GPU-native nested sampling in BlackJAX

For simulation-based inference at scale

Will Handley
<wh260@cam.ac.uk>

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

29th May 2025

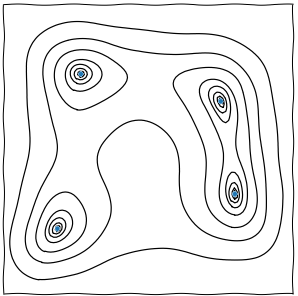

UNIVERSITY OF CAMBRIDGE


What is Nested Sampling?

- Nested sampling is a radical, multi-purpose numerical tool.
- Given a (scalar) function f with a vector of parameters θ, it can be used for:

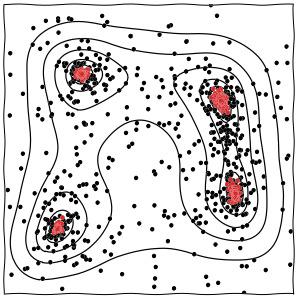
Optimisation

$$\theta_{\max} = \max_{\theta} f(\theta)$$



Exploration

draw/sample $\theta \sim f$

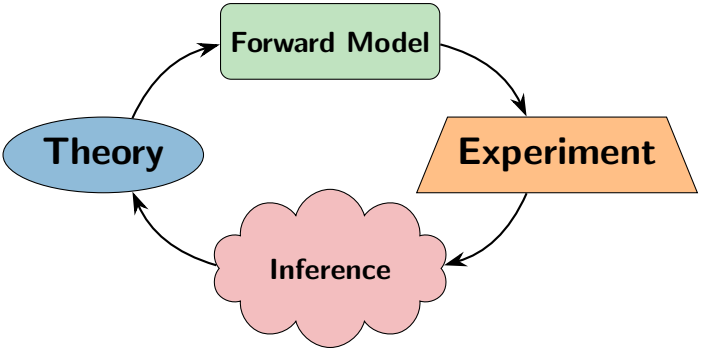


Integration

$$\int f(\theta) dV$$

Why nested sampling for SBI?

- Other than NPE, all SBI methods (NLE, NRE, NJE etc) need a sampler
- Most existing samplers are limited:
  - **Legacy implementations**: MultiNest, PolyChord (Fortran)
  - **Slow Python codes**: dynesty, ultranest, nautilus
- BlackJAX solution:
  - GPU-native implementation
  - Open source & community-driven
  - Dissociated from original authors



The GPU imperative

- The future is GPU, whether we like it or not:
  - All future HPC heavily weighted toward GPUs
  - Driven by machine learning adoption
- JAX does two *separate* things:
  1. **Automatic differentiation**
  2. **Just-in-time compilation** for GPUs
- People often conflate these - they are separate and glorious!
- Alternative tools (harmonic, floz) get strength from GPU, not sampling method

GPU ecosystem growth

CMB	`cosmopower`, `candl`
SNe	`BayesSN`
GW	`ripple`, `jim`
EP	`ExoJAX`

[github:JAXtronomy]

BlackJAX nested sampling

David Yallup
PDRA

- Very recent development (December 2024!)
- GPU-native nested slice sampler
- Modern alternative to MultiNest/PolyChord
- Slice sampling with adaptive live point allocation
- Compatible with existing BlackJAX ecosystem

Learn more

- Theory: `handley-lab.co.uk/nested-sampling-book`
- Implementation: David Yallup's PDRA work
- Workshop: Hands-on tutorial today

Installation

```
pip install git+https://github.com/handley-lab/blackjax
```

AI-enabled research workflows

- Modern scientific computing benefits from AI integration:
  - GPU-accelerated inference
  - Neural networks in SBI methods
  - Automatic differentiation for sampling
- But the real coming disruption is from using AI to do day-to-day scientific tasks.
  - I wrote this talk with AI assistance [claude code]. This morning.
  - I have written and been awarded several grants that were majority written by AI.
  - My most recent papers are drafted with AI.

AI-enabled research workflows

- Modern scientific computing benefits from AI integration:
  - GPU-accelerated inference
  - Neural networks in SBI methods
  - Automatic differentiation for sampling
- But the real coming disruption is from using AI to do day-to-day scientific tasks.
  - I wrote this talk with AI assistance [claude code]. This morning.
  - I have written and been awarded several grants that were majority written by AI.
  - My most recent papers are drafted with AI.

Workshop goals

- Today we'll explore:
  1. Running nested sampling with BlackJAX
  2. Visualization with Anesthetic
  3. Performance comparison: nested sampling vs AIES
  4. Integration with your JAX workflows
- Hands-on notebook environment
- Google Colab compatible
- Build on Viraj's JAX/SciML workshop

Links

- BlackJAX: [github:handley-lab/blackjax]
- Anesthetic: `anesthetic.readthedocs.io`
- Workshop materials: [github:handley-lab]

Performance promise

Compare BlackJAX nested sampling performance with traditional tools and see the GPU advantage firsthand!

