Machine-learning Bayesian inference with Gaussian Processes

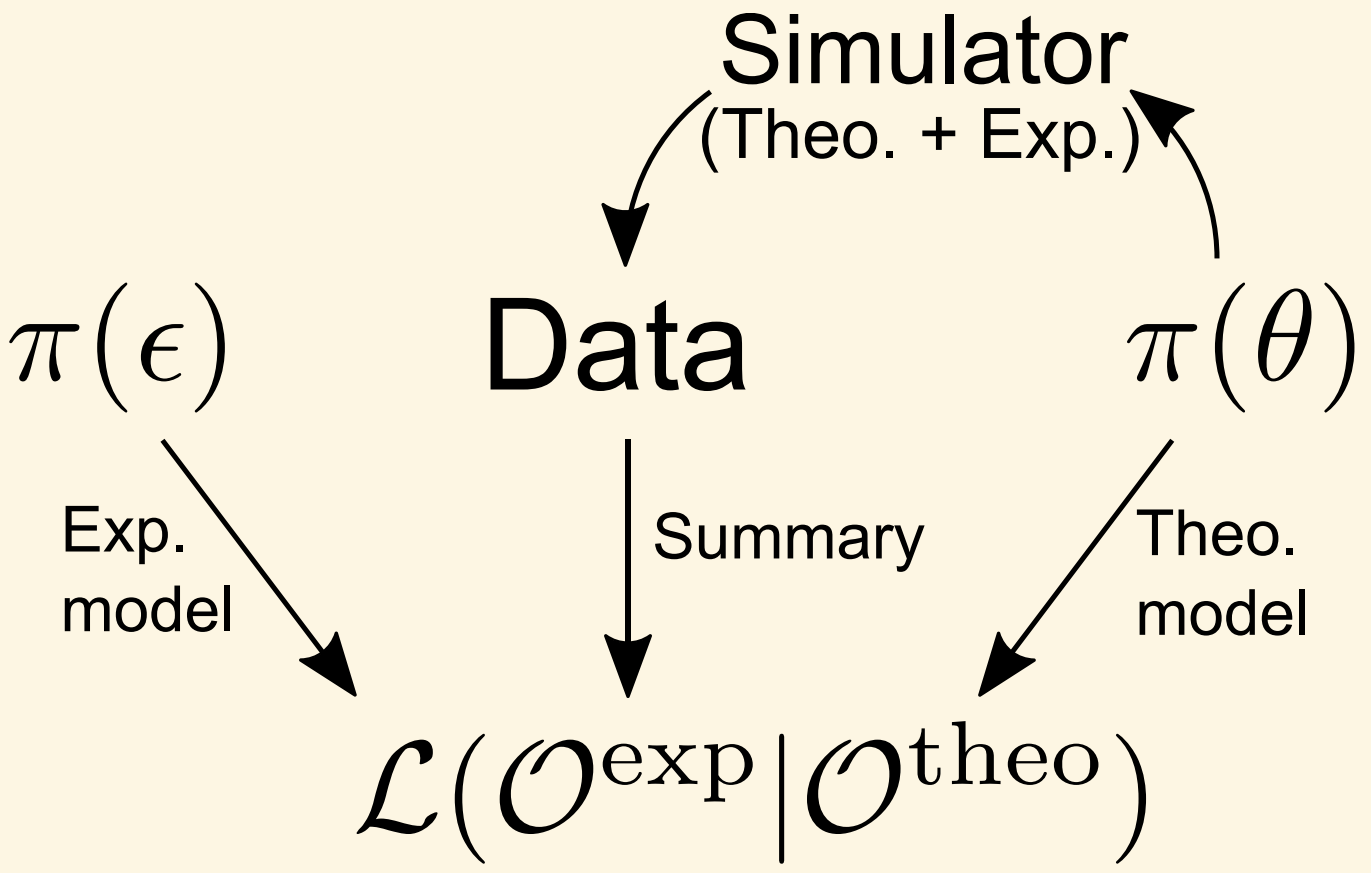
Jesús Torrado

IAU-IAA Seminar Series – 9th May 2023

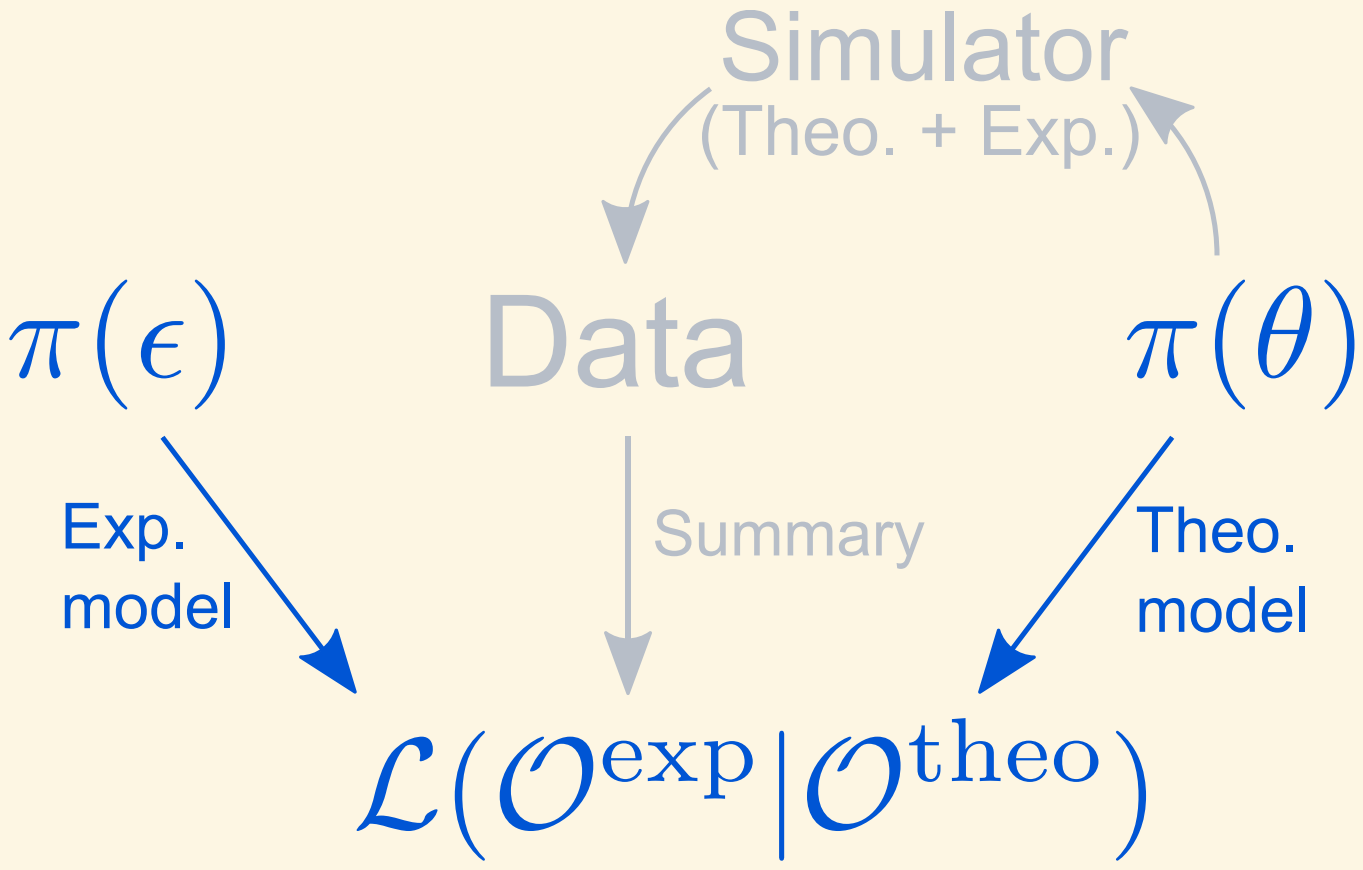
Based on arXiv:2211.02045
with Jonas el Gammal, Nils Schöneberg & Christian Fidler

Università degli Studi di Padova

Elements of an inference problem



Elements of an inference problem



The *traditional Monte Carlo pipeline*

CosmoMC, **Monte Python**, **CosmoSIS**, **Cobaya**...

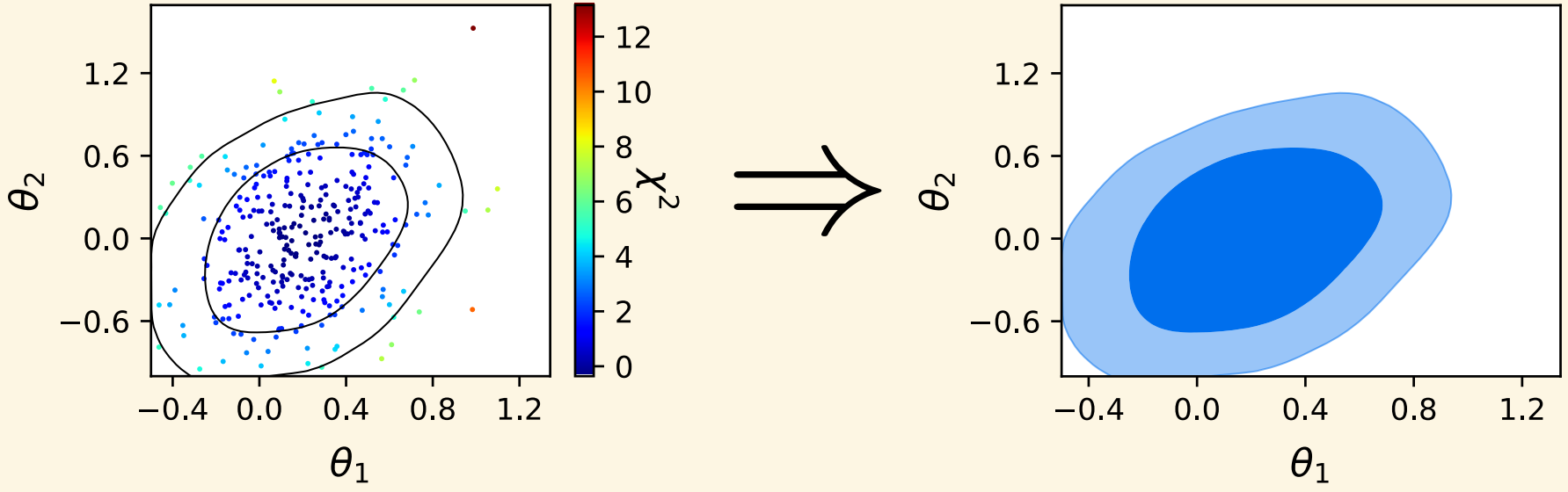
Bayesian parameter inference

So how do we go about *mapping and characterising the posterior*?

