Sampling techniques in high-dimensional parameter spaces

with ScannerBit 2.0

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

23rd Feb 2024


UNIVERSITY OF CAMBRIDGE

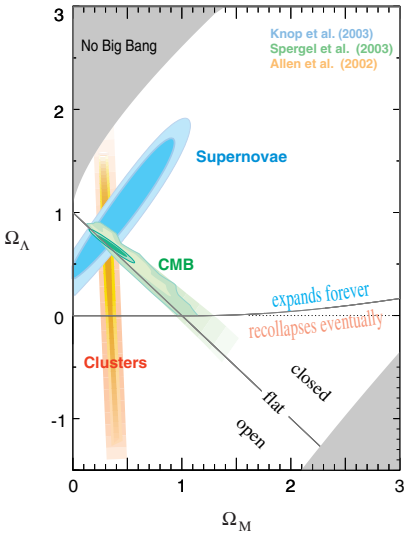


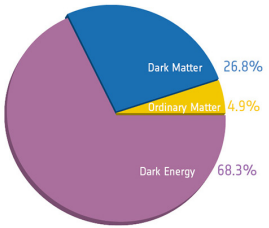
What do I mean by parameter spaces?

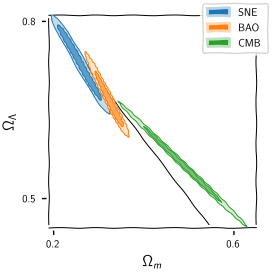
- In science we build models M to describe data D
- These models usually come with a vector of tunable unknown parameters θ
- The general task of science is to use data to define regions of "good" θ
- This talks considers the (increasingly common) case where θ is high-dimensional
  - High-dimensional: $d \geqslant 4$
- Let's look at some concrete examples across fields

Cosmology & Astrophysics

- Cosmological parameters:
  - matter fraction Ω_m
  - dark matter Ω_b
  - dark energy Ω_Λ
  - Hubble constant H_0
  - Initial conditions (A_s, n_s)
  - optical depth τ
- LIGO/Virgo/Kagra event:
  - masses m_1, m_2
  - spins χ_1, χ_2
  - sky position (θ, ϕ)
  - orientation (ι, ψ)
  - distance d_L
  - time & phase t_c, Φ_c
- Bayesian framework: $P(\theta|D)$ defines "credible regions"



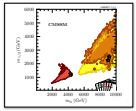




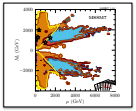
Particle physics

- The standard model has a several physical/phenomenological parameters
- Frequentist framework: $P(D|\theta)$ defines "confidence intervals"
- topical: pMSSM ATLAS scans $d = 19$

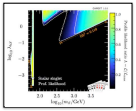
Param.	Definition	Range	Sampling
M_A	mass of pseudoscalar Higgs boson	100 GeV - 25 TeV	Log
$\tan\beta$	ratio of Higgs vevs	1 - 60	Log
$\lvert\mu\rvert$	Higgs-higgsino mass parameter	80 GeV - 25 TeV	Log
$\lvert M_1\rvert$	bino mass parameter	1 GeV - 25 TeV	Log
$\lvert M_2\rvert$	wino mass parameter	70 GeV - 25 TeV	Log
M_3	gluino mass parameter	200 GeV - 50 TeV	Linear
$m_{\tilde{L}^{1,2}}$	1st, 2nd gen. left-handed slepton mass	90 GeV - 25 TeV	Log
$m_{\tilde{R}^{1,2}}$	1st, 2nd gen. right-handed slepton mass	90 GeV - 25 TeV	Log
$m_{\tilde{L}^3}$	3rd gen. left-handed slepton mass	90 GeV - 25 TeV	Log
$m_{\tilde{R}^3}$	3rd gen. right-handed slepton mass	90 GeV - 25 TeV	Log
$m_{\tilde{q}^{1,2}}$	1st, 2nd gen. left-handed squark mass	200 GeV - 50 TeV	Linear
$m_{\tilde{u}^{1,2}}$	1st, 2nd gen. right-handed u-type squark mass	200 GeV - 50 TeV	Linear
$m_{\tilde{d}^{1,2}}$	1st, 2nd gen. right-handed d-type squark mass	200 GeV - 50 TeV	Linear
$m_{\tilde{q}^3}$	3rd gen. left-handed squark mass	100 GeV - 50 TeV	Linear
$m_{\tilde{u}^3}$	right-handed stop quark mass parameter	100 GeV - 50 TeV	Linear
$m_{\tilde{d}^3}$	right-handed sbottom quark mass parameter	100 GeV - 50 TeV	Linear
$\lvert A_\tau\rvert$	τ trilinear coupling	1 GeV - 7 TeV	Log
$\lvert A_b\rvert$	bottom trilinear coupling	1 GeV - 7 TeV	Log
$\lvert A_t\rvert$	top trilinear coupling	1 GeV - $3(m_{\tilde{q}^3} m_{\tilde{u}^3})^{1/2}$	Log