Conclusions
[github:handley-lab]

- **Nested sampling is essential** for most SBI methods (except NPE)
- **BlackJAX** provides GPU-native, community-driven implementation
- **JAX's dual power**: autodiff + JIT compilation for unprecedented performance
- **The future is GPU-accelerated** scientific computing with AI integration

Integration in Physics

- Integration is a fundamental concept in physics, statistics and data science:

Partition functions

$$Z(\beta) = \int e^{-\beta H(q,p)} dq dp$$

Path integrals

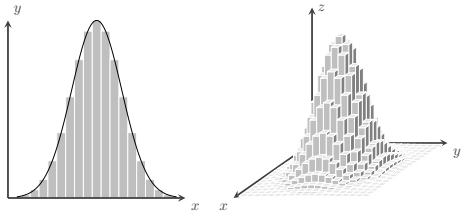
$$\Psi = \int e^{iS} \mathcal{D}x$$

Bayesian marginals

$$\mathcal{Z}(D) = \int \mathcal{L}(D|\theta)\pi(\theta) d\theta$$

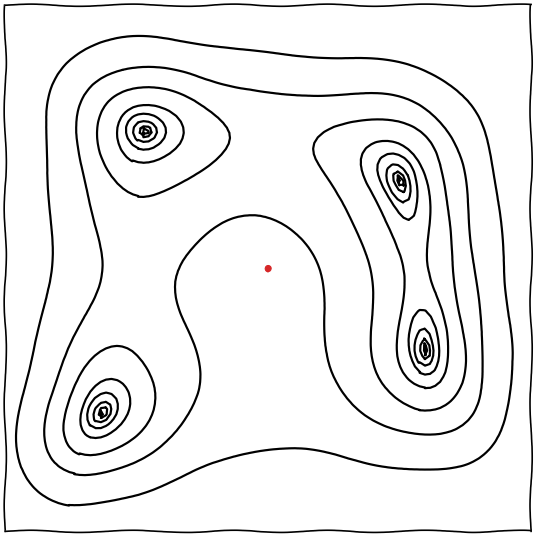
- Need numerical tools if analytic solution unavailable.
- High-dimensional numerical integration is hard.
- Riemannian strategy estimates volumes geometrically:

$$\int f(x) d^n x \approx \sum_i f(x_i) \Delta V_i \sim \mathcal{O}(e^n)$$

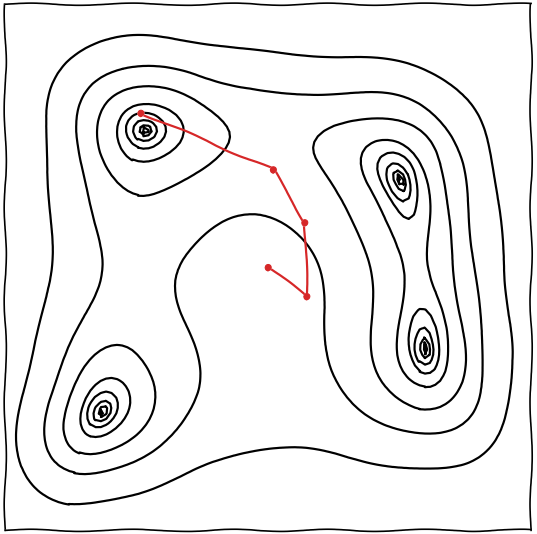


- Curse of dimensionality $\Rightarrow$ exponential scaling.