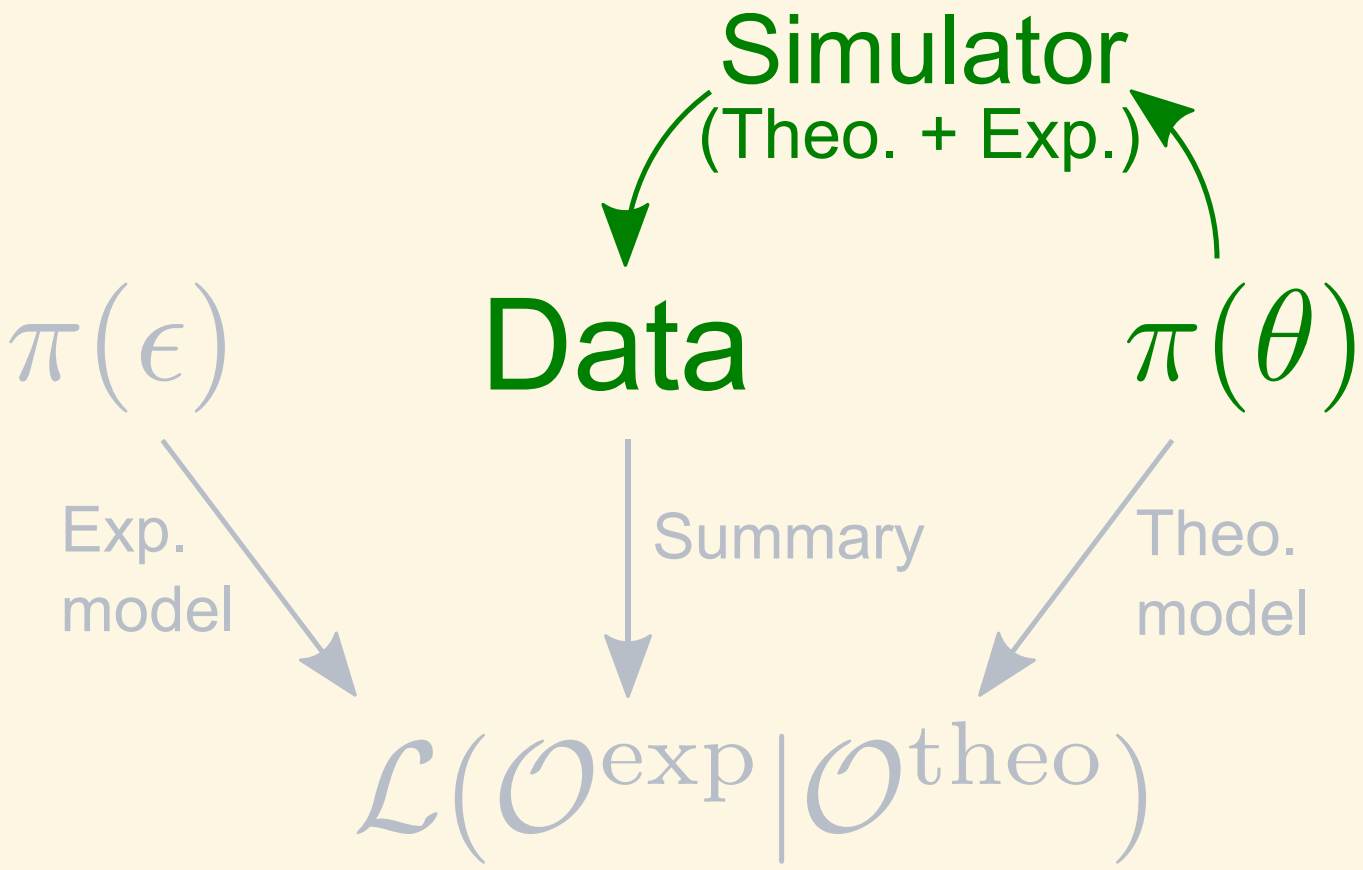
I.e., *where are the maxima are how does probability spread around them*?

This is not trivial for not-so-simple probability distributions.

Simplest way: produce a *sample* from that distribution using a **Monte Carlo** algorithm (MCMC, Nested Sampling...), i.e. a series of points whose density is proportional to the probability in their neighbourhood.



Alternative: forward modelling



***Likelihood-free forward modelling*, usually with ABC:**

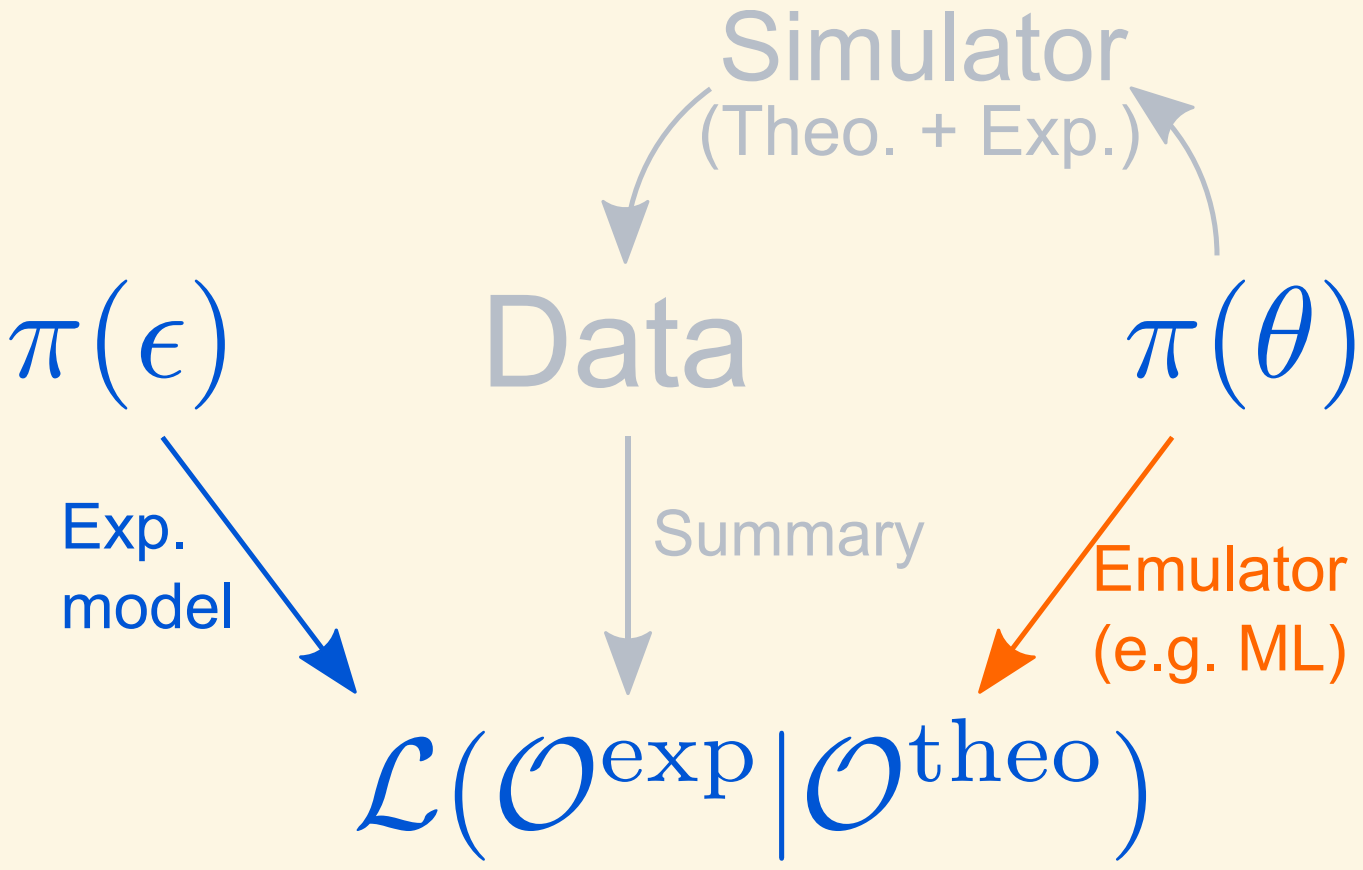
Akeret at el. '15, Jennings & Madigan '16, Kacprzak et al. '17...

Problems of conventional approaches

Traditional likelihood-based (MCMC...) or traditional forward-modelling-based (ABC...) inference need a lot of samples/realisations since they are based on probability estimation from sampling density.

Not feasible in finite time with heaps of data coming our way. We will probably need to reduce the computational cost by at least $\mathcal{O}(10^2)$, by using **emulation** (ML-based) at some level, and heavily rely on **parallelisation** (incl. other types such as GPU).

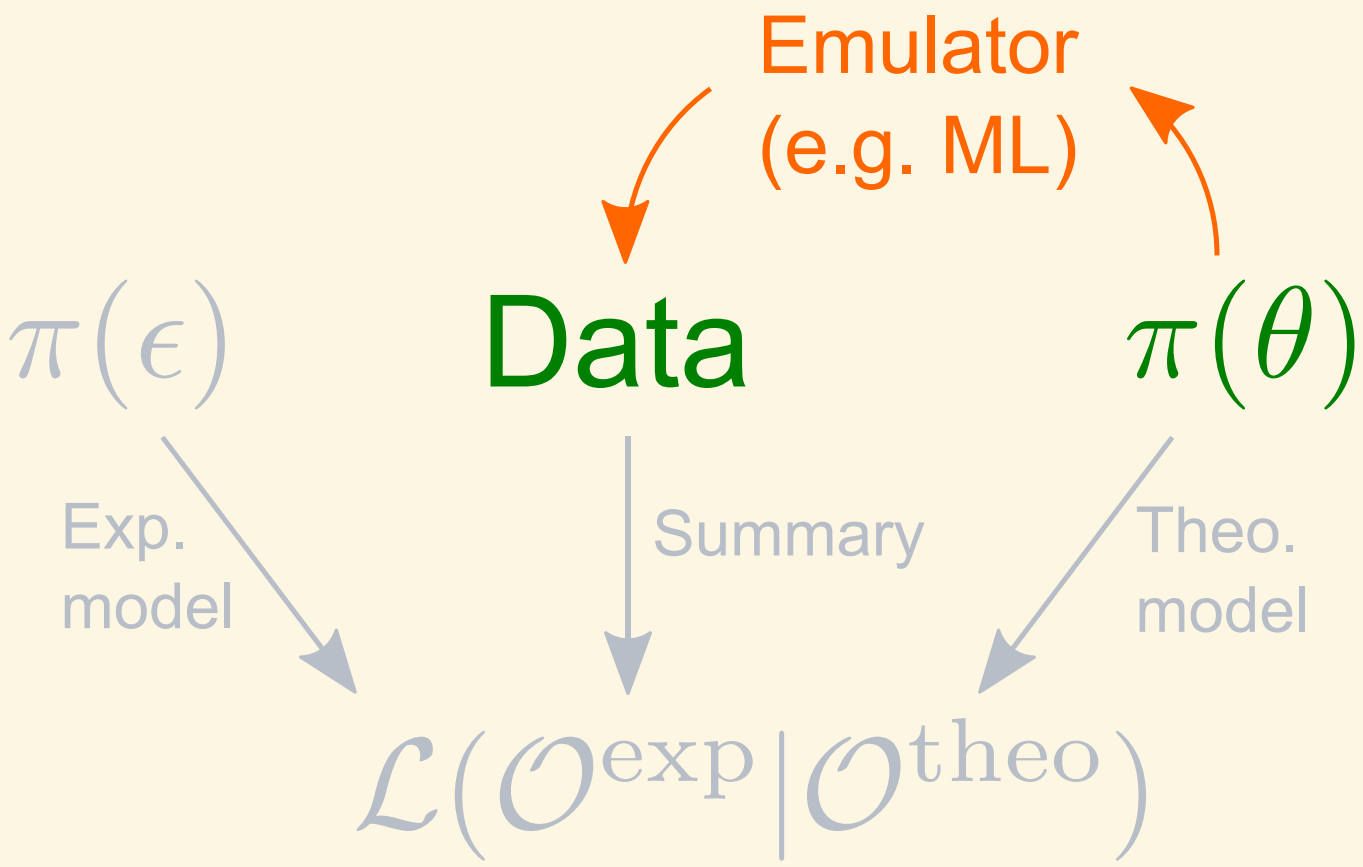
The ML emulator path



Emulation **of the observables (compatible with trad. pipeline)**

Fendt & Wandelt '06, Manrique-Yus & Sellentin ' 19, Albers et al. '19, Mancini et al. '21, Mootoovaloo et al. '20...

