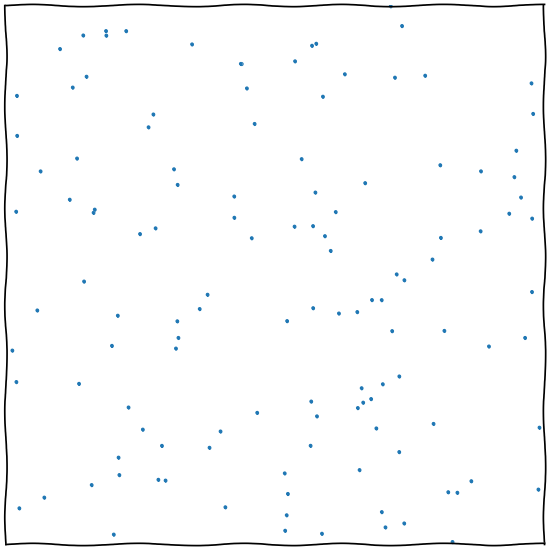


[2111.08030]

Nested sampling: numerical Lebesgue integration

0. Start with N random samples over the space.
i. Delete outermost sample, and replace with a new random one at higher integrand value.

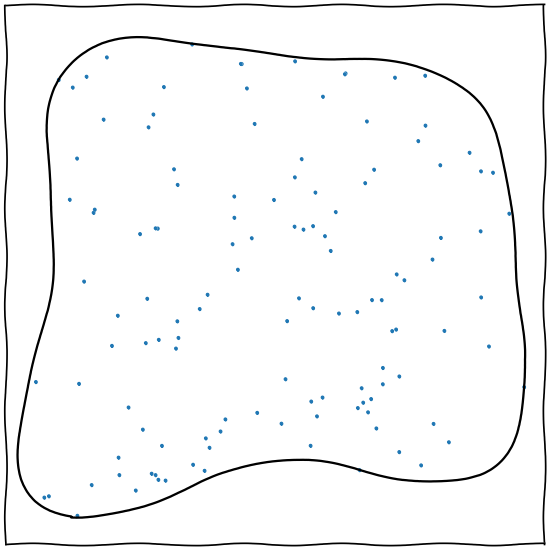
- The "live points" steadily contract around the peak(s) of the function.
- Discarded "dead points" can be weighted to form posterior, prior, or anything in between.
- Estimates the **density of states** and calculates evidences & partition functions.
- The evolving ensemble of live points allows:
  - implementations to self-tune,
  - exploration of multimodal functions,
  - global and local optimisation.



Nested sampling: numerical Lebesgue integration

0. Start with N random samples over the space.
i. Delete outermost sample, and replace with a new random one at higher integrand value.

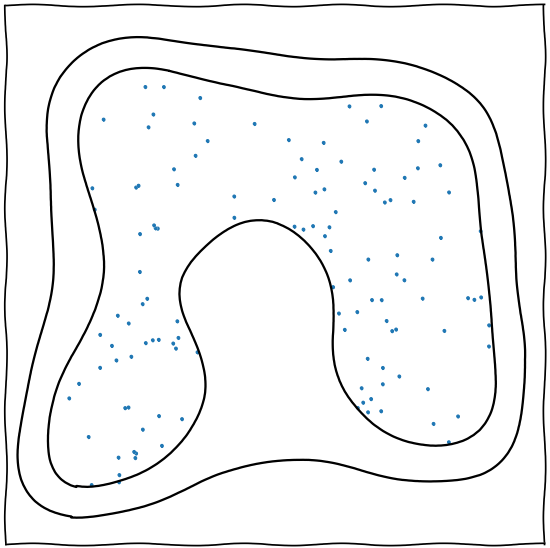
- The "live points" steadily contract around the peak(s) of the function.
- Discarded "dead points" can be weighted to form posterior, prior, or anything in between.
- Estimates the **density of states** and calculates evidences & partition functions.
- The evolving ensemble of live points allows:
  - implementations to self-tune,
  - exploration of multimodal functions,
  - global and local optimisation.



Nested sampling: numerical Lebesgue integration

0. Start with N random samples over the space.
i. Delete outermost sample, and replace with a new random one at higher integrand value.