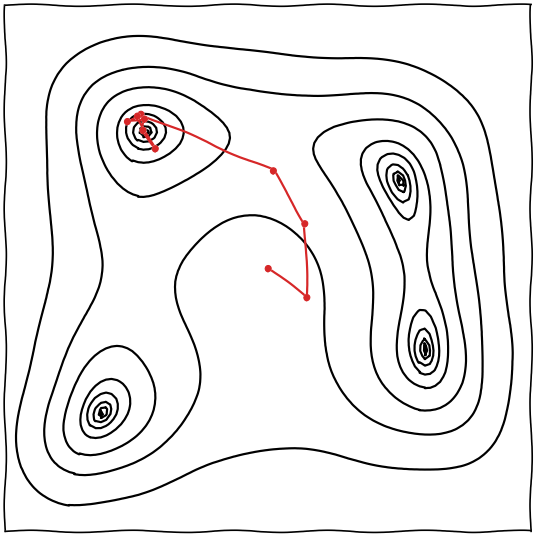
MCMC



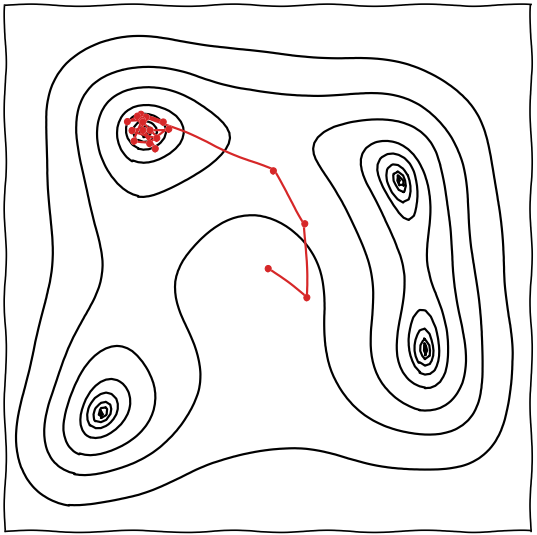
MCMC



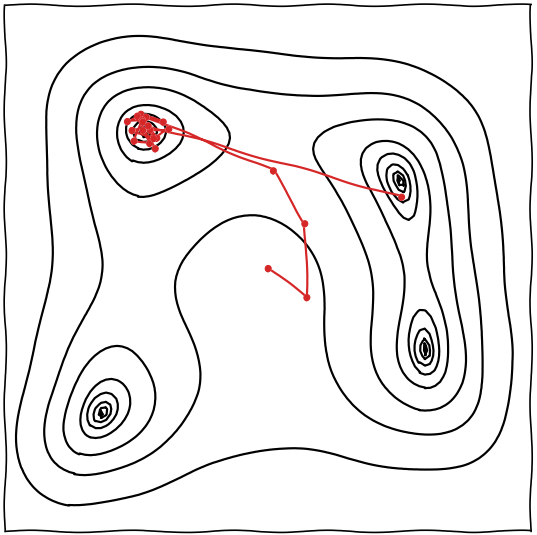
MCMC



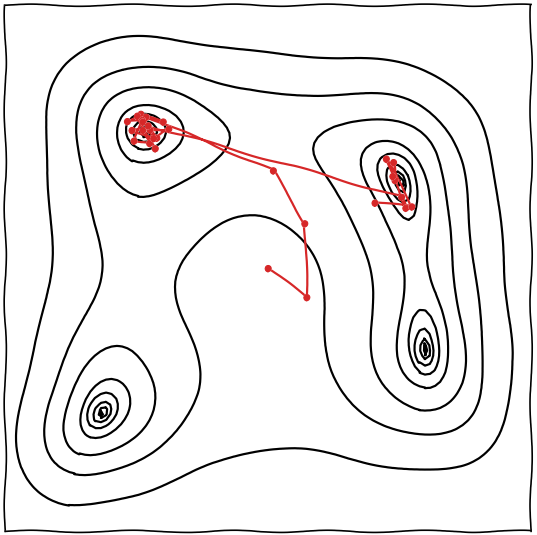
MCMC



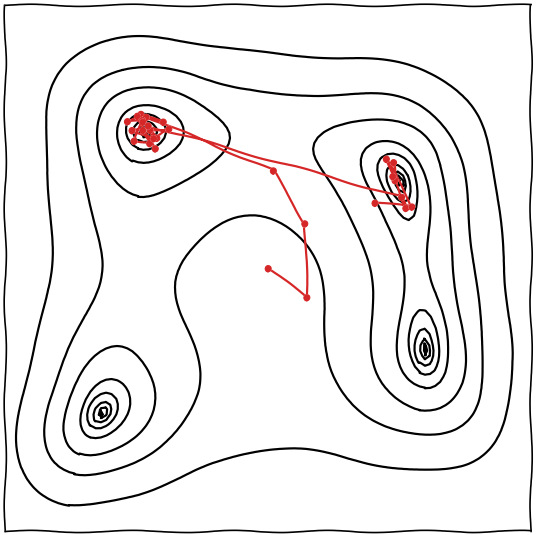
MCMC



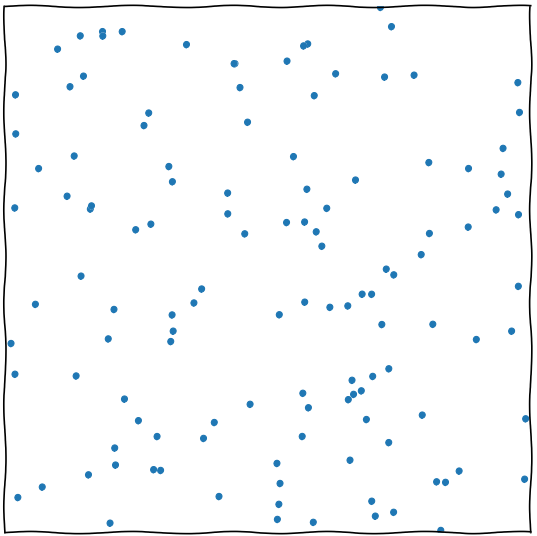
MCMC



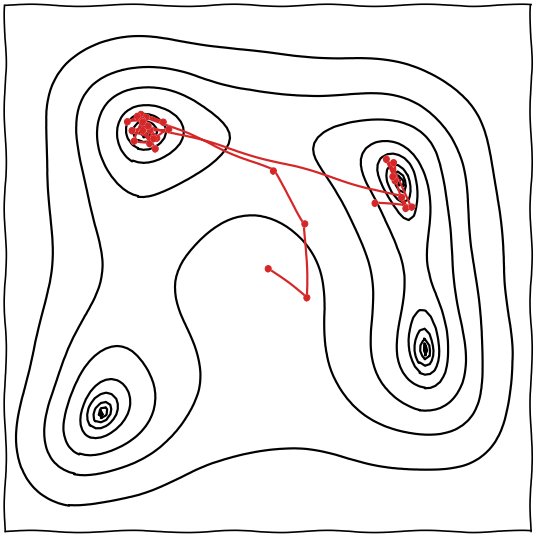
MCMC



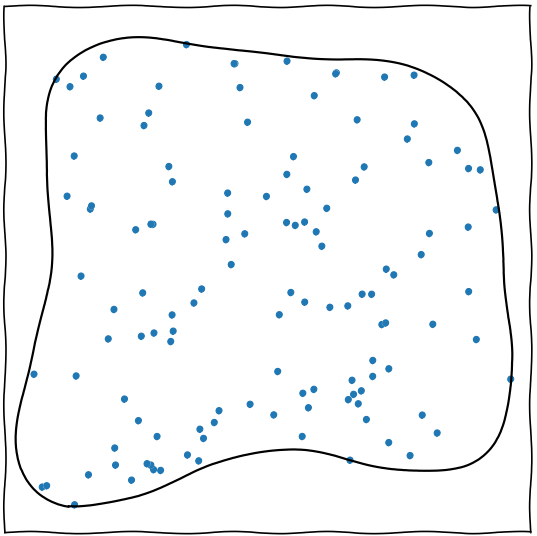
Nested sampling