The ML forward-modeling path



***Likelihood-free forward modelling, ML-enhanced*:**

Leclercq '18, Alsing et al. '19, Miller et al. '20, Day & Seljak '22 (in this seminar series: Nisimichi, Villaescusa-Navarro, McEwen)... (plus many more, including applications)

A note on likelihood-free approaches

The idea of likelihood-free approaches:

- Create a simulator $\theta \rightarrow \mathcal{D}^{\mathrm{sim}}(\theta)$ (stochastic)
- Sample from joint $(\theta, \mathcal{D}^{\mathrm{sim}})$ space, and estimate density w.r.t. some **divergence** between $\mathcal{D}^{\mathrm{sim}}$ and the actual data $\mathcal{D}$.
  That **density** implicitly defines a **data likelihood** $\mathcal{L}(\mathcal{D} \mid \theta, \mathcal{D}^{\mathrm{sim}})$.

This is (part of) the future of inference, but it comes with limitations:

- Low reusability, since derived likelihoods depend on models.
- Less statistical control (propagation of realisation stochasticity is simulated)
- Limited discoverability (better than summary statistics, but not that much)
- Assessing robustness is an open problem.

"Are summary statistics an unfortunately necessary intermediate step? Or a useful tool for Physics?"

Our approach:

Emulate the much simpler mapping **[parameters** $\longrightarrow$ **posterior]**:

$$\Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R}_+ \in L^1, C^{\geq 2}$$

GOAL: **drop-in** replacement for traditional MC

Probability density functions (our target!) are **positive**: let's model their $\log$

WISHLIST:

- No need for pretraining
- Small overhead / fast on CPUs (though GPU-capable)
- Modest memory requirements
- Robustness(?)

$\Rightarrow$*TOOL:* **Gaussian Processes** $\sim$ probabilistic interpolators

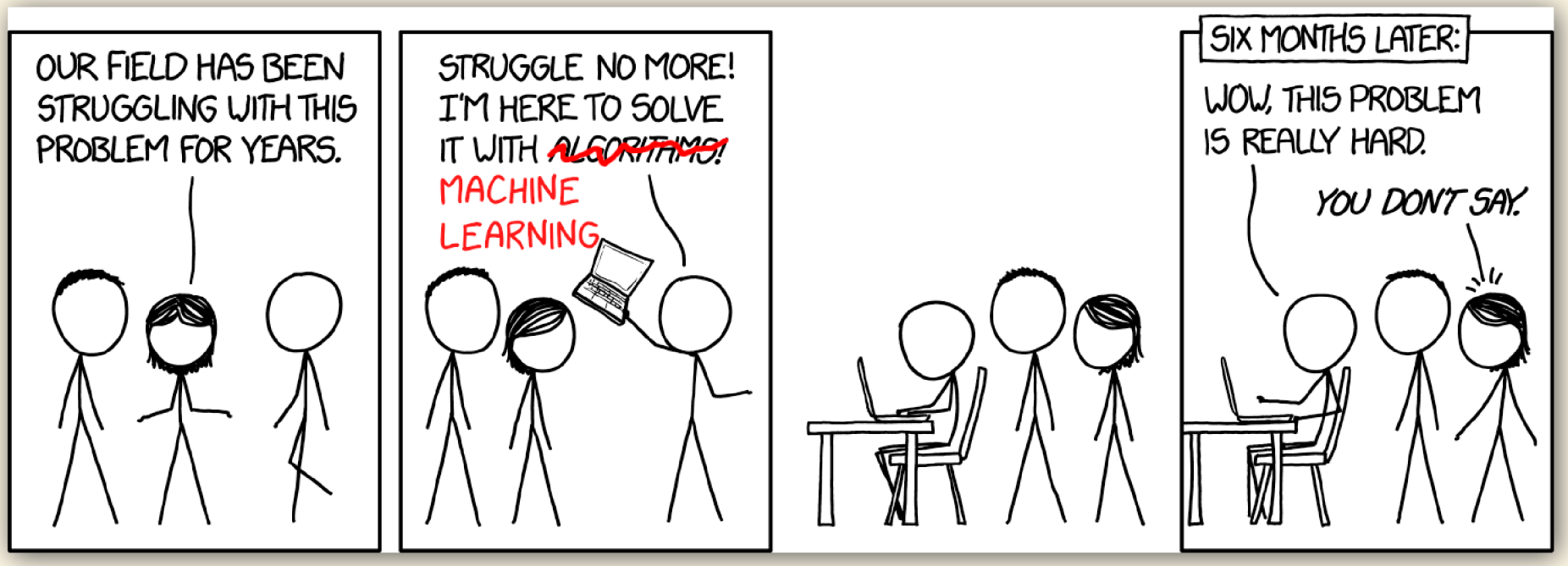
(Similar approaches: Pellejero-Ibáñez et al. '19, Acerbi '18...)

Why not NNs

Advantages of Gaussian Process *for this particular problem*:

- Size of the model / number of hyperparameters
  - Much less training needed
  - Interpretability (noise, correlation length...)
  - Robustness to overfitting / better extrapolation
  - Much faster / trained on-the-fly
- Easier to incorporate prior information
- Probabilistic: natural way to define *active sampling*
- Always differentiable

If you start adding restrictions to a NN (smoothness, predictivity so that it can be trained with small sets of samples...) it starts looking like a GP.



https://xkcd.com/1831/

The elements of GP parameter inference:

1. **Modelling**: choice of GP kernel, variable transformations...
2. **Acquisition** (**active sampling**): search for the next optimal locations.
3. **Evaluation of the true model** at the optimal locations.
4. **Re-fitting** of the GP model: adding new points and finding new optimal hyperparameters
5. **Convergence checks**
6. **Profit!** Inference with your shiny, analytic, differentiable GP model.

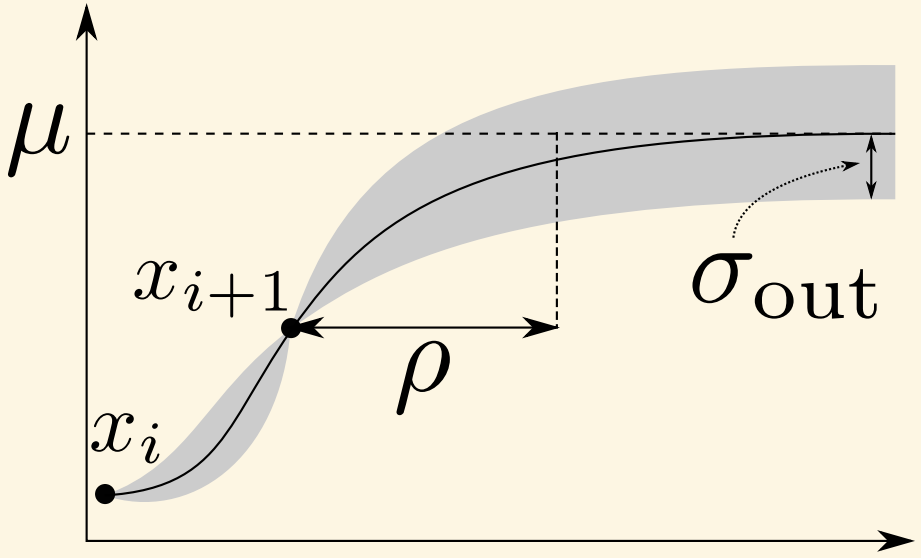
GP model: mean, kernel and hyper parameters

Rasmussen & Williams '06

$$f(x) \sim \mathcal{GP}\left(\mu(x),\, k(x, x')\right)$$

mean μ: baseline value to which uncorrelated values return.

kernel k: describes the correlation between points. Encodes prior information on f: e.g. smoothness, noise, periodicity...



$$k(x_1, x_2) = \sigma^2_{\text{out}} \cdot \mathcal{C}\left(|x_1 - x_2|; \rho\right) + \sigma_n^2 \delta_{ij}$$

σ^2_{out}: variance of *uncorrelated* points

ρ: correlation length(s)

σ_n^2: white noise variance (*)

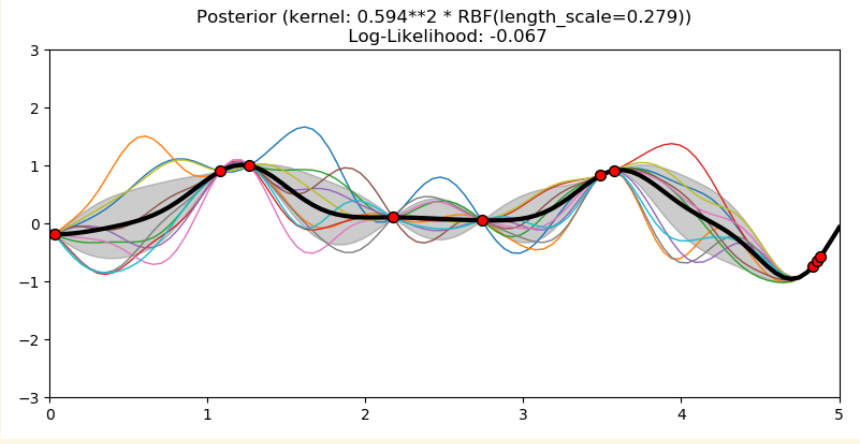
GP model: which kernel?

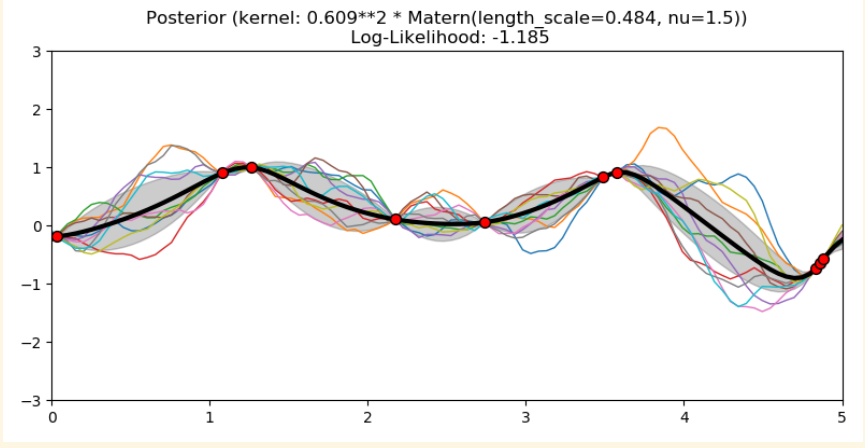
The **RBF** (or squared-exp) kernel imposes a long smoothness length.

The **Matérn kernel** ($\nu = 3/2$): allows for less smooth functions, while staying compressed along the mean.

$$\mathcal{C}_{\text{Gauss}} = \exp\left(-\frac{|x_1 - x_2|^2}{2\rho^2}\right)$$

$$\mathcal{C}_{\text{Matern},\,\nu=3/2} = \exp\left(-\frac{\sqrt{3}|x_1 - x_2|}{\rho}\right)\left(1 + \frac{\sqrt{3}|x_1 - x_2|}{\rho}\right)$$





Active sampling – where to evaluate next?

Key to reducing the number of evaluations: choosing well where to evaluate next.

Optimal locations are chosen such that they maximise an **acquisition function**, usually defined in terms of how **reduction in local uncertainty** would affect some **global target**.