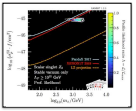


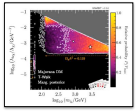
GUT-scale SUSY: 1705.07935

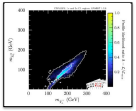


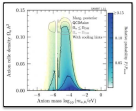
MSSM7: 1705.07917

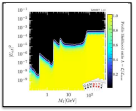

Scalar Higgs portal DM: 1705.07931

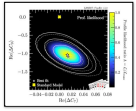

Scalar Higgs portal DM w/ vac. stability: 1806.11281

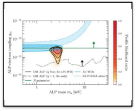

Vector and fermion Higgs portal DM: 1808.10465

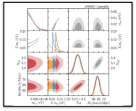

EW-MSSM: 1809.02097

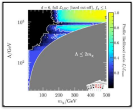

Axion-like particles: 1810.07192

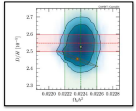

Right-handed neutrinos: 1908.02302

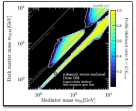

Flavour EFT: 2006.03489

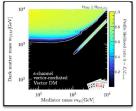

More axion-like particles: 2007.05517

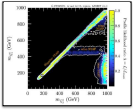

Neutrinos and cosmo: 2009.03287


Dark matter EFTs: 2106.02056


Cosmo ALPs: 2205.13549

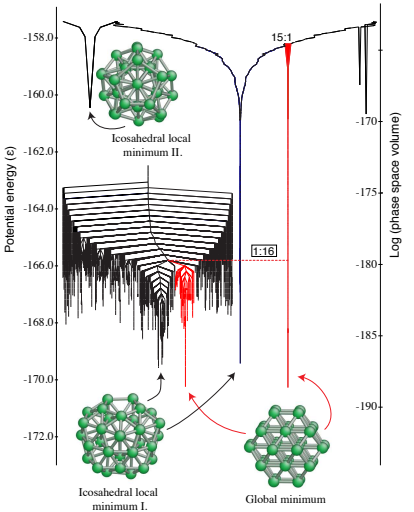

Simplified DM, scalar/fermion: 2009.13268


Simplified DM, vector: 2303.08351


EW-MSSM w/ light gravitino: 2303.09082

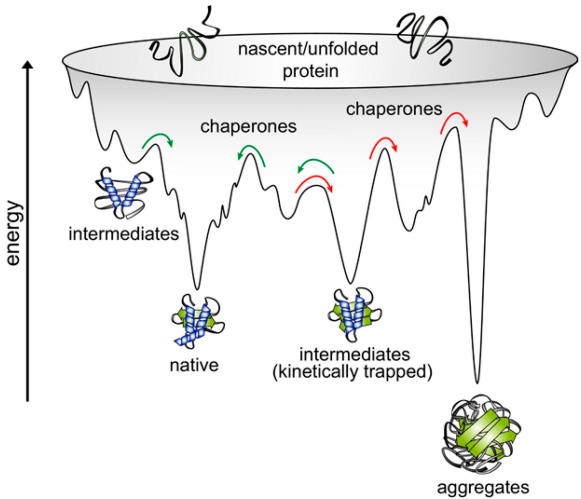
Materials/Chemistry

- Here the relevant function is the free energy $F(\theta)$ as a function of system configuration (bond angles, bond lengths).
- Naturally very high dimensional $d \sim \mathcal{O}(10^2 \rightarrow 10^5)$.
- Complicated parameter spaces with many solutions of interest
- Boltzmann distribution $P(\theta) = e^{-\beta E(\theta)}$ defines the thermodnamic ensemble.
- Added challenge of temperature free parameter, meaning ideally one wants $Z(\beta) = e^{-\beta F(\beta)} = \int P(\theta|\beta) d\theta$



Biophysics

- A specific molecular example is protein molecules
- Alphafold made waves in this space in 2022
- Essentially a powerful machine learning optimisation solution
- Full problem is far from solved!
- Ideally one would compute misfolds & partition functions



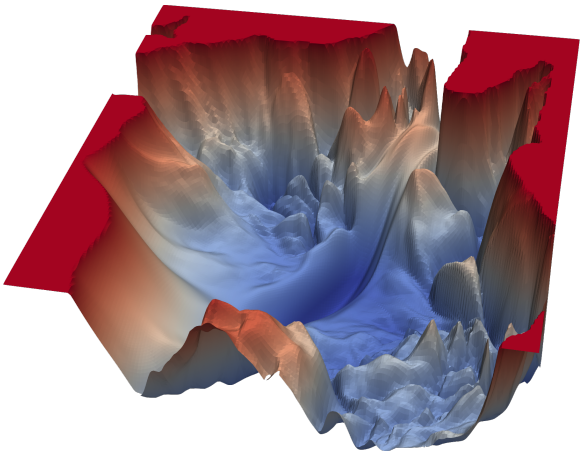
Machine learning

- Standard machine learning training setup:

$$\hat{\theta} = \arg\min_{\theta} L(\theta)$$

$$L(\theta) = \min_{\theta} \sum_{i \in \text{train}} |f(x_i; \theta) - y_i|^2$$

  - x: input data
  - y: target data
  - f: neural network
  - θ: model parameters (weights and biases)
- The "Loss function" L of the training/testing data defines the "good" θ
- Usually optimised with gradient descent (adam).



losslandscape.com

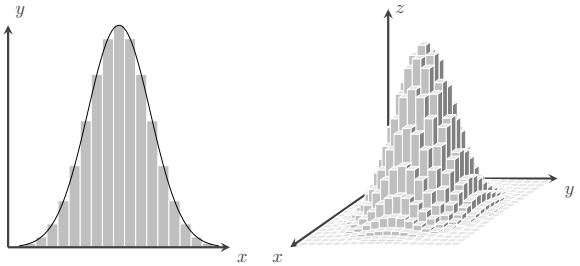
Why is high-dimensional $d \geqslant 4$?

- 4D is the crossover where the "curse of dimensionality" kicks in.
- We can see this effect in practiceif we consider trying to do a 1, 2, 3 & 4D integral numerically.
- Integration grinds to a halt as the number of hyperboxes in the grid becomes exponntially large.
- Visualisation requires a 2D projection
- 2D has very specific properties which do not lend themselves well to intuiting the behaviour of the full parameter space

```python
import numpy as np
from scipy.integrate import nquad

def f(*x):
    x = np.array([*x])
    return np.exp(-(x**2).sum())

nquad(f, [[-5, 5]])      # 322  micro s
nquad(f, [[-5, 5]] * 2) # 33.4 ms
nquad(f, [[-5, 5]] * 3) # 3.4  s
nquad(f, [[-5, 5]] * 4) # ...
```

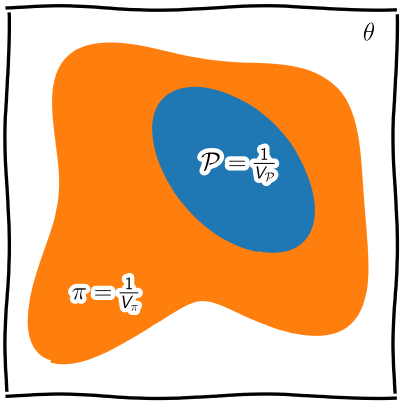


Observation #1: The typical set is small

- The non-zero region of the function (the typical set) occupies a tiny fraction of the space
- From a 1D/2D projected view, this is not obvious
- If each parameter has a compressed region $\delta\theta_i$ relative to the initial bounds $\Delta\theta_i$, then the total fraction of the compressed region is $\prod_i^d \frac{\delta\theta_i}{\Delta\theta_i}$ which is exponential in the number of parameters d
- This is generalised by the KL divergence:

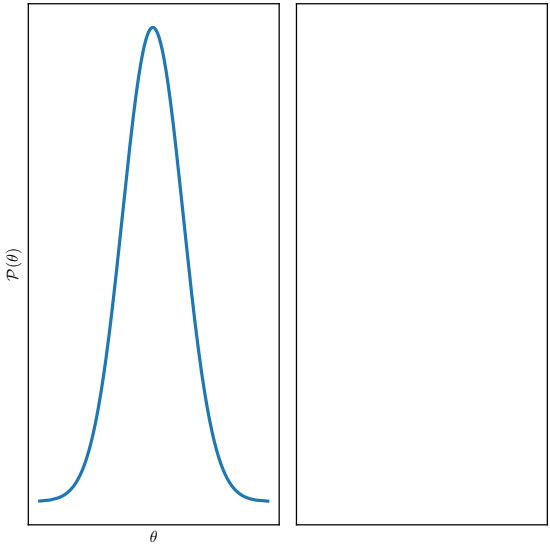
$$\mathcal{D}_{\mathrm{KL}} = \int \mathcal{P}(\theta) \log \frac{\mathcal{P}(\theta)}{\pi(\theta)} d\theta \sim \log \frac{V_\pi}{V_\mathcal{P}}$$

- e.g. $6d$ cosmological Planck ΛCDM $e^{-\mathcal{D}_{\mathrm{KL}}} \sim 10^{-14}$