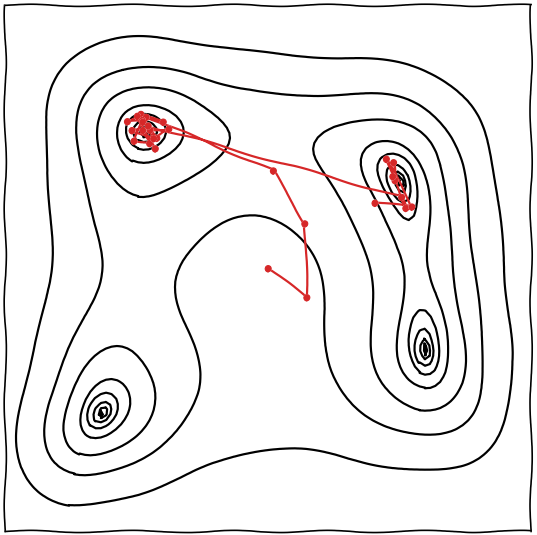
MCMC



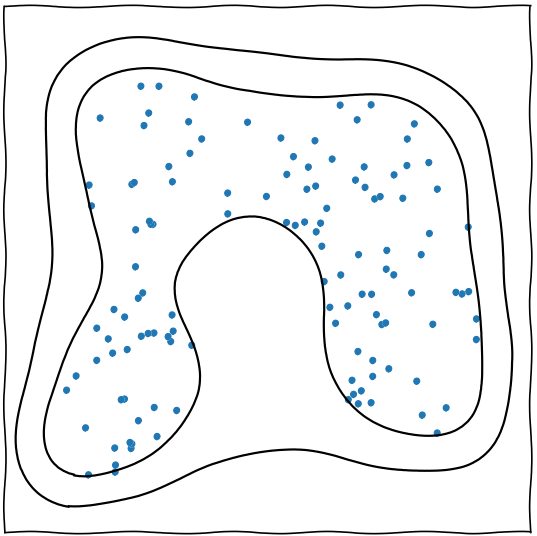
Nested sampling



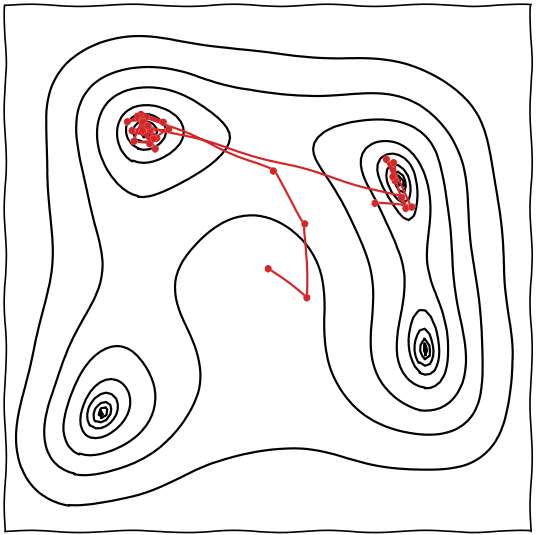
MCMC



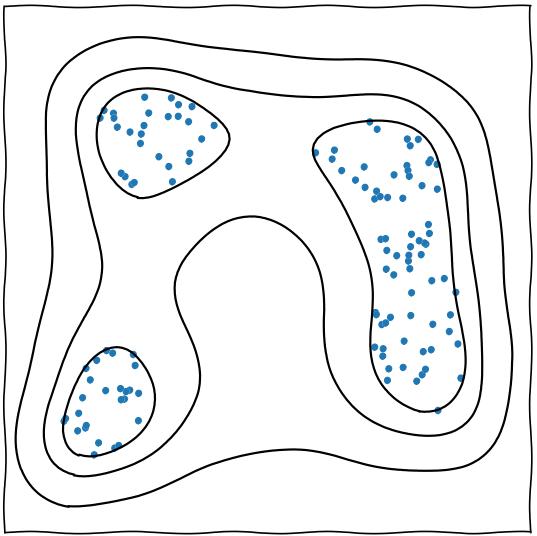
Nested sampling



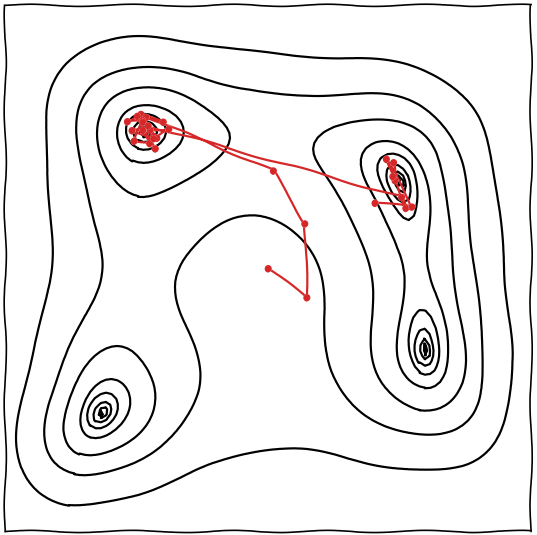
MCMC



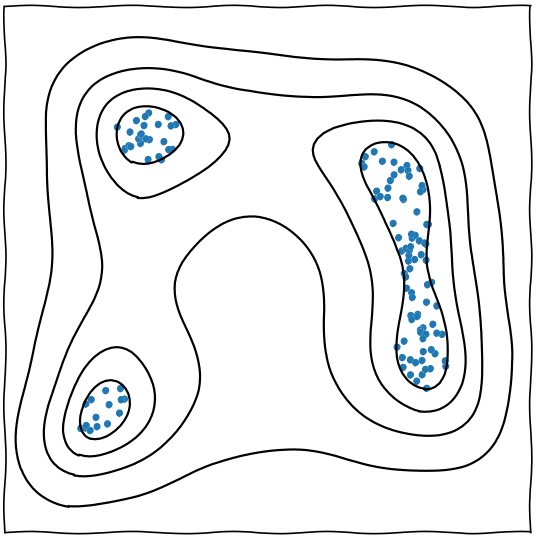
Nested sampling



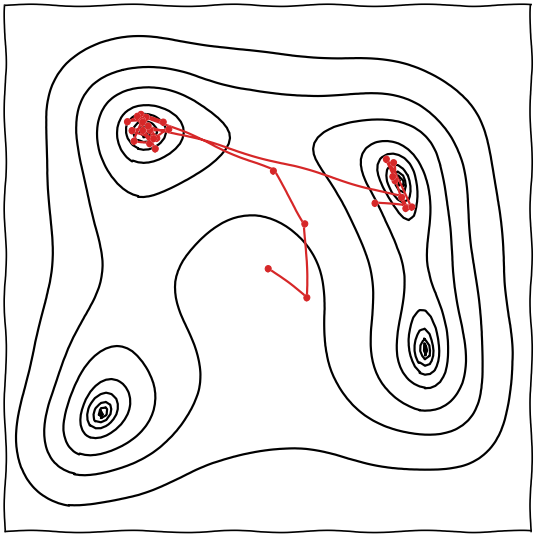
MCMC



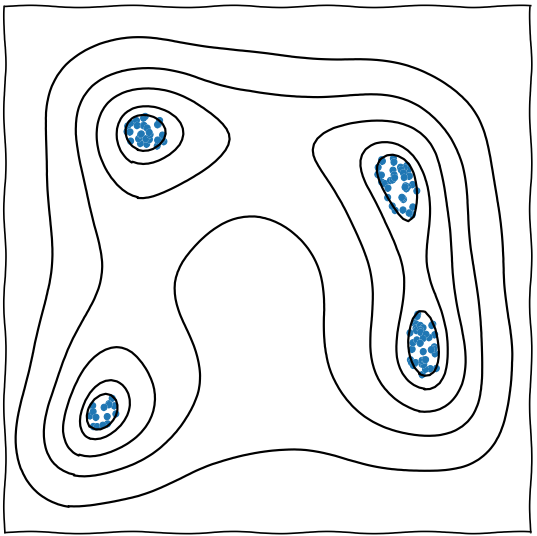