E.g.

$$\text{target: pdf } p(\theta) \quad \Longrightarrow \quad \text{acq. function: } \Delta\mathrm{Var}(\mathcal{Z}|\log\mathcal{GP})$$

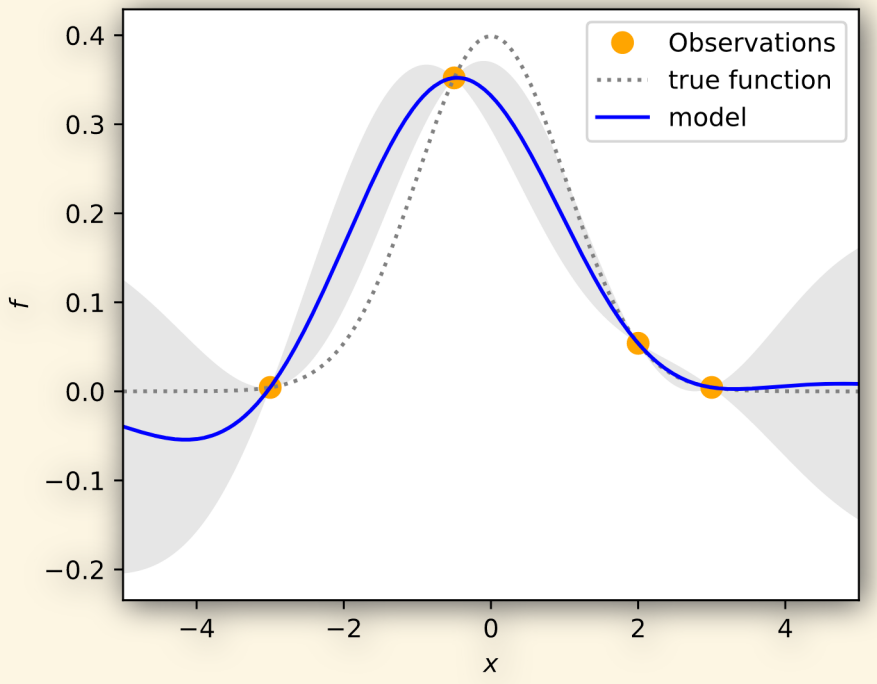
But that is **very expensive** to compute! (involves an integral)

Ideally, we will try to use a local measure instead of a global one.

Active sampling – a local function

Let's look at the GP uncertainty at each point *if we had modelled the posterior directly, instead of its* **log**:

$\mathcal{GP}$ model on p



The uncertainty could be used as an acquisition function, but ignores what regions are **most relevant**; just cares about far away one is from training points.

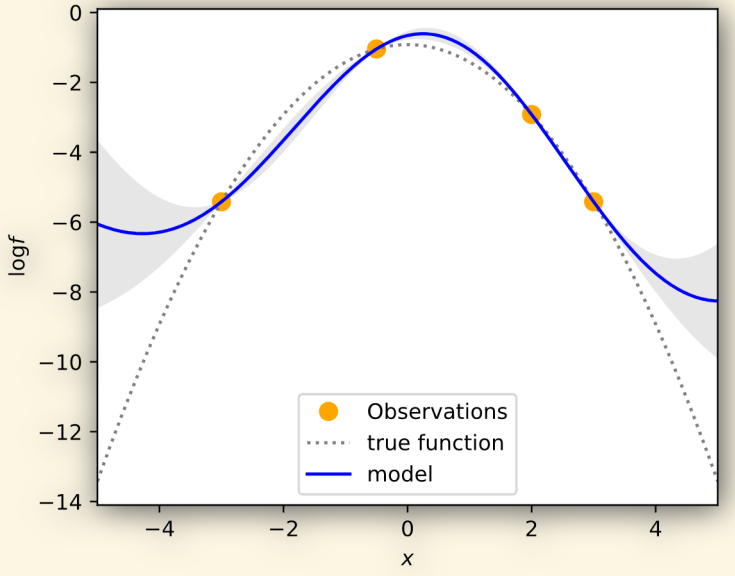
Active sampling – a local function (II)

But when modelling the **log**-posterior, the derived posterior uncertainty does care about the target function value:

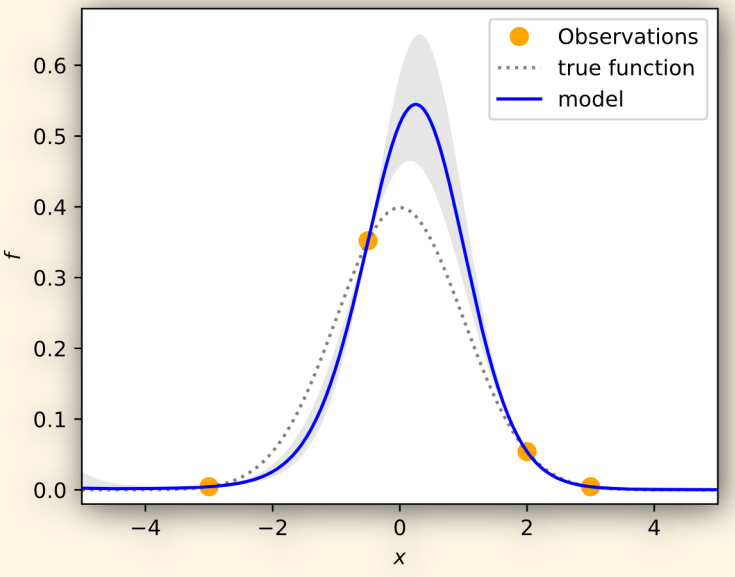
$$\log p \sim \mathcal{GP} \;\Rightarrow\; \sigma_p = \mathrm{e}^{\mu_{\log p}} \left(\mathrm{e}^{\sigma_{\log p}} - 1\right) \propto \mu_p$$

Gunter, Osborne et al '14; Chai & Garnett '18

$\mathcal{GP}$ model on $\log p$



$\mathrm{e}^{\log p}$



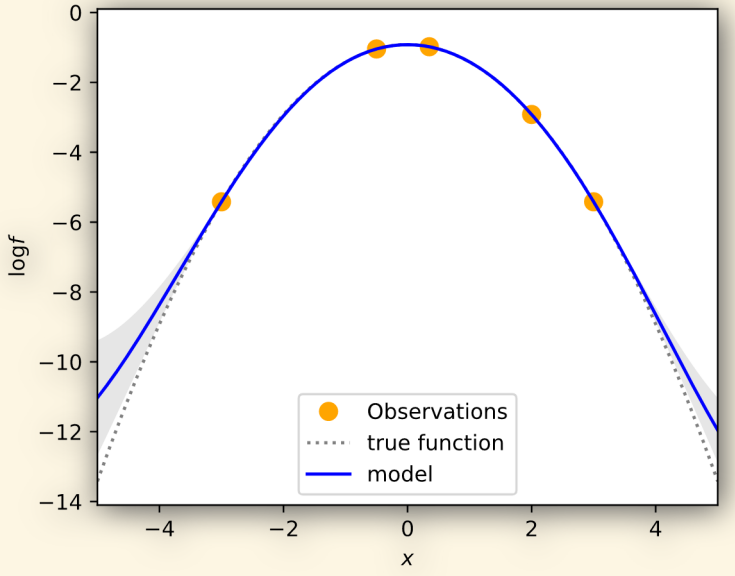
Active sampling – a local function (II)

But when modelling the **log**-posterior, the derived posterior uncertainty does care about the target function value:

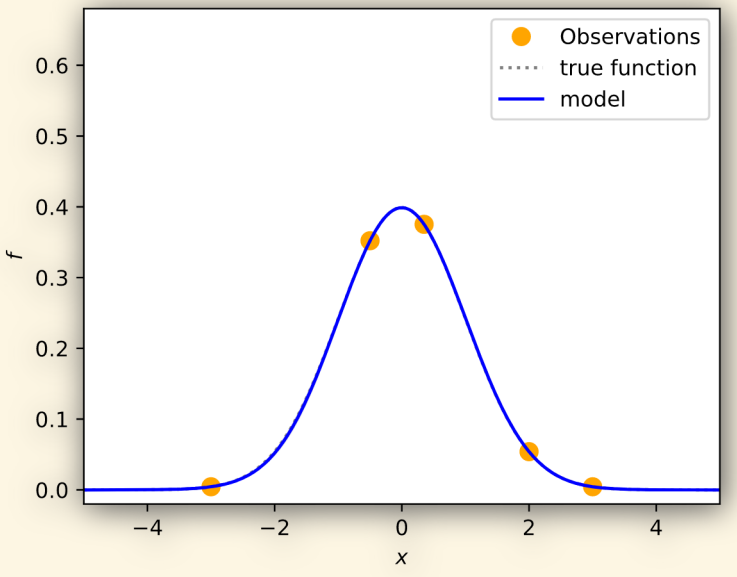
$$\log p \sim \mathcal{GP} \;\Rightarrow\; \sigma_p = \mathrm{e}^{\mu_{\log p}} \left(\mathrm{e}^{\sigma_{\log p}} - 1\right) \propto \mu_p$$

Gunter, Osborne et al '14; Chai & Garnett '18

$\mathcal{GP}$ model on $\log p$



$\mathrm{e}^{\log p}$



Active sampling – exploration vs exploitation

$$\alpha(x) = \mathrm{e}^{\mu_{\log p}(x) \cdot \zeta} \left(\mathrm{e}^{\sigma_{\log p}(x)} - 1 \right)$$

- ***Exploration***: encourages looking far away from training points, where uncertainty is higher.
- ***Exploitation***: focus on areas of high value of the target.
- ***Balance***: regulate or switch between the two.

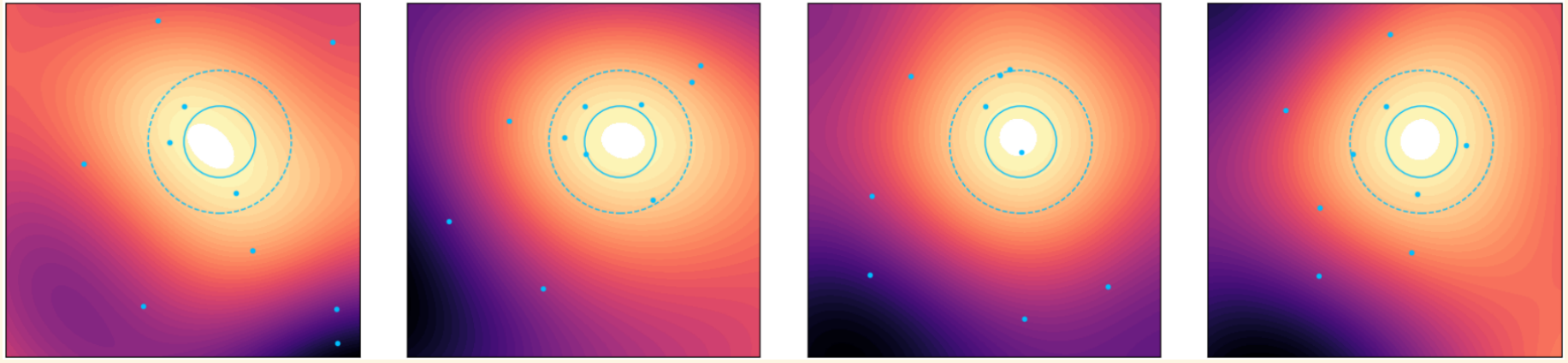
Unless checked, acquisition functions of this kind tend to **turn increasingly greedy/exploitative during learning**, because the surrogate model gets more confident (smaller σ).