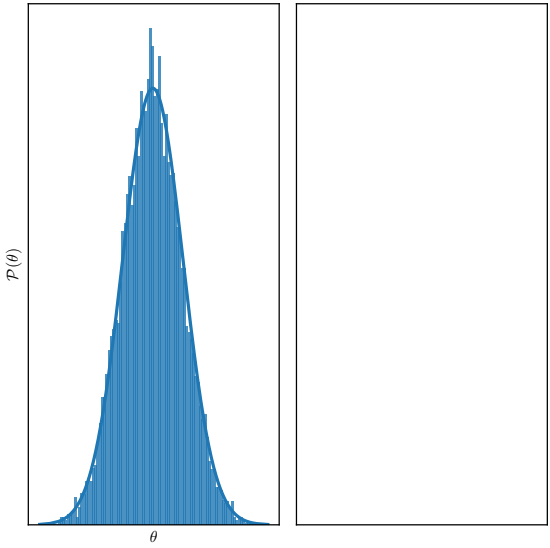
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



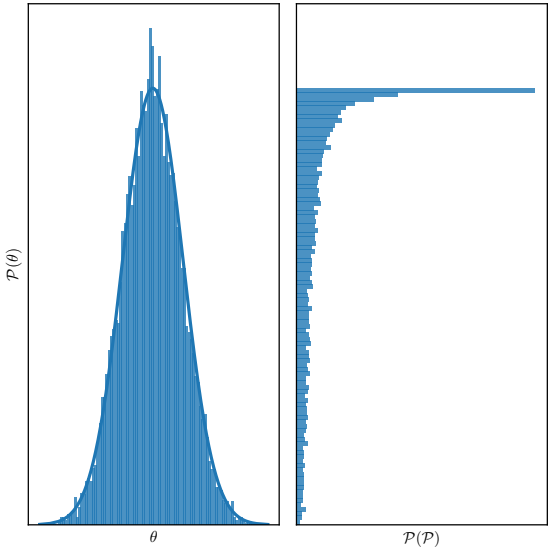
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



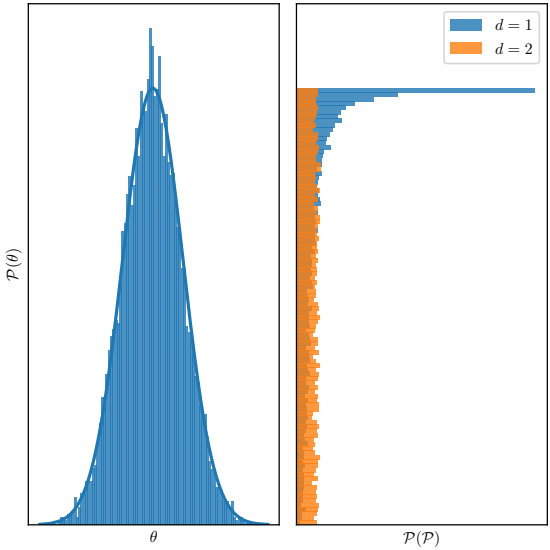
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



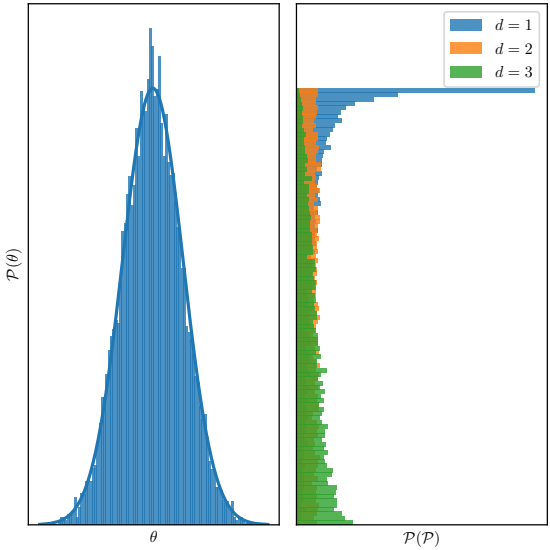
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



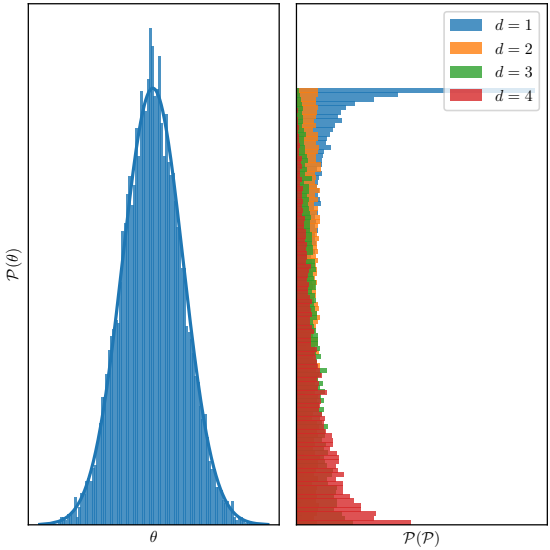
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



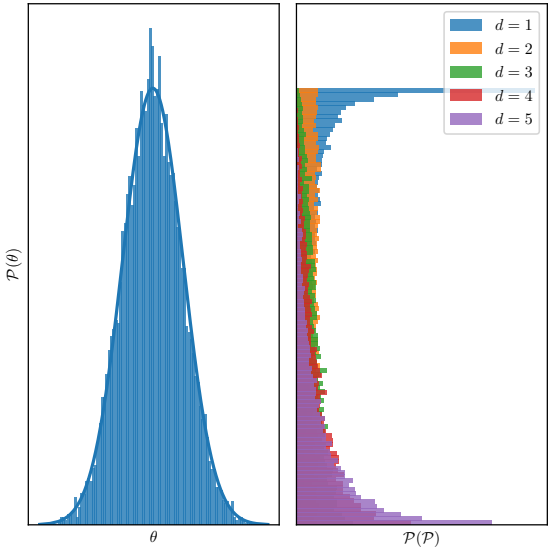
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



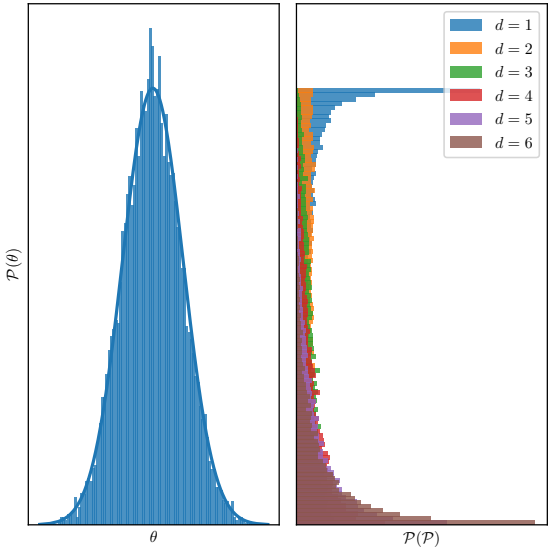
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



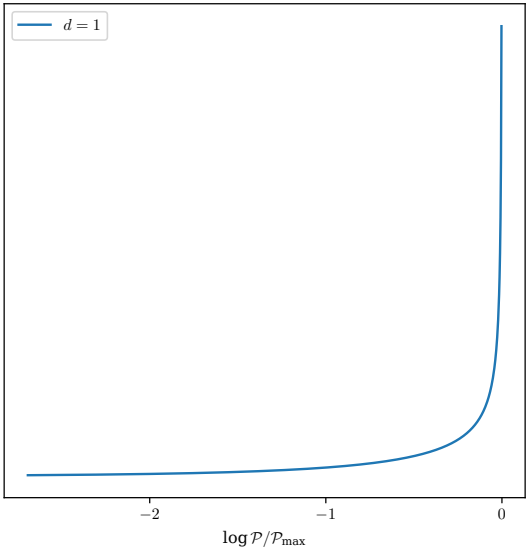
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