Nested sampling



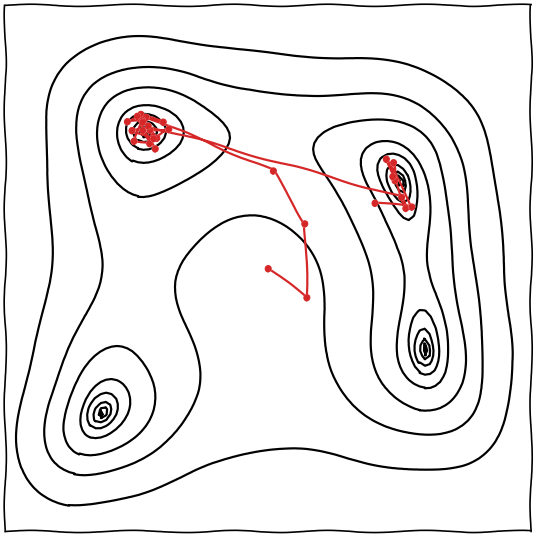
MCMC



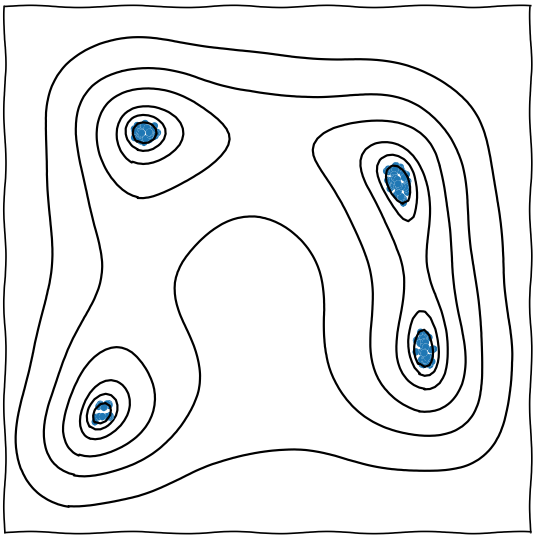
Nested sampling



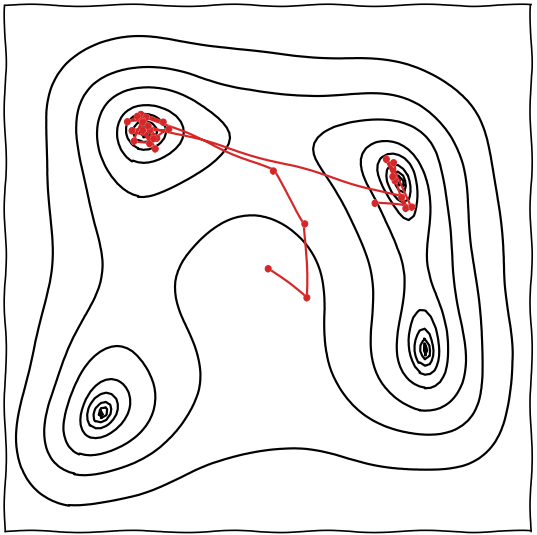
MCMC



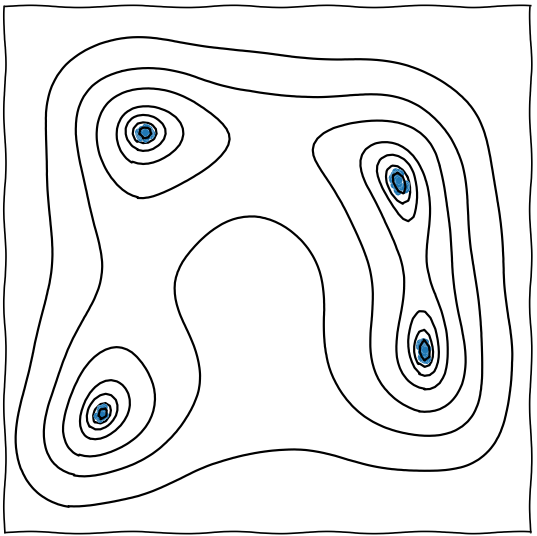
Nested sampling



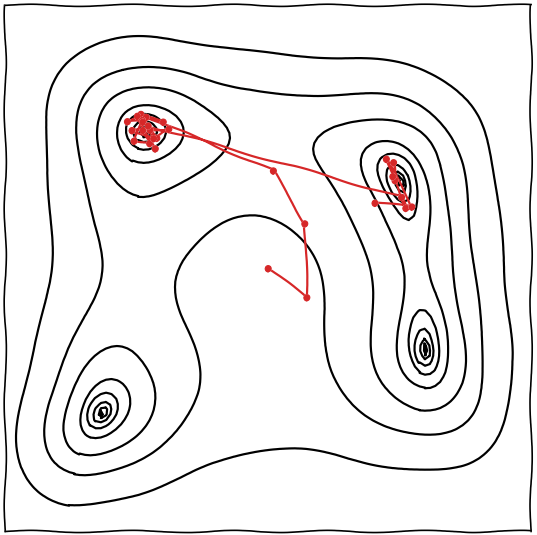
MCMC



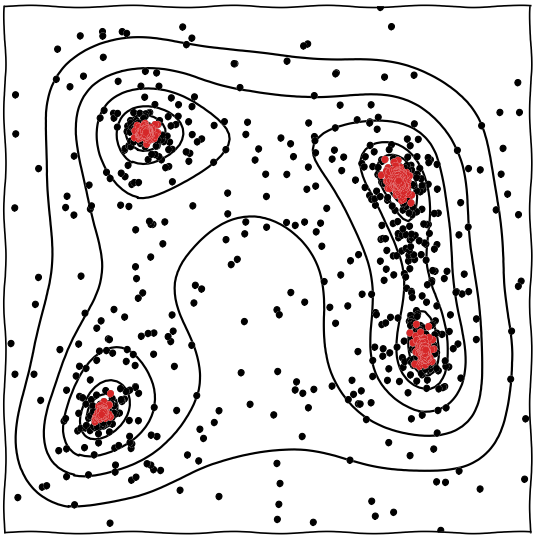
Nested sampling



MCMC

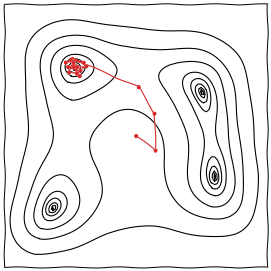