May require fine-tunning for efficient convergence!

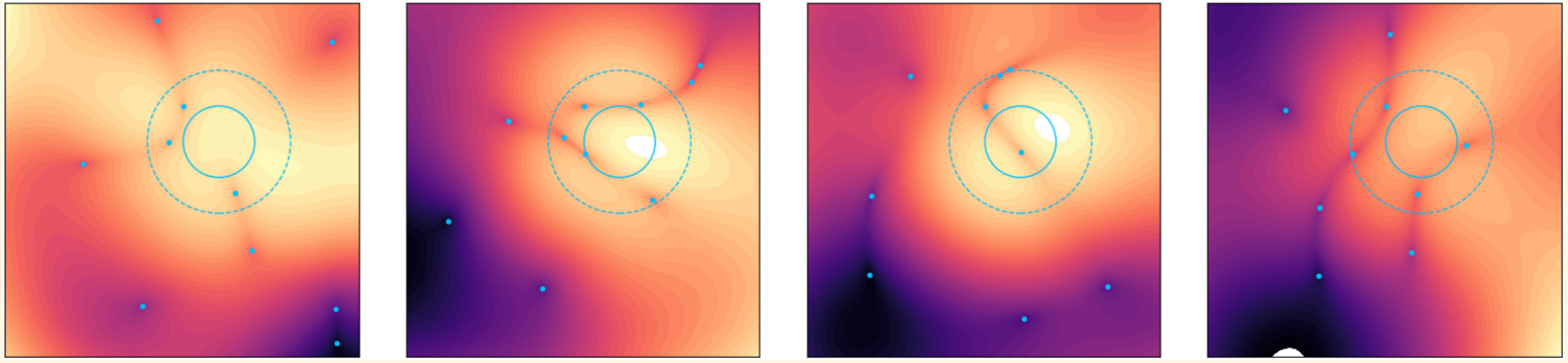
Active sampling – optimising

Global maxima of **highly multi-modal** function $\longrightarrow$ **Hard!**

Even if the model is nice and simple...



... its acquisition function has many local maxima



Active sampling – parallelisation

Key for the **slow-posterior** regime: **parallel** evaluations of the true posterior.

$\Rightarrow$ we need **simultaneous maxima of the acquisition function**.

$$\theta^1, \theta^2, \cdots \Leftarrow \underset{\theta^1, \theta^2, \ldots}{\arg\max}\, a(\theta)$$

Simultaneous optimisation is *very* expensive.

Problems with simply taking **multiple local maxima**:

a) Our maximiser would run into the same maximum more than once

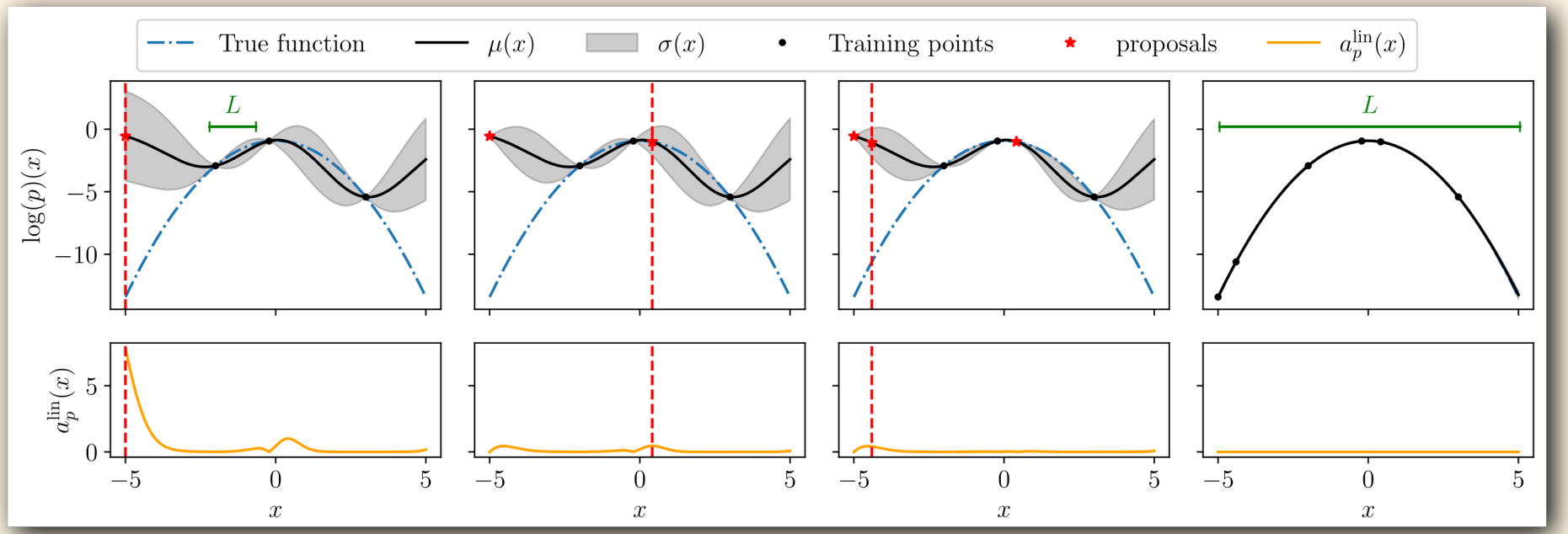
b) The information in one maximum may be redundant with another one

Active sampling – parallelisation (II)

A yypical (but not unique) choice is **"Kriging believer"**:

[find global maximum] $\longrightarrow$ [add GP-predicted mean there] $\longrightarrow$ [repeat]

Works well up to $n_{\text{parallel}} \approx d$.



NB: Adding a training point with the value of the mean GP does not modify the mean GP, but excludes points around the new one.

Model fitting

Once the **true posterior** is evaluated at the new optimal locations, we need to **re-fit** the GP model.

Re-computing the model (kernel matrix) has a cost $\mathcal{O}(N^3)$.

- If hyperparameters are kept constant, this only needs to be done **once**.
- If they are to be re-fitted, there are many $\mathcal{O}(N^3)$ evaluations.

This is way costlier than active sampling, which only requires GP model predictions that scale as $\mathcal{O}(N^2)$.

Usual strategy: re-fit hyperparameters only every m active-sampling steps.

After a while parameters don't change, in any case.

Convergence criterion

Specially because we target expensive functions, we need to know when we have approximated the target function well enough, so that we can stop.

A good convergence criterion would look at the **stability** over iterations of a **global measure**; e.g. the **Bayesian evidence** for the case in which we model probability density functions.

But global measures are (usually) expensive to compute, so we try **local measures** when possible, e.g. that the N last added training points are close enough to the real log-posterior:

$$\left|\mu(x^{(i)}) - \log p(x^{(i)})\right| < \epsilon$$

Local measures tend to require **fine tuning**.

Convergence criterion

Specially because we target expensive functions, we need to know when we have approximated the target function well enough, so that we can stop.

A good convergence criterion would look at the **stability** over iterations of a **global measure**; e.g. the **Bayesian evidence** for the case in which we model probability density functions.

But global measures are (usually) expensive to compute, so we try **local measures** when possible, e.g. that the N last added training points are close enough to the real log-posterior:

$$\left|\mu(x^{(i)}) - \log p(x^{(i)})\right| < \epsilon$$

Local measures tend to require **fine tuning**.

But for now we have all the elements that we needed!
We are ready to model and infer!


I am developing a ML sampler based on Gaussian Processes
It will be robust, right?
It will be robust, right?

Robustness – *it's the hard part!*

Usual kernels guarantee convergence to any function after sufficient learning.

But in a realistic implementation will convergence...

- ... happen fast/early enough?
  - GP costs increase rapidly as $\mathcal{O}(N^3)$
  - GP's break down due to bad-conditioning of kernel matrices
- ... be detected by the convergence criterion when it happens, and not before (false positive) or *too much* later (false negative).
- ... depend on fine-tuning of hyperparameters or initial conditions?
- ... happen reliably for non-gaussian and multimodal pdf's?

Our focus in this project:

Convergence robustness by incorporating prior information

Remember our emulation problem:

$$\Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R}_+ \in L^1, C^{\geq 1}$$

The target L^1 positive continuous function on a compact domain will either

(a) be approximately constant within the domain (easy!)

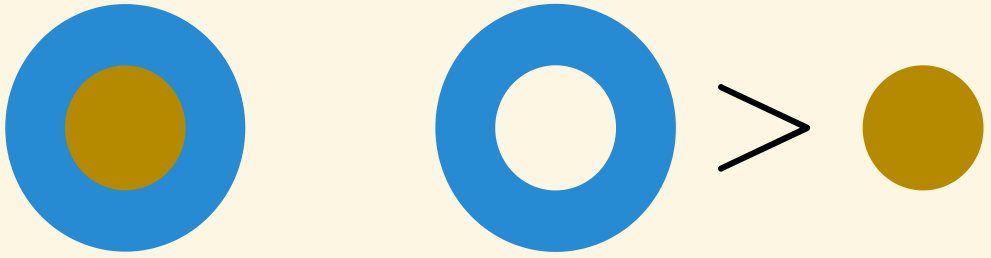
(b) have its mass concentrated around some *mode(s)*

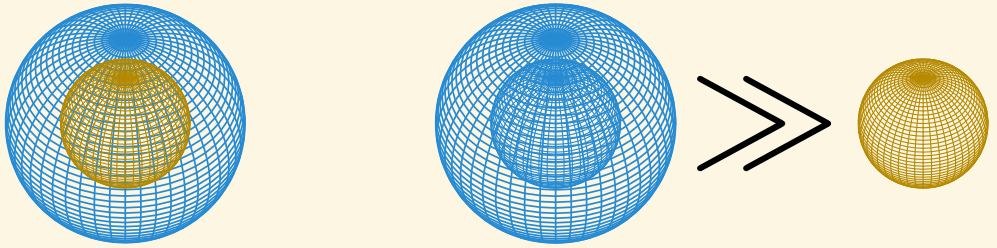
(or a combination of both in different dimensions)

The target are **probability distributions**, so we do not need to map the whole domain correctly, just the part of the domain **with the most *mass* below it**.