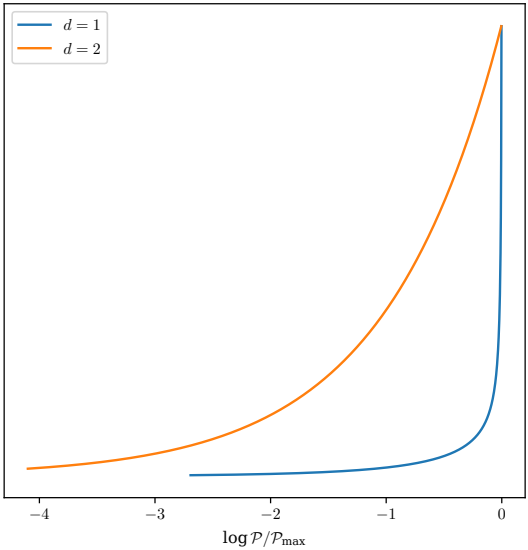
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



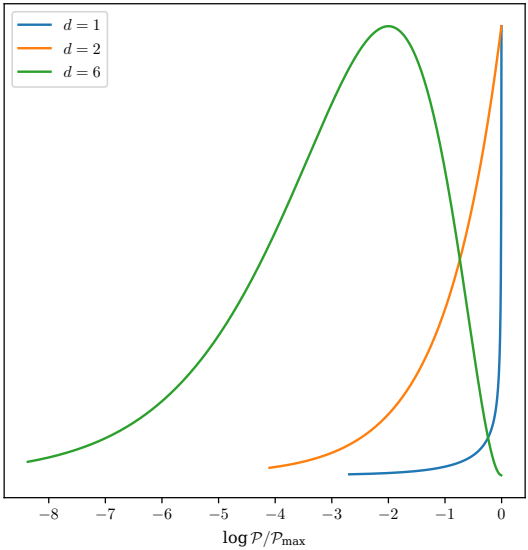
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



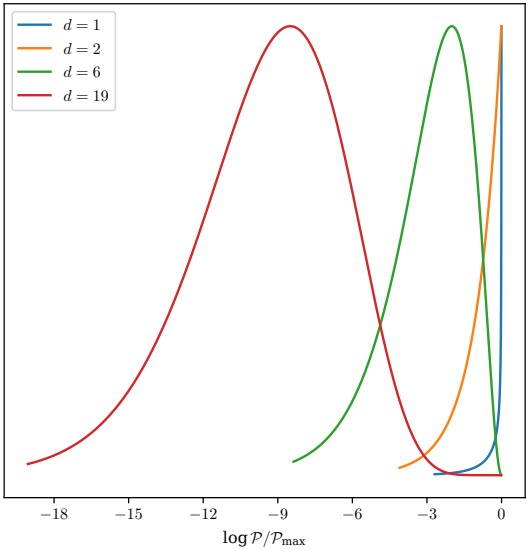
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



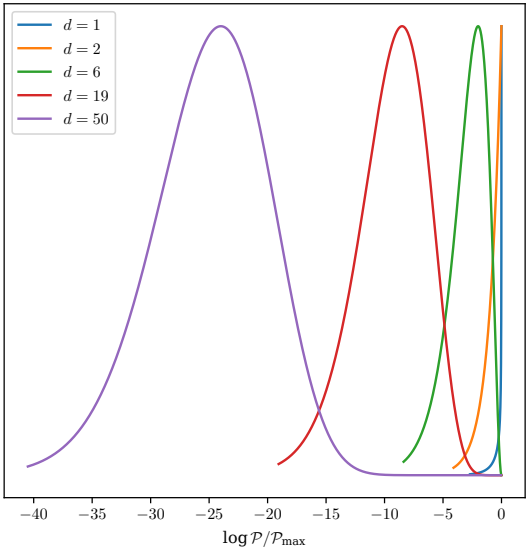
Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



Observation #2: The typical set is not at the peak

- In low dimensions, the "best fit point" is in the "best fit region".
- This is not true in high dimensions.
- This can be thought of as a balance of:
  - probability density and volume
  - energy and entropy
  - likelihood and prior



Principles of parameter exploration

1. **What do I want to do?** *(Optimise/Profile/Sample)*
   - Optimisation can be done with very few function calls
   - exploration algorithms require more: $\mathcal{O}(10^4 \to 10^8)$ in practice
2. **How long does each function call take?** *(Microseconds/Seconds/Days)*
   - The first can be done on a laptop, the second on HPC, the third restricts one to optimisation
3. **How high dimensional is the problem?** *($4 \to 10$, $10 - 500$, $500 - 10^6$, $> 10^6$)*
   - rejection/importance algorithms up to $\sim$10 dimensions (normalising flows, multinest, VEGAS)
   - many statistical approaches can work in 100s (Gibbs, Slice, NS)
   - Beyond that HMC is king up to millions.
   - Current deep learning record training is 530 billion parameters.
4. **Do I have the gradient of the function?**
   - autodiff from differentiable programming languages (`jax`,`pytorch` etc) makes this more common
   - some things are not differentiable even in principle
5. **Is the function an energy≡log probability?**
   - Sampling is well defined for energies
   - If you're optimising something non-probabilistic e.g. financial return, resolution of beam etc then it is more involved (but not impossible) to turn this into a probabilistic problem.

Exploring parameter spaces

Option 1 Optimize

- Solve $\theta = \arg\max_\theta f(\theta)$

Option 2 Sample

- Generate a set of samples $\{\theta_i\}$ from the distribution $\mathcal{P}(\theta)$
- Works if your function is a probability in θ e.g. a Bayesian posterior $P(\theta|D)$

Option 3 Profile

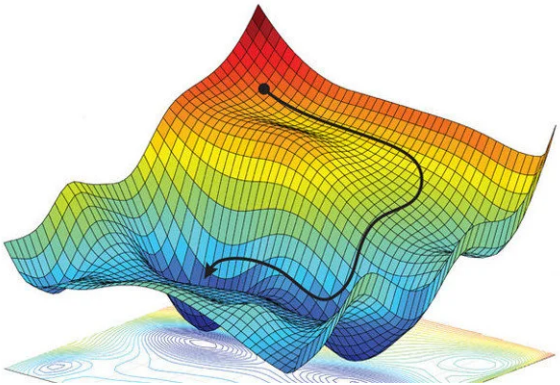
- Solve $f_i = \max_{\theta_{j\neq i}} f(\theta)$ for a range of θ_i
- i.e. fix one parameter, and maximise over the rest, to give a 1D profile
- Also works for 2D profiles.