Nested sampling



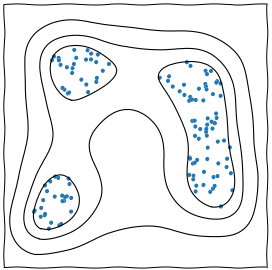
MCMC

- Single "walker"
- Explores posterior
- Fast, if proposal matrix is tuned
- Parameter estimation, suspiciousness calculation
- Channel capacity optimised for generating posterior samples



Nested sampling

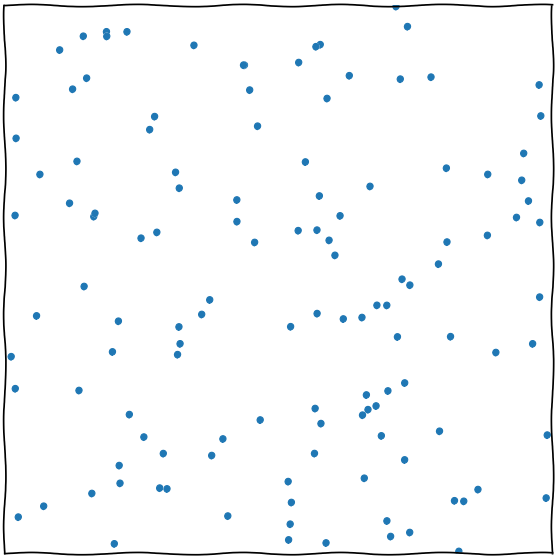
- Ensemble of "live points"
- Scans from prior to peak of likelihood
- Slower, no tuning required
- Parameter estimation, model comparison, tension quantification
- Channel capacity optimised for computing partition function



The nested sampling meta-algorithm: live points

- Start with n random samples over the space.
- 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.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n}$ of their volume.
- This is an exponential contraction, so