The curse(?) of dimensionality

$d = 1$ 

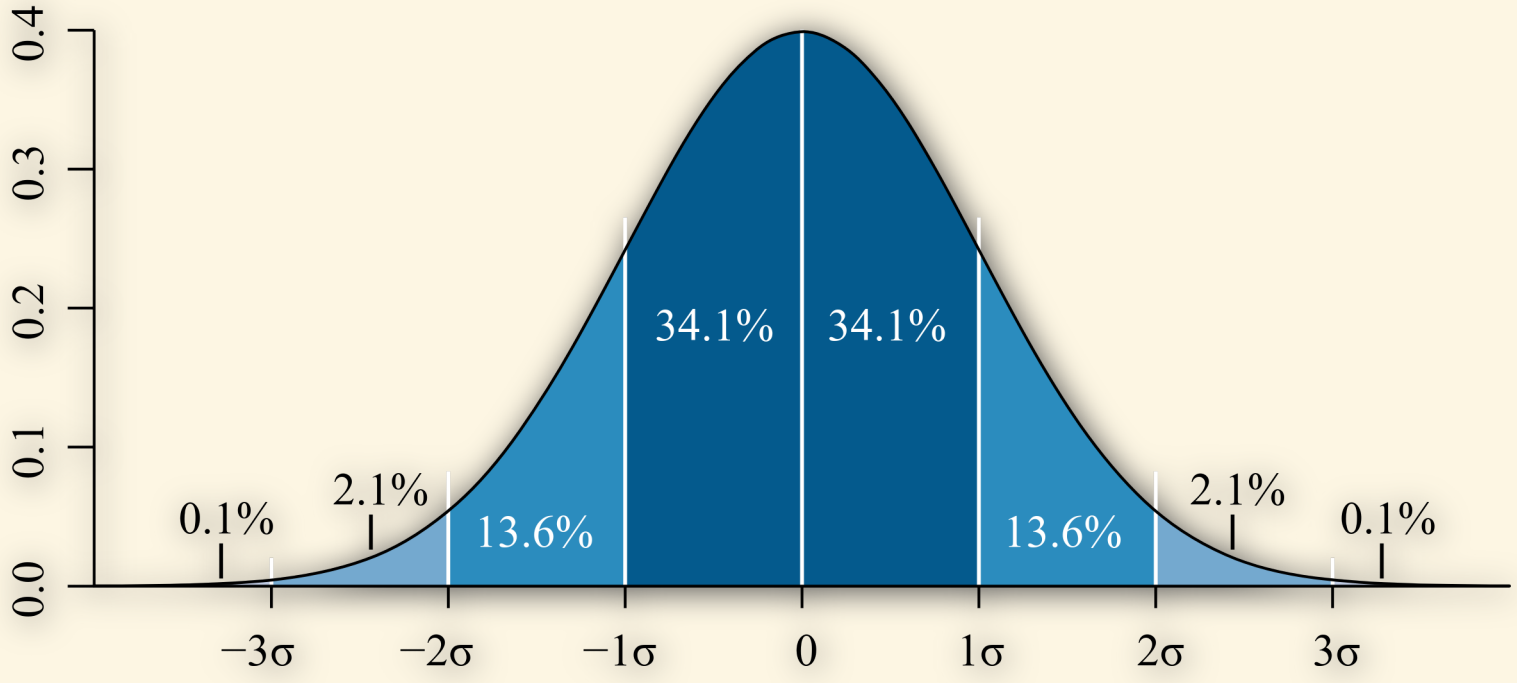
$d = 2$ 

$d = 3$ 

$\Rightarrow$ the cost of $\approx$grid-based methods diverges exponentially!

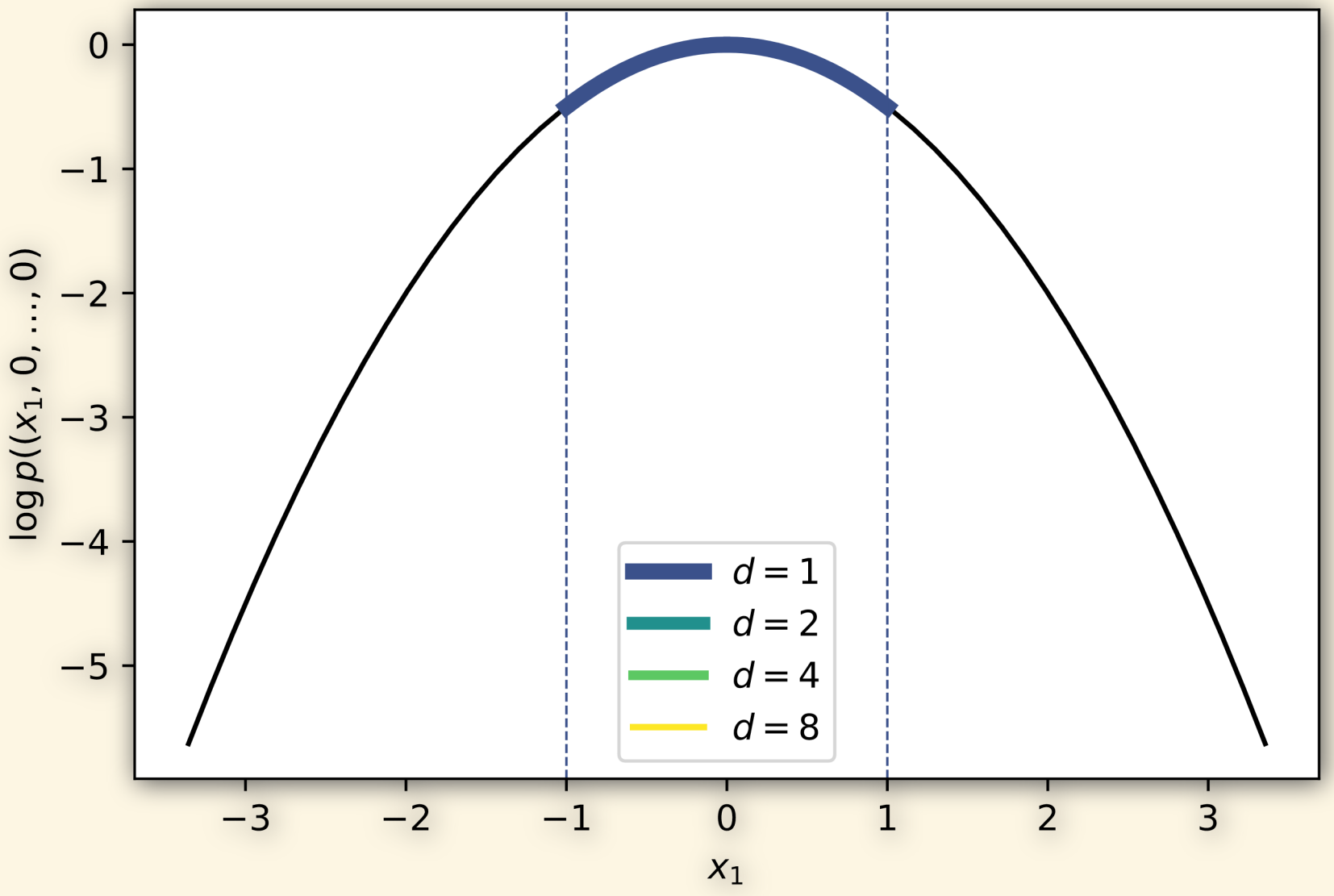
The curse(?) of dimensionality in Bayes. inference

Tails contain more and more of the **hypervolume/probability**!
It is important to map them correctly as dimensionality grows.

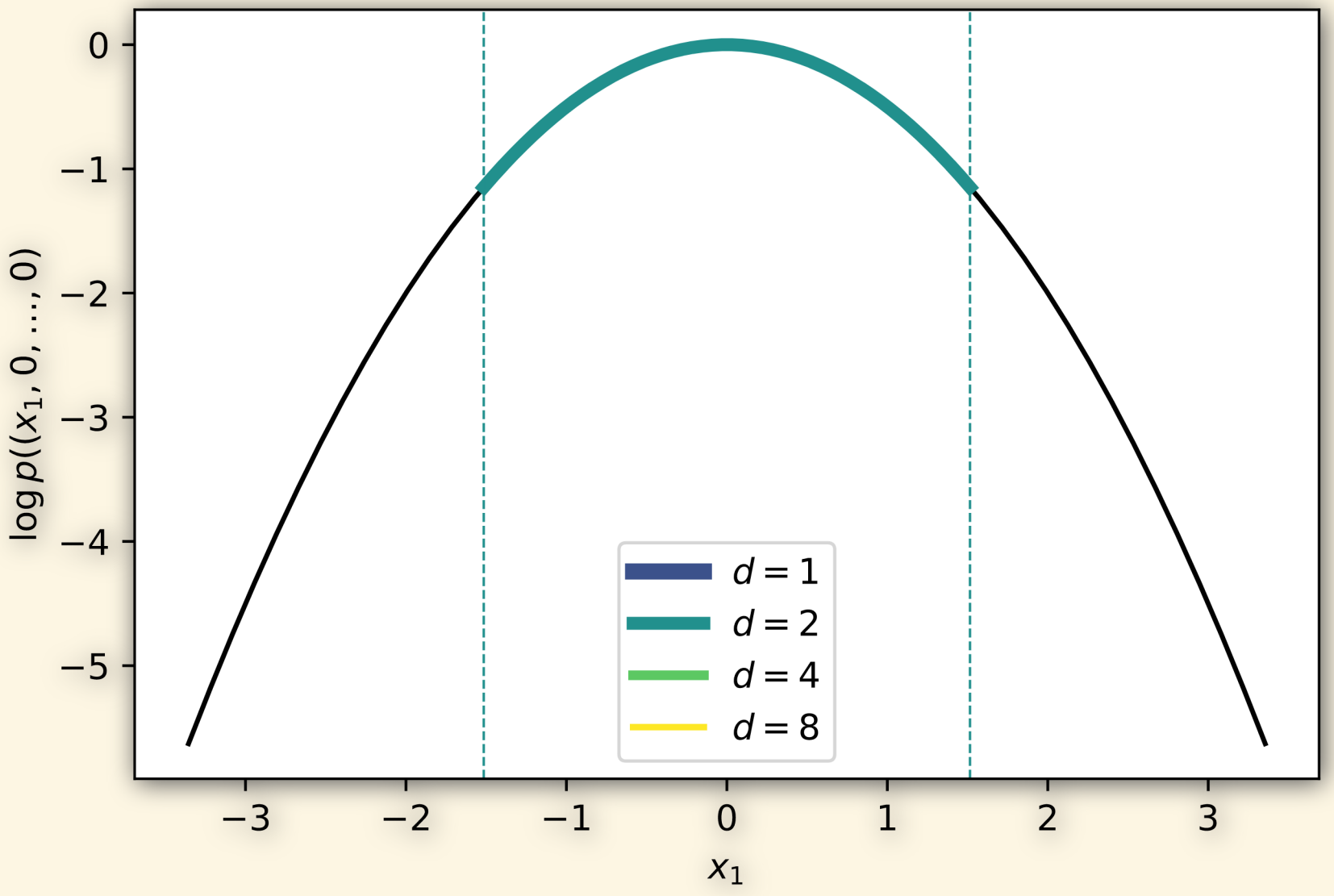


M.W. Toews, CC-BY 2.5 (Wikipedia)