Optimisation: Gradient descent (for highest dimensions)

- The first thing one thinks of.
- Fastest class of algorithms (used in deep learning e.g. Adam)
- Can get stuck in local minima
- Finds a maximum, but hard to use the samples it's calculated along the way.
- Nontrivial consideration:

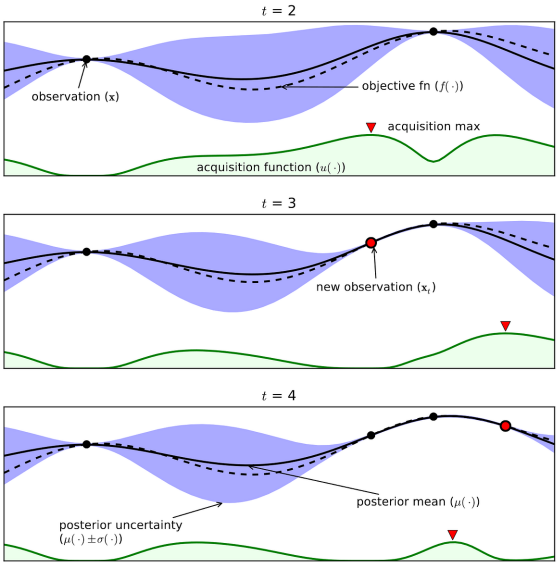
  "While finding the gradient of an objective function is a splendid idea, ascending the gradient directly may not be." (MacKay)

  (Most gradient descent algorithms are not covariant out-of-the-box).



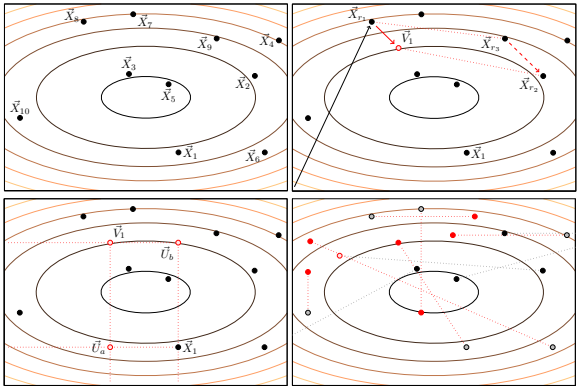
Optimisation: Bayesian optimisation (for expensive functions)

- Best approach for punitively expensive functions
- Uses a Gaussian process to model the function, and propose the optimal new point which achieves an objective (e.g. maximisation), taking into account all uncertanties
- These are used to e.g. optimise hyperparameters in machine learning/N body simulations



Optimisation: Genetic Algorithms (for profiling)

- Algorithms use an evolutionary process:
  - Generate a population of points
  - Reproduce: Recombine, mutate
  - Kill off low ranking points
- `diver` is the genetic algorithm for differential evolution preferred by GAMBIT
- Generally accepted as being good for solving multimodal functions
- Not statistically principled or interpretable if it goes wrong
- These in practice are good for profiling in medium dimensions, since the evolutionary process generates many good candidate points – but technically a global optimiser!

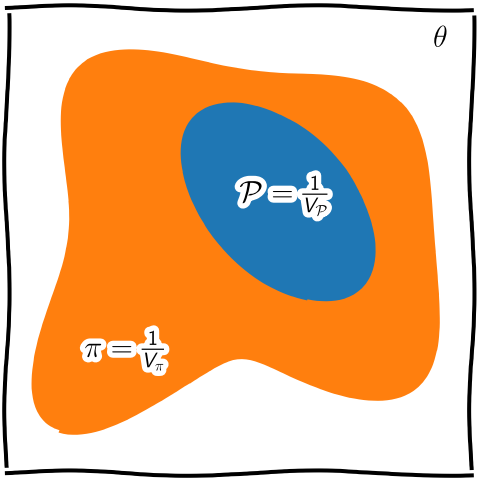


Why do sampling?

- The cornerstone of numerical Bayesian inference is working with samples.
- Generate a set of representative parameters drawn in proportion to the posterior $\theta \sim \mathcal{P}$.
- The magic of marginalisation $\Rightarrow$ perform usual analysis on each sample in turn.
- The golden rule is stay in samples until the last moment before computing summary statistics/triangle plots because

$$\boxed{f(\langle X \rangle) \neq \langle f(X) \rangle}$$

- Generally need $\sim \mathcal{O}(12)$ independent samples to compute a value and error bar.

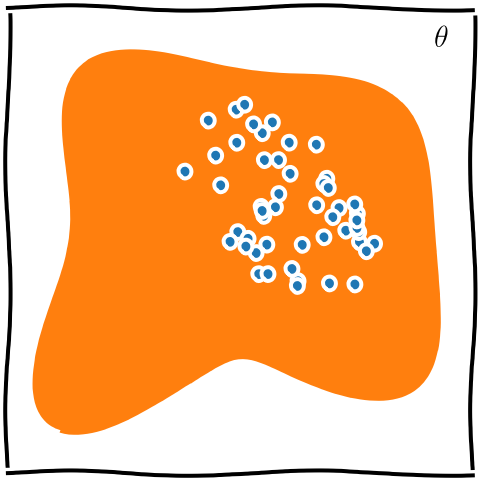


Why do sampling?

- The cornerstone of numerical Bayesian inference is working with samples.
- Generate a set of representative parameters drawn in proportion to the posterior $\theta \sim \mathcal{P}$.
- The magic of marginalisation $\Rightarrow$ perform usual analysis on each sample in turn.
- The golden rule is stay in samples until the last moment before computing summary statistics/triangle plots because

$$\boxed{f(\langle X \rangle) \neq \langle f(X) \rangle}$$

- Generally need $\sim \mathcal{O}(12)$ independent samples to compute a value and error bar.



Random scanning: 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 sublinear $\mathcal{D}$ scaling with d.
- Turning this around, you can use the inefficiency of random sampling to estimate $\mathcal{D}$
- *"Exploring phase space with Nested Sampling"* Handley, **Janßen**, Schumann & **Yallup** [2205.02030] Also **Carragher** et al [2101.00428]

Metropolis Hastings (baseline sampling algorithm)

- 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 (ultra high dimensional sampling)

- 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 already normalised (physicists naturally do this, which explains why it works so well out of the box).

TOOLS

- `stan` is a fully fledged, mature programming language with HMC as a default sampler.
- `TensorFlow`, `PyTorch` & `JAX` all have HMC implementations.

Ensemble sampling (robust alternative to MH)

- 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

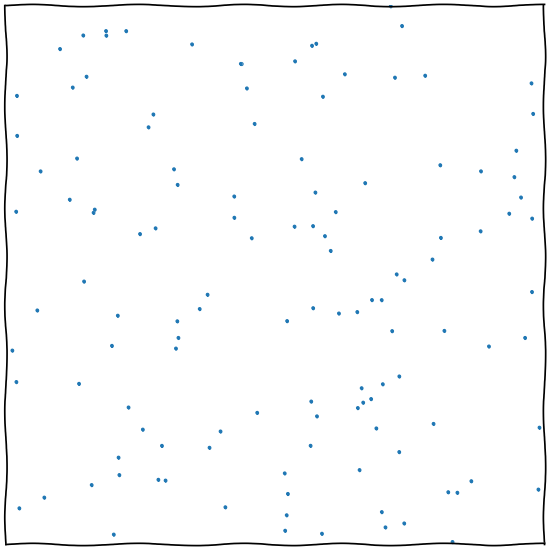
TOOLS

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

Nested sampling (high dimensional integration)