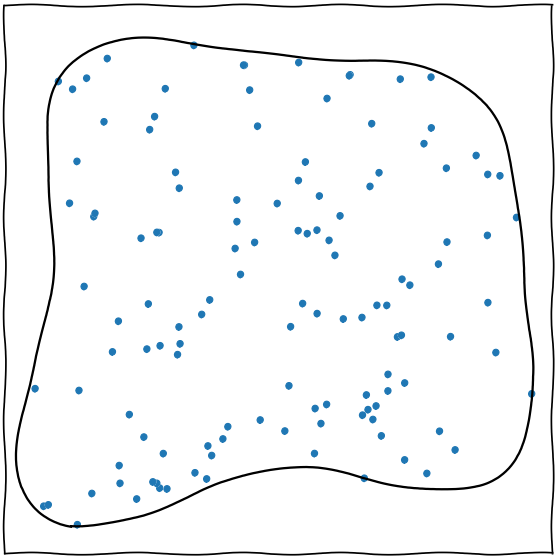
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



The nested sampling meta-algorithm: live points

- Start with n random samples over the space.
- 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.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n}$ of their volume.
- This is an exponential contraction, so

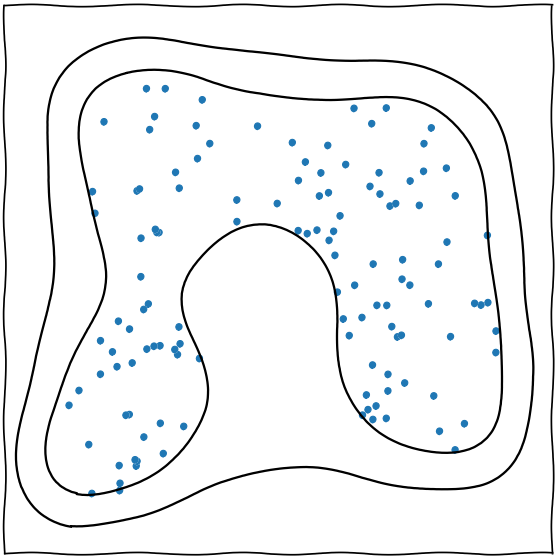
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



The nested sampling meta-algorithm: live points

- Start with n random samples over the space.
- 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.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n}$ of their volume.
- This is an exponential contraction, so

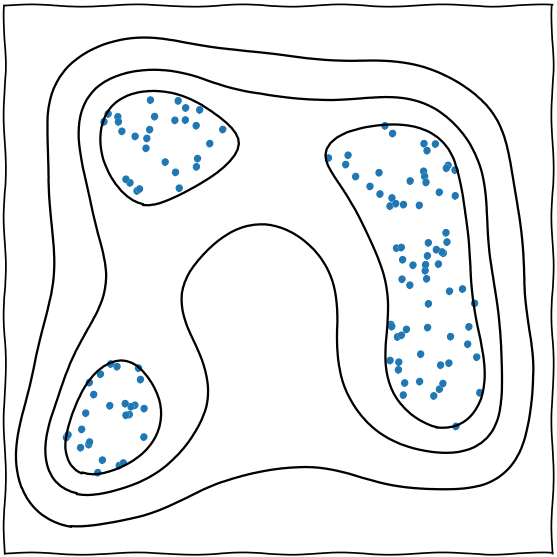
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



The nested sampling meta-algorithm: live points

- Start with n random samples over the space.
- 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.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n}$ of their volume.
- This is an exponential contraction, so

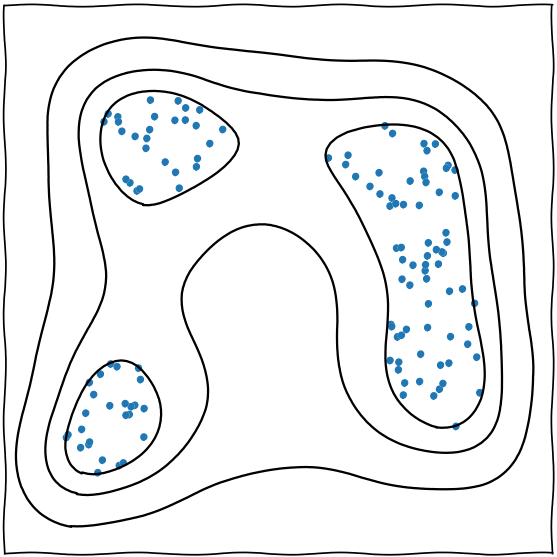
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



The nested sampling meta-algorithm: live points

- Start with n random samples over the space.
- 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.
- We can use this evolution to estimate volume *probabilistically*.
- At each iteration, the contours contract by $\sim \frac{1}{n} \pm \frac{1}{n}$ of their volume.
- This is an exponential contraction, so

$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-(i \pm \sqrt{i})/n}$$



Types of nested sampler

- 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.

- Nested samplers usually come with:
  - *resolution* parameter n_{live} (which improve results as $\sim \mathcal{O}(n_{\text{live}}^{-1/2})$).
  - set of *reliability* parameters, which don't improve results if set arbitrarily high, but introduce systematic errors if set too low.
  - e.g. `Multinest` efficiency `eff` or `PolyChord` chain length n_{repeats}.