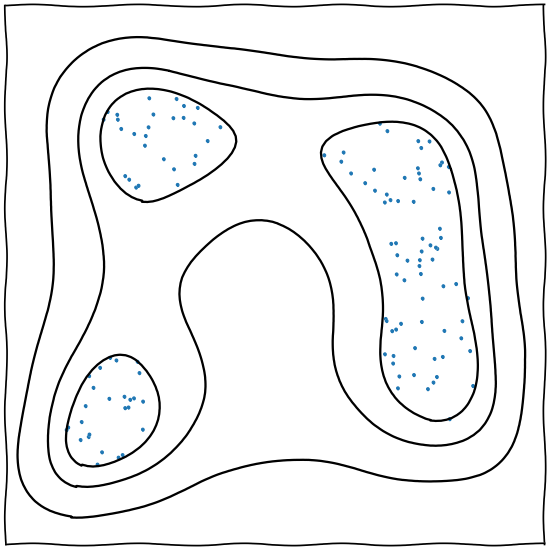
- The "live points" steadily contract around the peak(s) of the function.
- Discarded "dead points" can be weighted to form posterior, prior, or anything in between.
- Estimates the **density of states** and calculates evidences & partition functions.
- The evolving ensemble of live points allows:
  - implementations to self-tune,
  - exploration of multimodal functions,
  - global and local optimisation.



Nested sampling: numerical Lebesgue integration

0. Start with N random samples over the space.
i. Delete outermost sample, and replace with a new random one at higher integrand value.

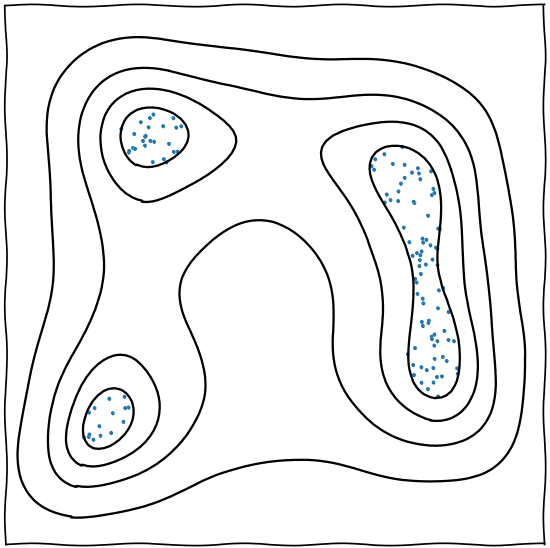
- The "live points" steadily contract around the peak(s) of the function.
- Discarded "dead points" can be weighted to form posterior, prior, or anything in between.
- Estimates the **density of states** and calculates evidences & partition functions.
- The evolving ensemble of live points allows:
  - implementations to self-tune,
  - exploration of multimodal functions,
  - global and local optimisation.



Nested sampling: numerical Lebesgue integration

0. Start with N random samples over the space.
i. Delete outermost sample, and replace with a new random one at higher integrand value.

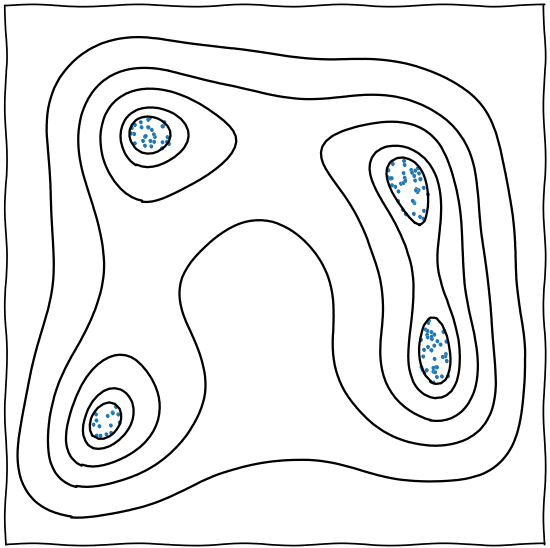
- The "live points" steadily contract around the peak(s) of the function.
- Discarded "dead points" can be weighted to form posterior, prior, or anything in between.
- Estimates the **density of states** and calculates evidences & partition functions.
- The evolving ensemble of live points allows:
  - implementations to self-tune,
  - exploration of multimodal functions,
  - global and local optimisation.