- 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

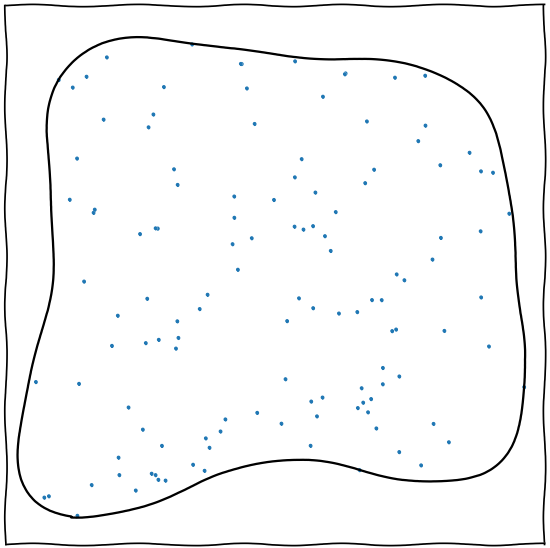
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



Nested sampling (high dimensional integration)

- 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

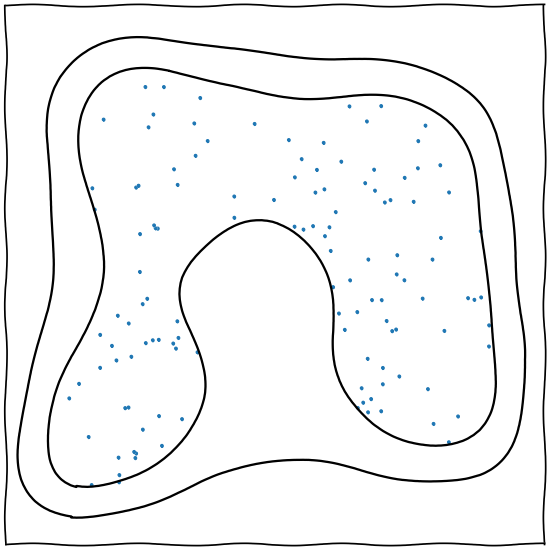
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



Nested sampling (high dimensional integration)

- 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

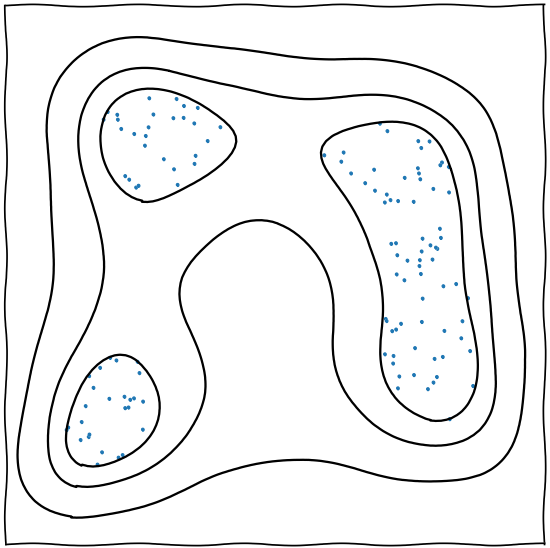
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



Nested sampling (high dimensional integration)

- 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

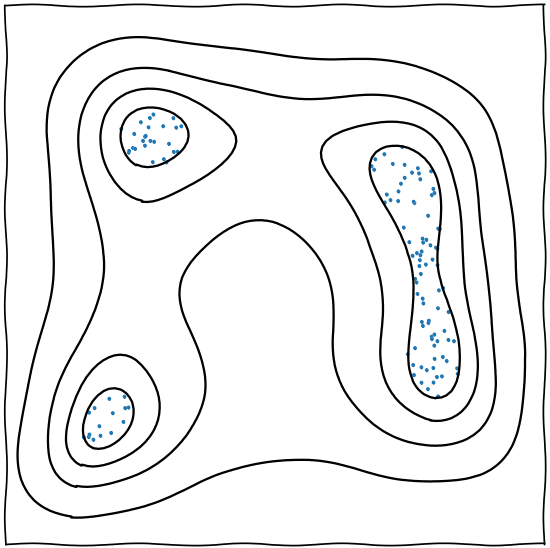
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



Nested sampling (high dimensional integration)

- 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

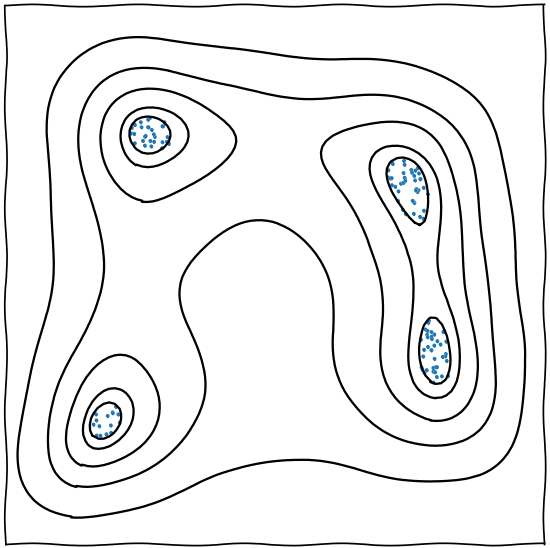
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



Nested sampling (high dimensional integration)

- 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

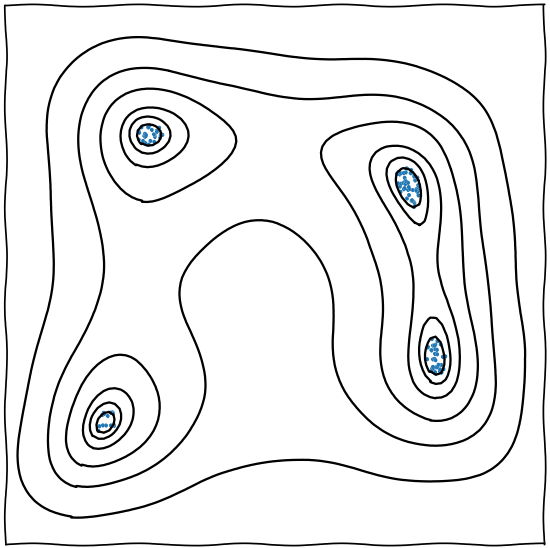
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



Nested sampling (high dimensional integration)

- 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

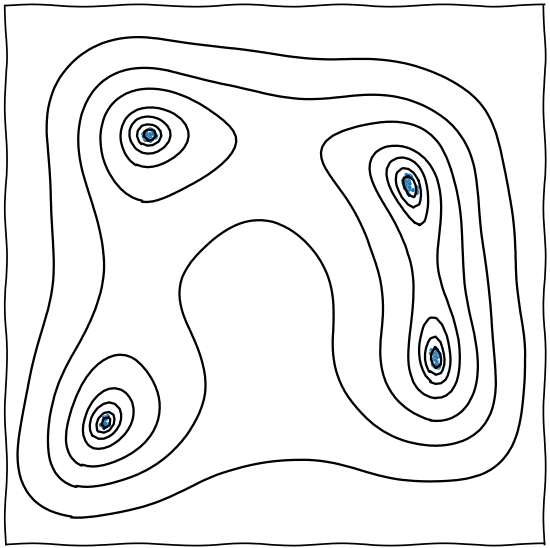
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



Nested sampling (high dimensional integration)