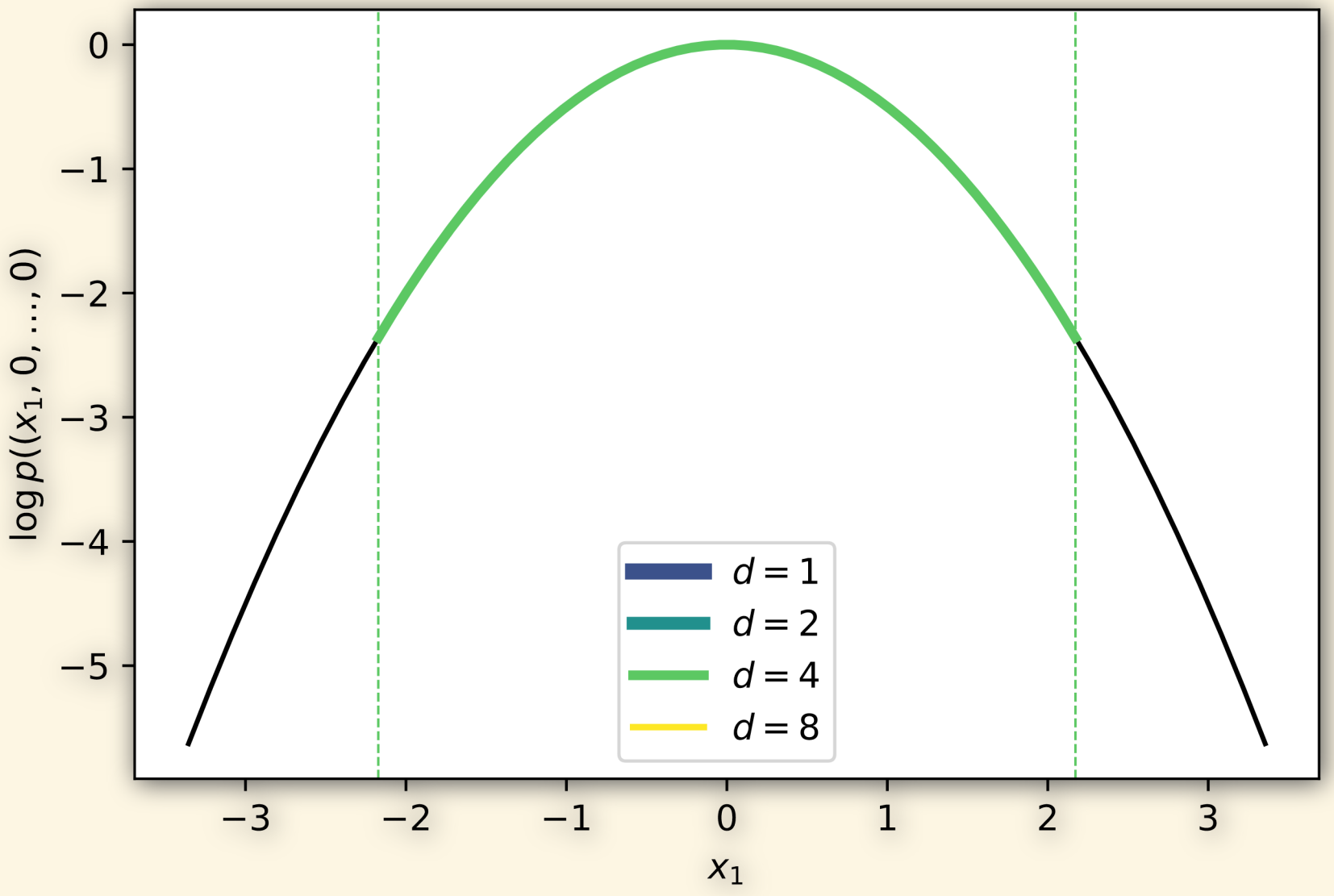
The radius of a 68% (hyper)contour



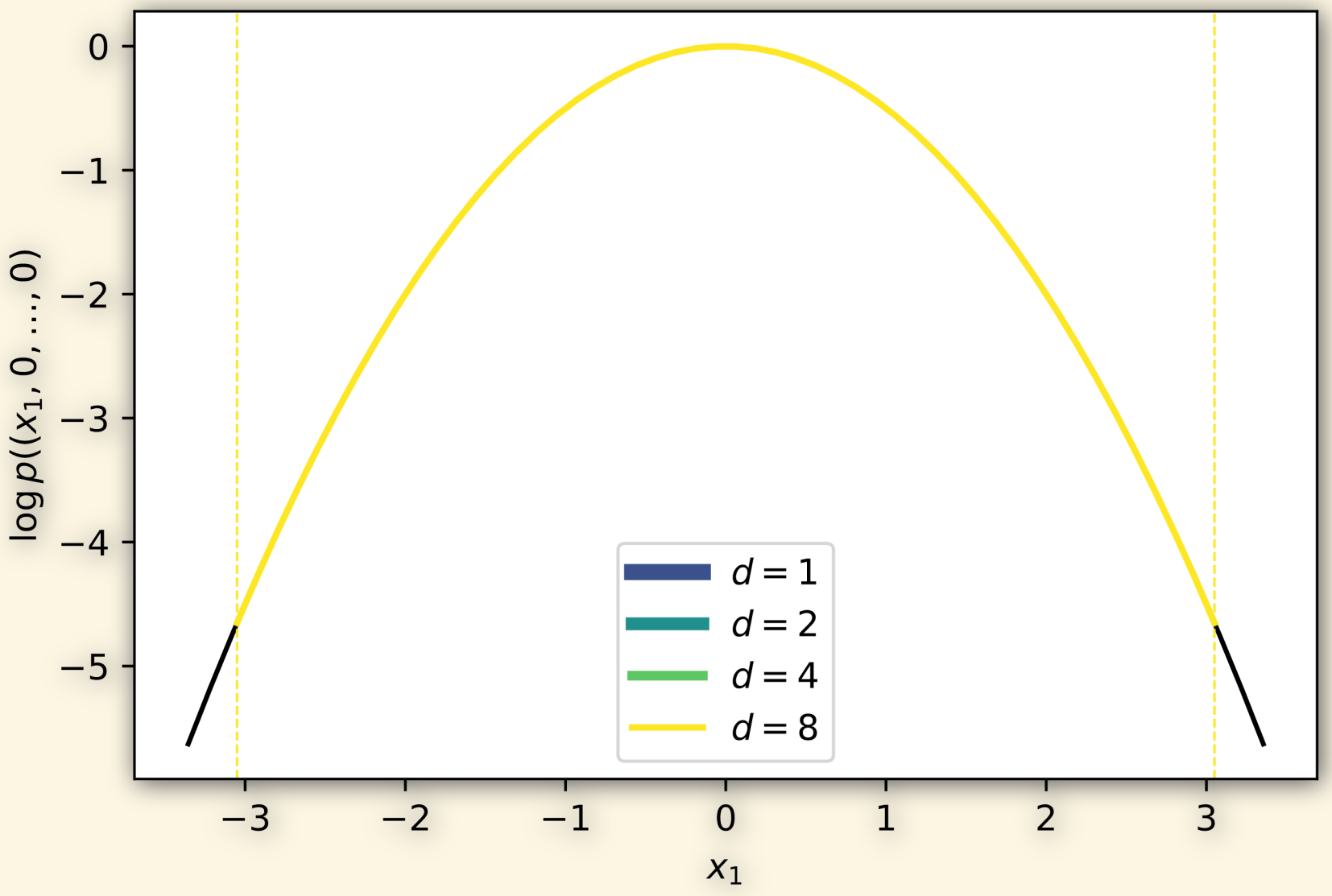
The radius of a 68% (hyper)contour



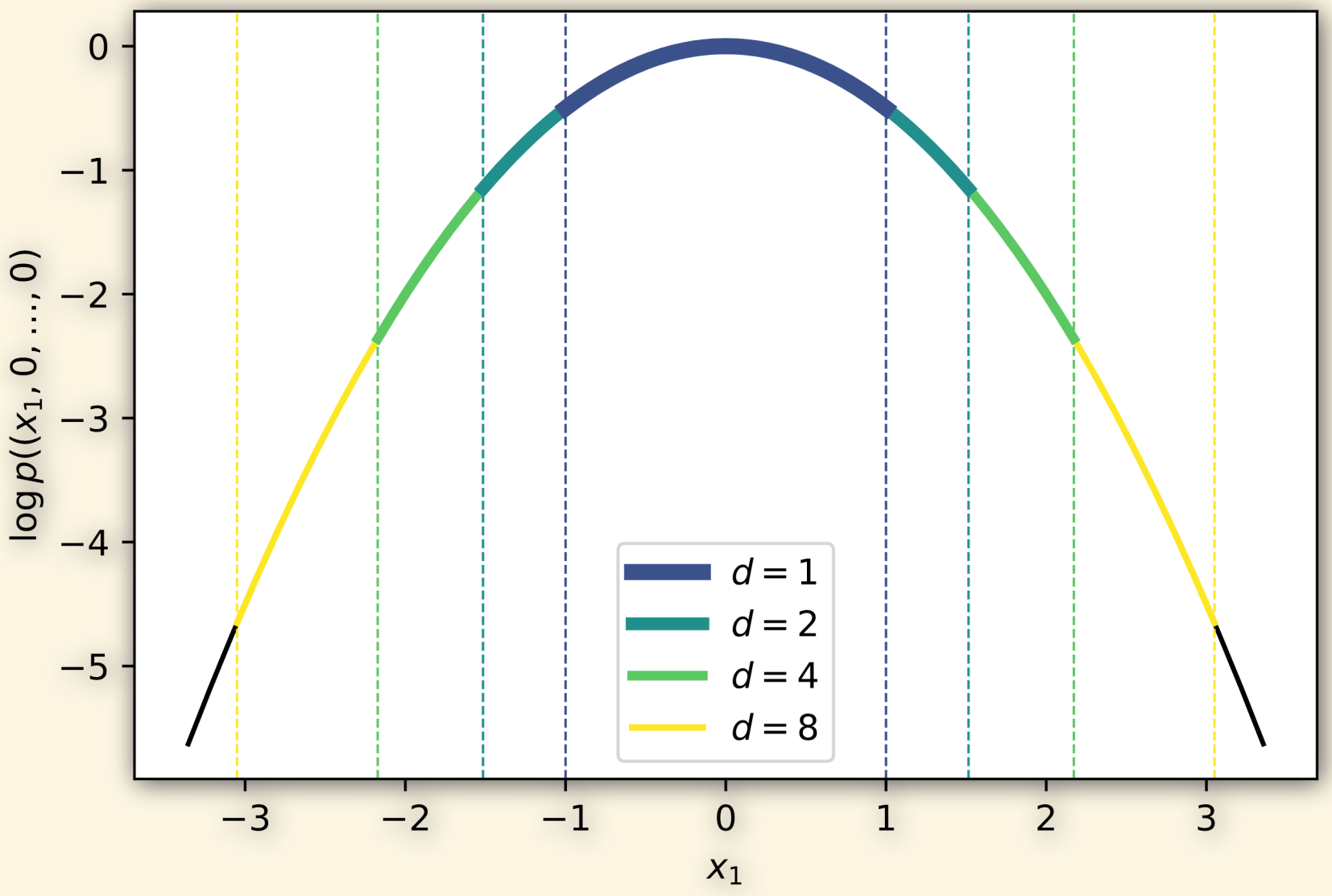
The radius of a 68% (hyper)contour



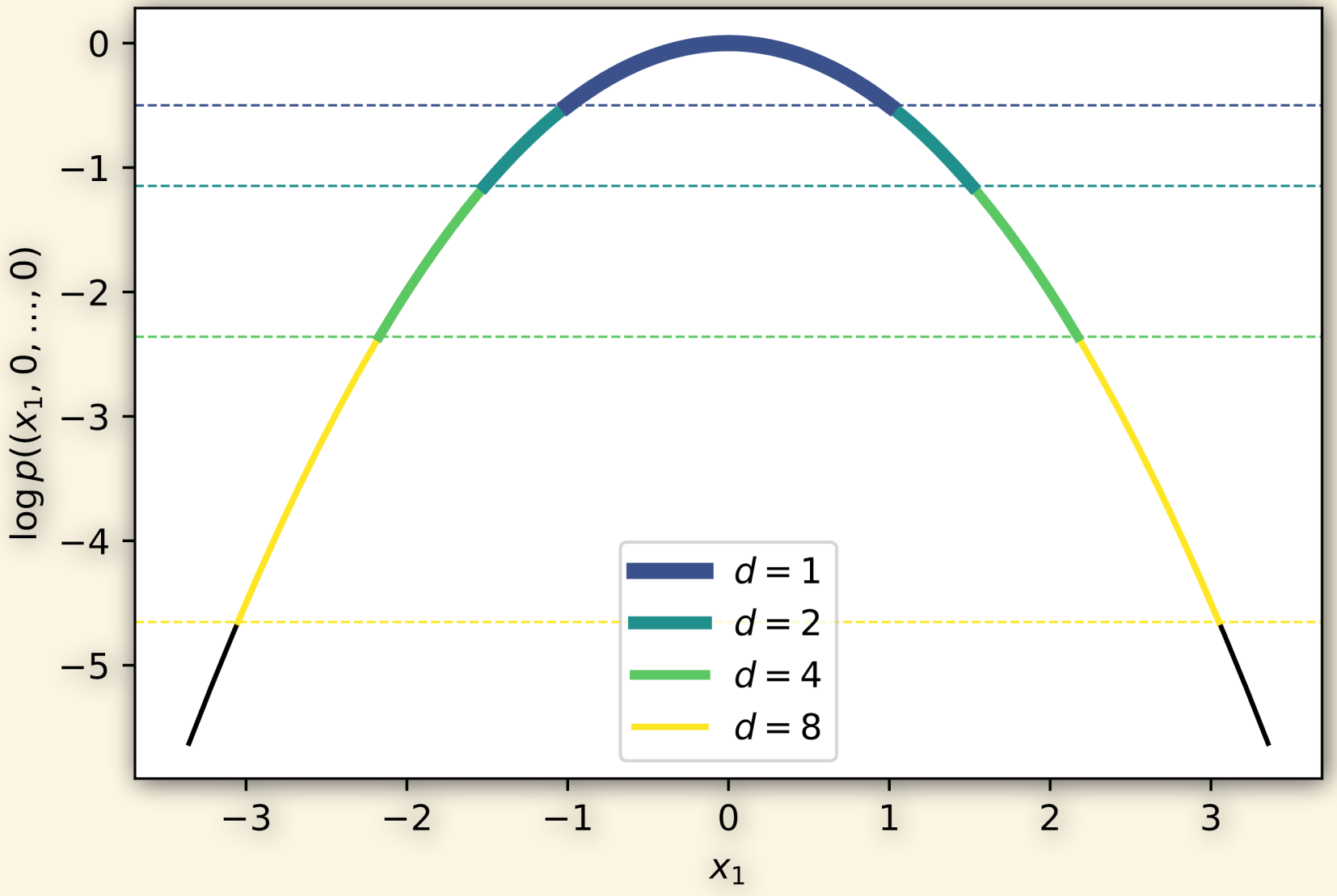
The radius of a 68% (hyper)contour



The radius of a 68% (hyper)contour



The radius of a 68% (hyper)contour $\Rightarrow$ dynamic range!



Acquisition function, revisited

We discussed the acquisition function

$$\alpha(x) = \mathrm{e}^{\mu_{\log p}(x) \cdot \zeta} \left(\mathrm{e}^{\sigma_{\log p}(x)} - 1 \right)$$

We use a linearised version of it: (the factor 2 is a convention)

$$\alpha^{\mathrm{lin}}(x) = \mathrm{e}^{\mu_{\log p}(x) \cdot 2\zeta} \sigma_{\log p}(x)$$

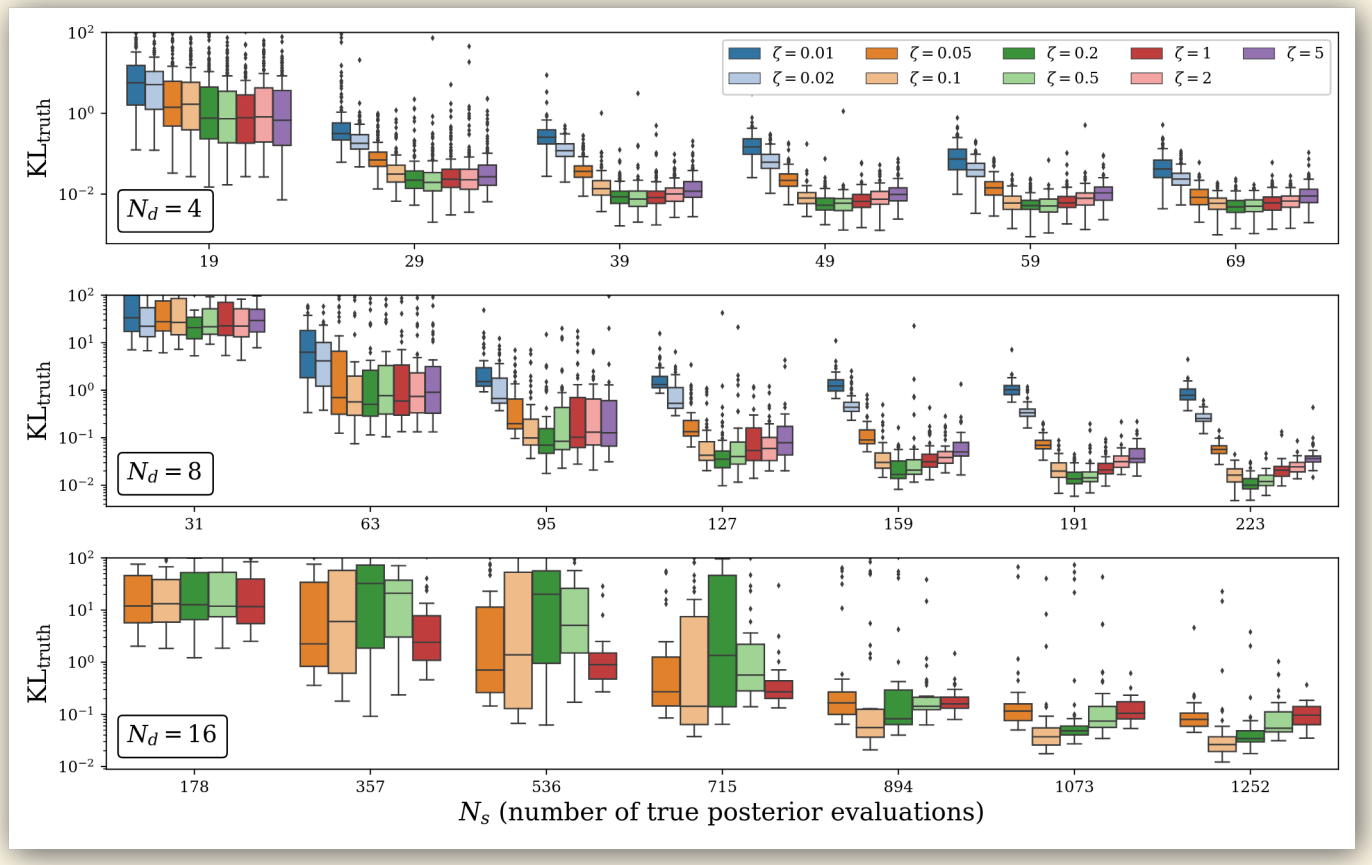
Since the **dynamic range of $\log p$ grows with dimensionality**, and the *exploitation* term e^{μ} depends on absolute values of the $\log p$, this **acquisition function(s) becomes greedier and greedier with dimensionality**.