Nested sampling: numerical Lebesgue integration

0. Start with N random samples over the space.
i. Delete outermost sample, and replace with a new random one at higher integrand value.

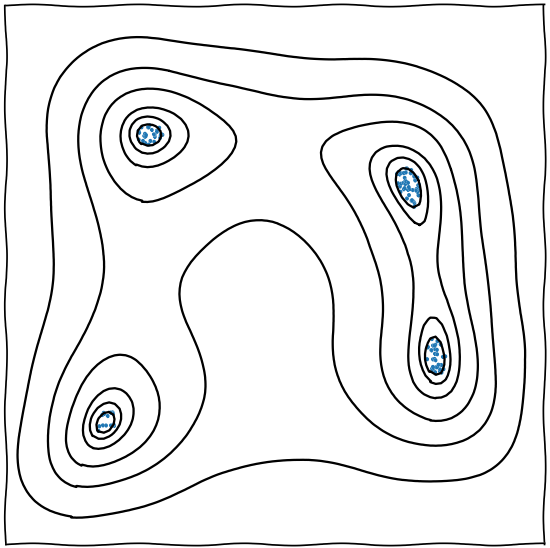
- The "live points" steadily contract around the peak(s) of the function.
- Discarded "dead points" can be weighted to form posterior, prior, or anything in between.
- Estimates the **density of states** and calculates evidences & partition functions.
- The evolving ensemble of live points allows:
  - implementations to self-tune,
  - exploration of multimodal functions,
  - global and local optimisation.



Nested sampling: numerical Lebesgue integration

0. Start with N random samples over the space.
i. Delete outermost sample, and replace with a new random one at higher integrand value.

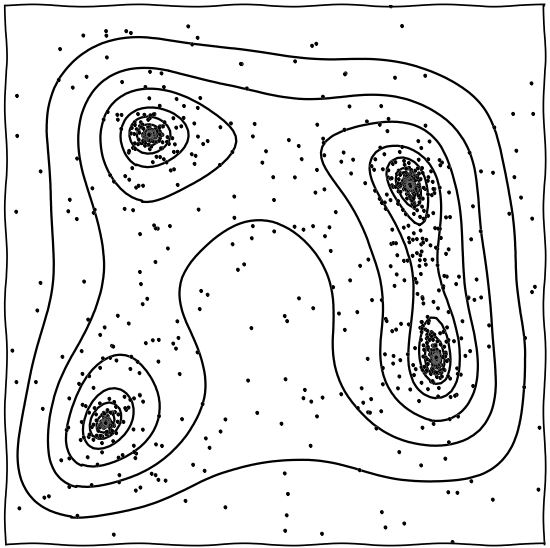
- The "live points" steadily contract around the peak(s) of the function.
- Discarded "dead points" can be weighted to form posterior, prior, or anything in between.
- Estimates the **density of states** and calculates evidences & partition functions.
- The evolving ensemble of live points allows:
  - implementations to self-tune,
  - exploration of multimodal functions,
  - global and local optimisation.



Nested sampling: numerical Lebesgue integration

0. Start with N random samples over the space.
i. Delete outermost sample, and replace with a new random one at higher integrand value.

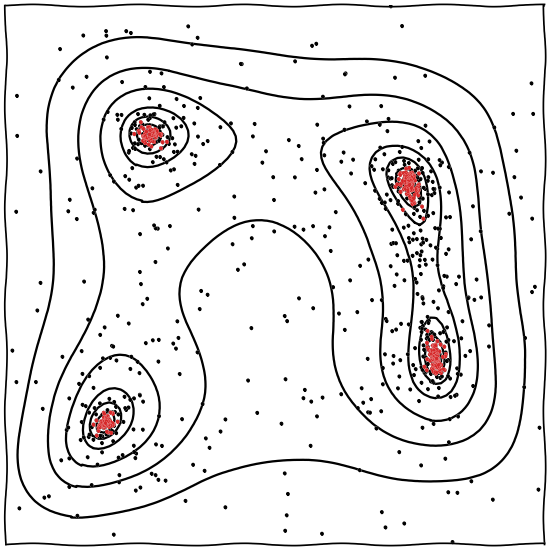
- The "live points" steadily contract around the peak(s) of the function.
- Discarded "dead points" can be weighted to form posterior, prior, or anything in between.
- Estimates the **density of states** and calculates evidences & partition functions.
- The evolving ensemble of live points allows:
  - implementations to self-tune,
  - exploration of multimodal functions,
  - global and local optimisation.