- 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

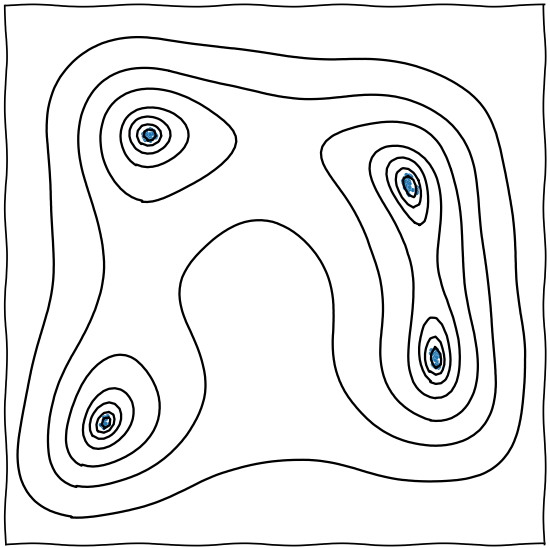
$$\int f(x)dV \approx \sum_i f(x_i)\Delta V_i, \quad V_i = V_0 e^{-i/n}$$



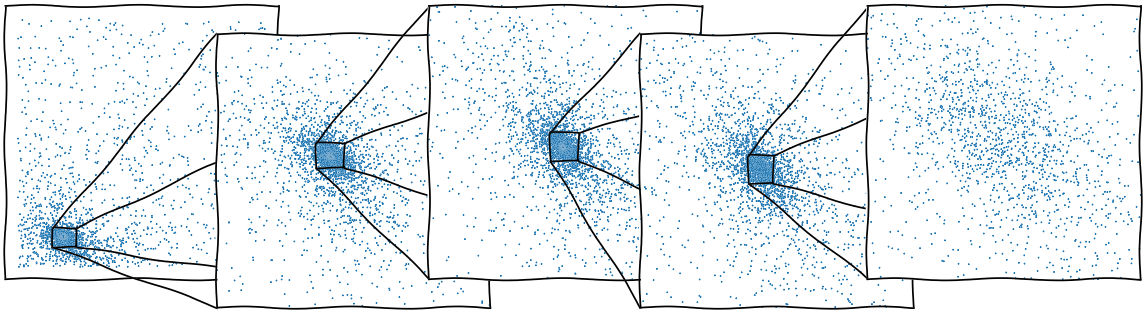
Nested sampling (high dimensional integration)

- 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

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



The dead measure [2312.00294]



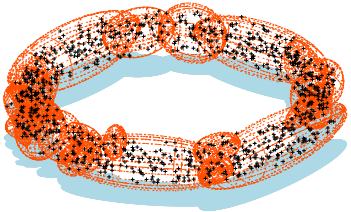
- Dead points have a unique scale-invariant distribution $\propto \frac{dV}{V}$.
- Uniform over original region, exponentially concentrating on region of interest (until termination volume).
- Full coverage of tails enables integration.
- Good for training emulators (HERA [2108.07282]).

Applications

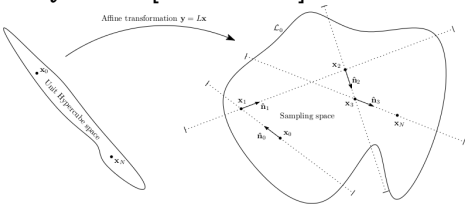
- training emulators.
- gridding simulations
- beta flows...
- good name for a band

Implementations of Nested Sampling [2205.15570](NatReview)

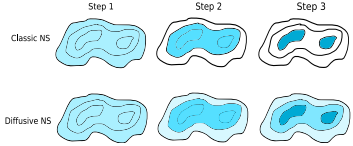
MultiNest [0809.3437]



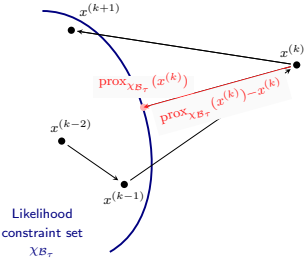
PolyChord [1506.00171]



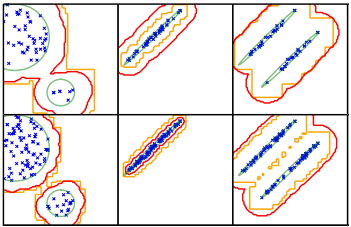
DNest [1606.03757]



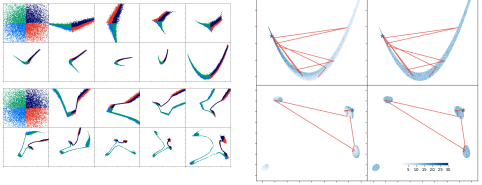
ProxNest [2106.03646]



UltraNest [2101.09604]



NeuralNest [1903.10860]



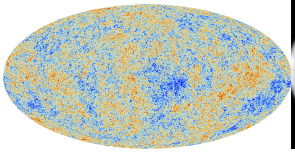
nessai [2102.11056]

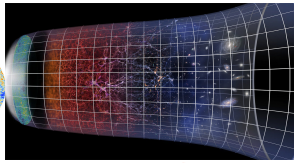
nora [2305.19267]

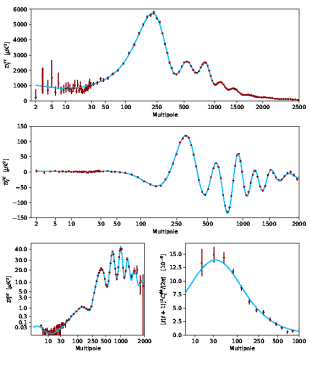
dynesty [1904.02180]

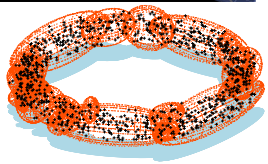
Software frameworks

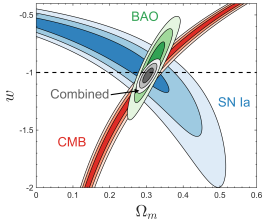
- Principled scientific data analysis requires a lot of different components:
  - Data
  - Theory
  - Statistics
  - Likelihoods
  - Samplers/scanners
  - Post-processing
- Mature communities develop frameworks to do all of the above in one go:
  - Cosmology: `CosmoMC`, `MontePython`, `Cobaya`, `CosmoSIS`, `CosmoLike`
  - Gravitational Waves : `LAL`, `Bilby`, `PyCBC`
  - Particle physics: `GAMBIT`, `MasterCode`
- Much harder if you want to combine data from multiple communities











GAMBIT: The Global And Modular BSM Inference Tool

gambit.hepforge.org github.com/GambitBSM EPJC 77 (2017) 784 arXiv:1705.07908

- Extensive model database, beyond SUSY
- Fast definition of new datasets, theories
- Extensive observable/data libraries
- Plug&play scanning/physics/likelihood packa
- Various statistical options (frequentist /Bayesian)
- Fast LHC likelihood calculator
- Massively parallel
- Fully open-source

Members of: ATLAS, Belle-II, CLiC, CMS, CTA, Fermi-LAT, DARWIN, IceCube, LHCb, SHiP, XENON