Nested sampling: numerical Lebesgue integration

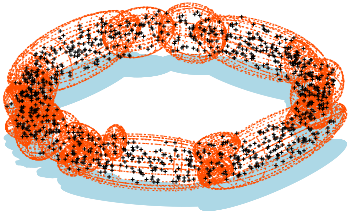
0. Start with N random samples over the space.
i. Delete outermost sample, and replace with a new random one at higher integrand value.

- The "live points" steadily contract around the peak(s) of the function.
- Discarded "dead points" can be weighted to form posterior, prior, or anything in between.
- Estimates the **density of states** and calculates evidences & partition functions.
- The evolving ensemble of live points allows:
  - implementations to self-tune,
  - exploration of multimodal functions,
  - global and local optimisation.

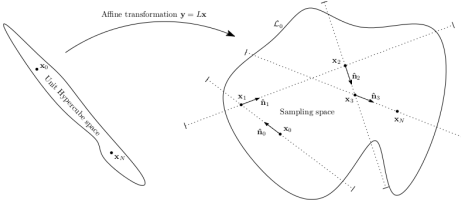


Implementations of Nested Sampling [2205.15570](NatReview)

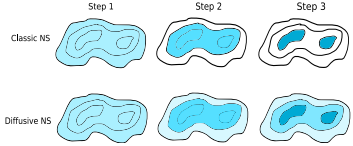
MultiNest [0809.3437]



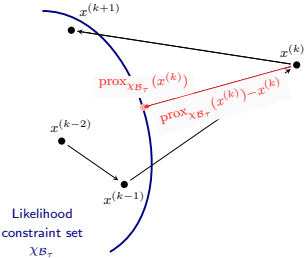
PolyChord [1506.00171]



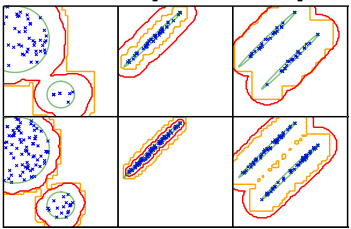
DNest [1606.03757]



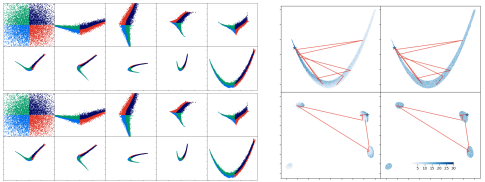
ProxNest [2106.03646]



UltraNest [2101.09604]



NeuralNest [1903.10860]



nessai [2102.11056]

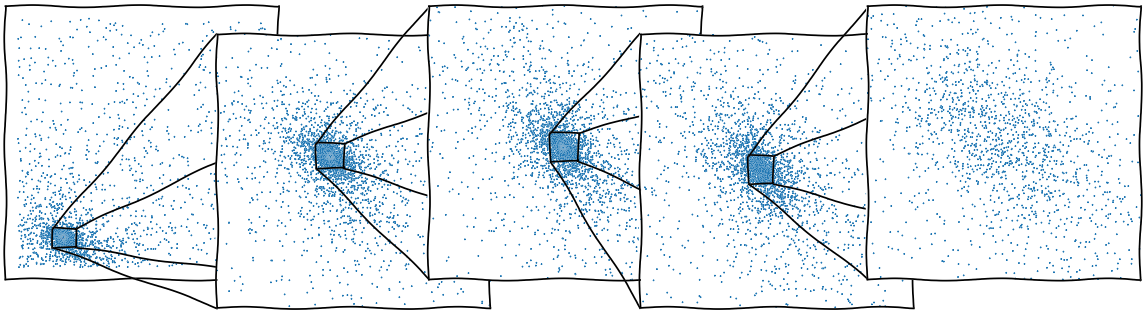
nora [2305.19267]

jaxnest [2012.15286]

nautilus [2306.16923]

dynesty [1904.02180]

The nested sampling meta-algorithm: dead points

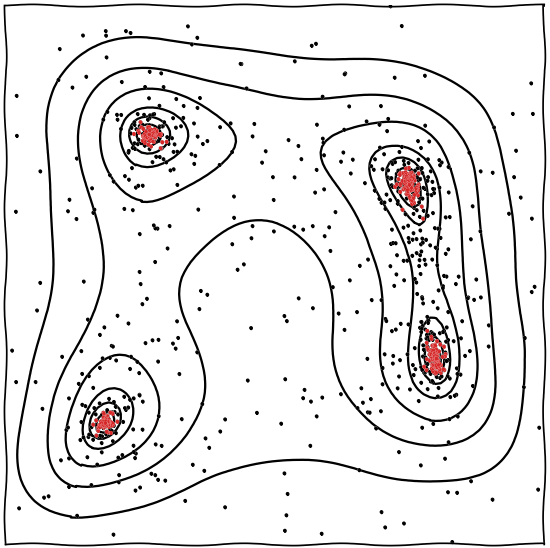


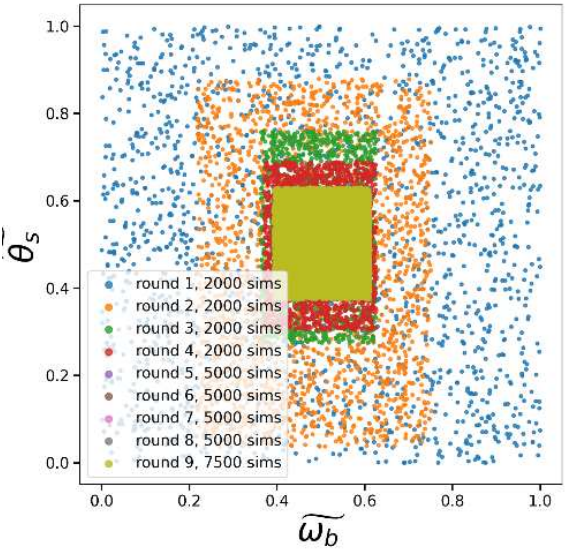
- At the end, one is left with a set of discarded "dead" points.
- Dead points have a unique scale-invariant distribution $\propto \frac{dV}{V}$.
- Uniform over original region, exponentially concentrating on region of interest (until termination volume).
- Good for training emulators (HERA [2108.07282]).

Applications

- training emulators.
- gridding simulations
- beta flows
- "dead measure"

Similarities





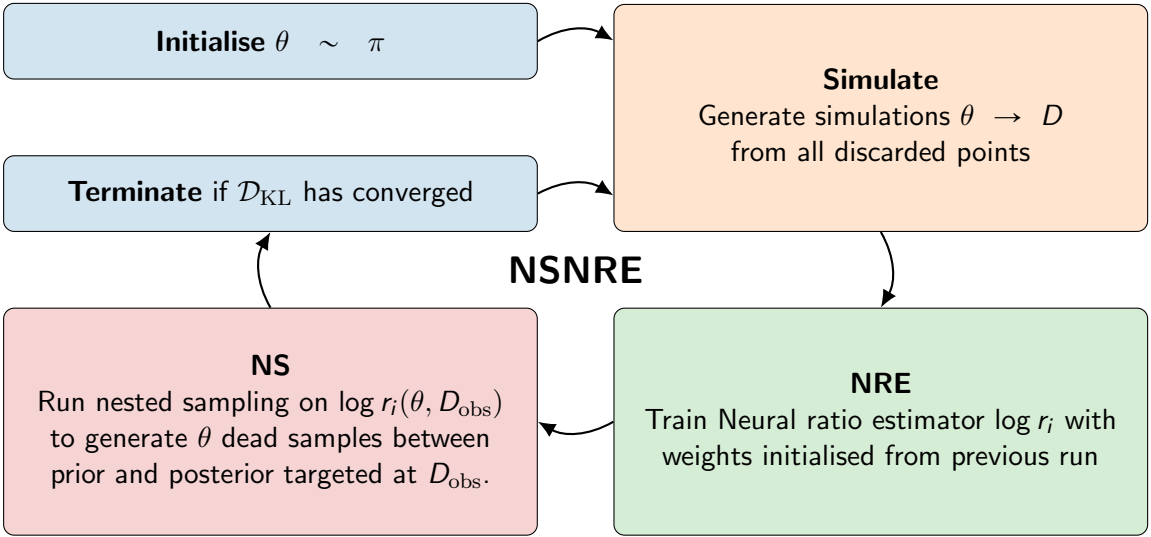
Why it's hard to do SBI with nested sampling