Authors of: BubbleProfiler, Capt'n General, Contur, DarkAges, DarkSUSY, DDCalc, DirectDM, Diver, EasyScanHEP, ExoCLASS, FlexibleSUSY, gamLike, GM2Calc, HEPLike, IsaTools, MARTY, nuLike, PhaseTracer, PolyChord, Rivet, SOFTSUSY, Superlso, SUSY-AI, xsec, Vevacious, WIMPSim

Recent collaborators: V Ananyev, P Athron, N Avis-Kozar, C Balázs, A Beniwal, LL Braseth, T Bringmann, A Buckley, J Butterworth, JE Camargo-Molina, C Chang, J Cornell, M Danninger, A Fowlie, T Gonzalo, W Handley, S Hoof, A Jueid, F Kahlhoefer, A Kvellestad, M Lecroq, C Lin, M Lucente, FN Mahmoudi, DJE Marsh, G Martinez, H Pacey, MT Prim, T Procter, F Rajec, A Raklev, R Ruiz, A Scaffidi, P Scott, W Shorrock, C Sierra, P Stöcker, W Su, J Van den Abeele, A Vincent, M White, A Woodcock, Y Zhang ++

70+ participants in many experiments and numerous major theory codes

What is GAMBIT?

GAMBIT as a software framework

- Combines collider, direct detection, neutrino & telescope data.
- Allows joint analysis of dark matter, neutrinos & BSM physics
- MPI + OpenMP parallelisation (record of 115,000 CPUs)
- Combines libraries & codes written in: `C++`, `Fortran`, `Python`, `Mathematica`...
- Highly modular "Bits"
- Often with several alternatives

GAMBIT as a community

- Particle physicists, cosmologists & statisticians (>80 members)
- Generates interdisciplinary expertise & inspires new techniques
- Open source software
- Access to large community computing resources (40MCPUh/y)
- Modularity allows parallel teams
- Short-author papers by default
- in-person meetings every 9 months

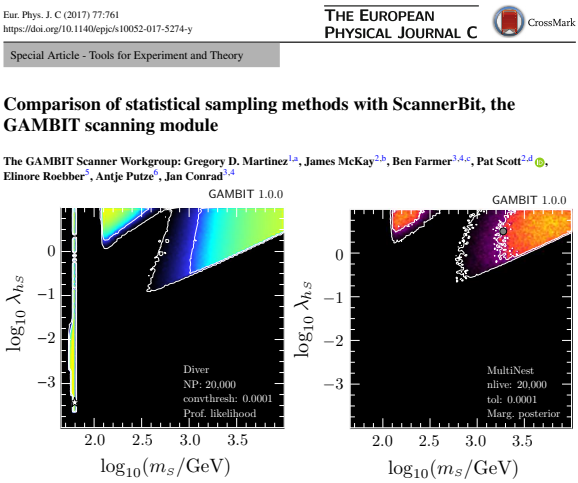
What is GAMBIT not?

- GAMBIT is not pip-installable
- It should be viewed as a "power tool" for heavy tasks
- Designed with HPC in mind
  - expectation that forward modelling for frontier science takes seconds to hours
- Combining data, theory & code across multiple communities is non trivial
- Installing on a new system takes patience
  - Stickers for newcomers!

ScannerBit [1705.07959]

Original paper:

- describes code design & structure
- compares performance of the main two scanning algorithms: `diver` & `MultiNest`
  - also compared `Great`(MH) and `Twalk`(affine invariant)
- Discusses Frequentist & Bayesian analyses
- 15D MSSM scalar singlet (2+13 parameters)
- Modular & flexible plugins homogenise scanning algorithms (agnostic of model/prior)



ScannerBit 2.0

Developments since original ScannerBit paper:

1. New scanning algorithms
   - `MINUIT2`
   - `PolyChord`
   - particle swarm
2. `PyScannerBit`
   - Isolation of scanners from GAMBIT
   - documentation: pyscannerbit.readthedocs.io
3. Python Scanners
   - Allows easy incorporation of new scanning algorithms into GAMBIT
   - Modern machine learning techniques
   - (embarassingly) easy to incorporate

- Paper release later this year.
- Beta testers welcome! gambit-scan@projects.hepforge.org

```python
"""scipy.optimize scanner."""
import scanner_plugin as splug
import scipy.optimize
import numpy as np


class Minimize(splug.scanner):
    __version__ = scipy_version

    def __init__(self, **kwargs):
        super().__init__(use_mpi=False,
                         use_resume=False)

    def run(self):
        start = np.random.rand(self.dim)
        bounds = [(0., 1.)] * self.dim

        def neg_loglike_hypercube(x):
            return -self.loglike_hypercube(x)

        res = scipy.optimize.minimize(
                  neg_loglike_hypercube,
                  start, bounds=bounds)

        return 0

__plugins__ = {"scipy_minimize": Minimize}
```

ScannerBit 2.0

Developments since original ScannerBit paper:

1. New scanning algorithms
   - `MINUIT2`
   - `PolyChord`
   - particle swarm
2. `PyScannerBit`
   - Isolation of scanners from GAMBIT
   - documentation: pyscannerbit.readthedocs.io
3. Python Scanners
   - Allows easy incorporation of new scanning algorithms into GAMBIT
   - Modern machine learning techniques
   - (embarassingly) easy to incorporate

- Paper release later this year.
- Beta testers welcome! gambit-scan@projects.hepforge.org

```python
"""Grid scanner with MPI."""
import scanner_plugin as splug
from utils import MPI
import numpy as np

class Grid(splug.scanner):

    __version__="1.0.0"

    def __init__(self, grid_pts=10, parameters=[],
                 **kwargs):
        super().__init__(use_mpi=True, use_resume=False)
        # Set up grid
        for param in self.parameter_names:
            n = grid_pts[parameters.index(param)]
            self.vecs.append(np.linspace(0.0, 1.0, n))

    def run(self):
        # Loop over grid with MPI
        grid = np.meshgrid(*self.vecs)
        grid = grid.reshape(self.dim, -1).T
        grid = grid[self.mpi_rank:self.size:
                    self.mpi_size]
        for pt in grid:
            self.loglike_hypercube(pt)
        return 0

__plugins__={"python_grid": Grid}
```

GAMBIT highlights

- Projects I have been involved in:

  CosmoBit [2009.03286](main paper) [2009.03287](neutrinos)
  DMEFT [2106.02056]
  CosmoALP [2205.13549]

- Projects I'm currently involved in:

  AnnualModulation DAMA/LIBRA, COSINE-100, ANAIS, SABRE comparison + EFT
  SubGeVDM Global analysis of sub-GeV dark matter

- GAMBIT Role: CosmoBit convener

Upcoming GAMBIT developments

- ScannerBit 2.0
  - Next-generation scanning algorithms
- GAMBIT-light
  - Installing GAMBIT framework independent of heavyweight HEP codes
- GravBit
  - Beginning with PTAs
  - Moving on to GW events

Conclusions

- Exploring high dimensional parameter spaces is challenging, and an area of active research
- If you would like to learn more about GAMBIT: gambitbsm.org
- If you would like to find out about joining GAMBIT: email gambit-community@projects.hepforge.org

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()`

A note on adaptive scanning algorithms

Some analyses (e.g. ATLAS) have a legitimate preference for random sampling because it allows parallelised physics computations offline.
Smart sampling algorithms require all likelihood computations on the fly, so need machinery like GAMBIT