- At each iteration i, nested sampling requires you to be able to generate a new live point from the prior, subject to a hard likelihood constraint

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

- This is hard if you don't have a likelihood!
- In addition, nested sampling does not do well if the likelihood is non-deterministic
- Previous attempts:
  - DNest paper [1606.03757](Section 10: Nested sampling for ABC)
  - ANRE [2308.08597] using non-box priors driven by current ratio estimate with slice sampling re-population.

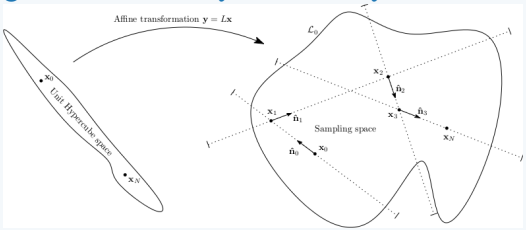
Sequential NRE with nested sampling



PolySwyft

PolyChord

github.com/PolyChord/PolyChordLite



- Widely used high-performance nested sampling tool (implementing slice sampling & clustering in MPI Fortran)

Swyft

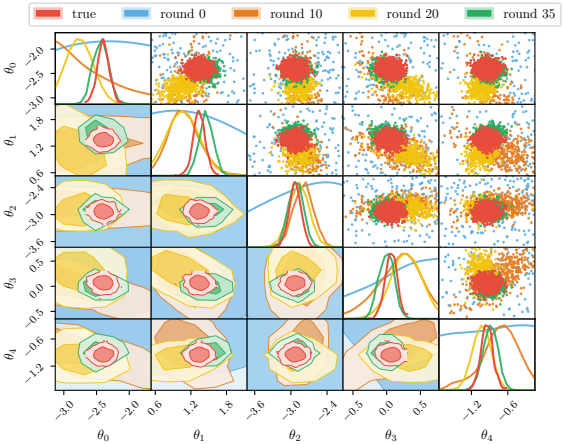
github.com/undark-lab/swyft

- Widely used TMNRE tool in cosmology/astrophysics.

However, NSNRE is general, and not specific to these choices.

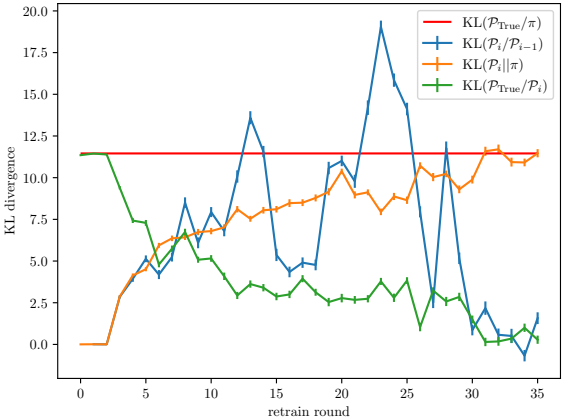
Convergence diagnostics

- Example for a $n = 5$ dimensional parameter space, with $d = 100$ data points, (`lsbi` gaussian mixture model).
- This is the regime for cosmological scale problems.
- Determining convergence is a work-in-progress.
- Currently track:
  - The change in KL divergence between rounds (blue), and check when this goes to zero.
  - The total KL divergence between prior and posterior estimate (orange), and check when this levels off (ground truth in red).
  - Also shown is the KL divergence between the estimate and the ground truth (green).



Convergence diagnostics

- Example for a $n = 5$ dimensional parameter space, with $d = 100$ data points, (`lsbi` gaussian mixture model).
- This is the regime for cosmological scale problems.
- Determining convergence is a work-in-progress.
- Currently track:
  - The change in KL divergence between rounds (blue), and check when this goes to zero.
  - The total KL divergence between prior and posterior estimate (orange), and check when this levels off (ground truth in red).
  - Also shown is the KL divergence between the estimate and the ground truth (green).



What does NSNRE give you that TMNRE doesn't?

- Use of dead points which scan from prior to logr peak avoids risk of 'trimming' important regions of the space
- Likelihood-driven contours (slightly) allow more parameters $n > 2$ to be considered in comparison with box priors

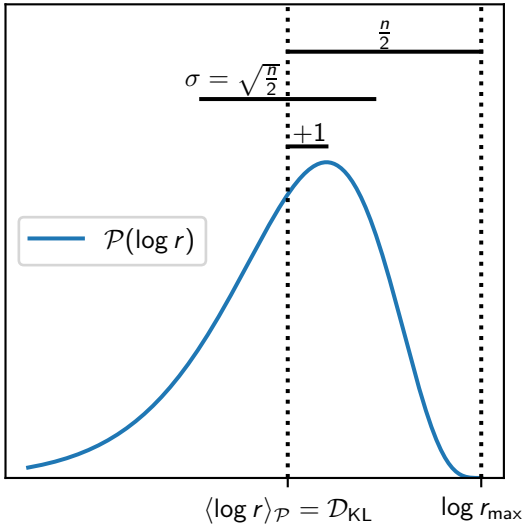
Considerations of ratio estimation

- Neural REs can in practice only estimate in a band of $\log r$ before the activation function saturates (typically $-5 < \log r < 5$).
- Consider a posterior $\mathcal{P}$ well approximated by a Gaussian profile in an n-dimensional parameter space [2312.00294]
- If $\mathcal{D}_{\mathrm{KL}} \gg 1$ between prior and posterior:

$$\log r = \frac{n}{2} + \mathcal{D}_{\mathrm{KL}} + \chi^2_n$$

$$\langle \log r \rangle_{\mathcal{P}} = \mathcal{D}_{\mathrm{KL}}, \qquad \sigma(\log r)_{\mathcal{P}} = \sqrt{\frac{n}{2}}$$

- Truncation (**T**MNRE) reduces $\mathcal{D}_{\mathrm{KL}}$, focusing the distribution into the $[-5, 5]$ band.
- Marginalisation (T**M**NRE) reduces n & σ.



Conclusions

github.com/handley-lab

- `PolySwyft` can perform NRE on $n = 5$ parameter spaces and $d = 100$ data spaces.
- This makes it relevant for cosmological applications.
- Investigating this raises (?existential) questions regarding NRE.
- Is the $-5 < \log r < 5$ saturation of NREs a fundamental problem, or an engineering one?
- Is there a "nested" approach to crossing this range in larger parameter spaces $n \gg 5$?
- Examples produced using `lsbi` package: github.com/handley-lab/lsbi

Cosmological forecasting

Have you ever done a Fisher forecast, and then felt Bayesian guilt?

- Cosmologists are interested in forecasting what a Bayesian analysis of future data might produce.
- Useful for:
  - white papers/grants,
  - optimising existing instruments/strategies,
  - picking theory/observation to explore next.
- To do this properly:
  1. start from current knowledge $\pi(\theta)$, derived from current data
  2. Pick potential dataset D that might be collected from $P(D)$ $(= \mathcal{Z})$
  3. Derive posterior $P(\theta|D)$
  4. Summarise science (e.g. constraint on θ, ability to perform model comparison)

- This procedure should be marginalised over:
  1. All possible parameters θ (consistent with prior knowledge)
  2. All possible data D
- i.e. marginalised over the joint $P(\theta, D) = P(D|\theta)P(\theta)$.
- Historically this has proven very challenging.
- Most analyses assume a fiducial cosmology θ_*, and/or a Gaussian likelihood/posterior (c.f. Fisher forecasting).
- This runs the risk of biasing forecasts by baking in a given theory/data realisation.

Fully Bayesian Forecasting [2309.06942]

Thomas Gessey-Jones
PhD

- Simulation based inference gives us the language to marginalise over parameters θ and possible future data D.
- Evidence networks give us the ability to do this at scale for forecasting [2305.11241].
- Demonstrated in 21cm global experiments, marginalising over:
  - theoretical uncertainty
  - foreground uncertainty
  - systematic uncertainty
- Able to say "at 67mK radiometer noise", have a 50% chance of 5σ Bayes factor detection.
- Can use to optimise instrument design
- Re-usable package: `prescience`