Solution: make $\zeta(d)$ a decreasing function of d.

No obvious way to model that behaviour analytically: find an empirical fit that maximises convergence.

Acquisition function – empirical fit of ζ

$$\zeta \approx d^{-0.85}$$



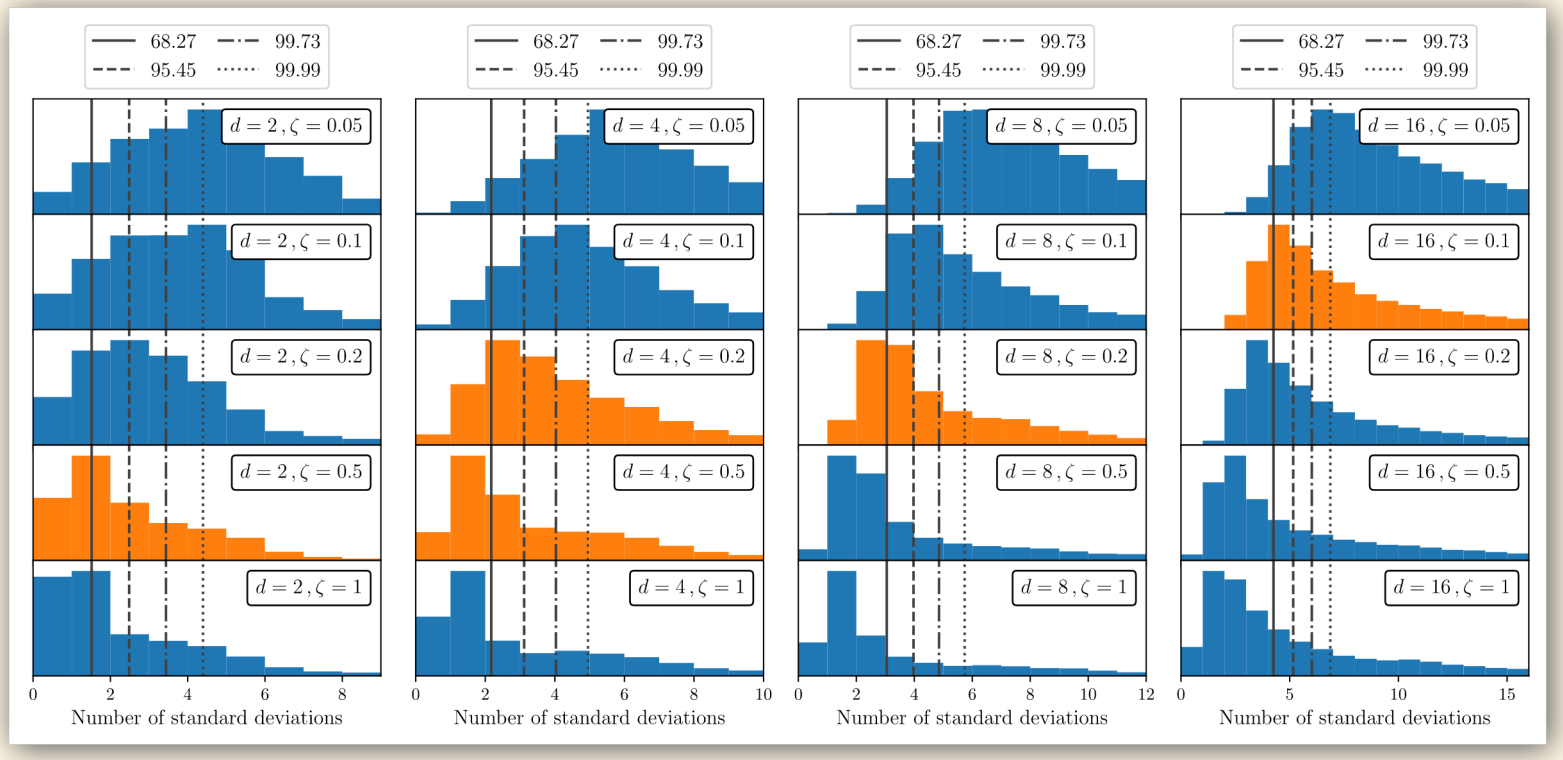
Acquisition function – empirical fit of ζ (II)

Do optimal ζ's follow consistent hypercontours across dimensionalities?

Acquisition function – empirical fit of ζ (II)

Do optimal ζ's follow consistent hypercontours across dimensionalities?

Yes, they do! In particular they focus in the 68% one.



NB: Here we have not used the modelling of the dimensionality dependence presented before. It just chooses to follow it to get better convergence!

Analytic calculation of dynamic range in a Gaussian

The log-probabilities themselves follow a (1D) χ^2 with d d.o.f.:

$$(x_1, \ldots, x_d) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \Longrightarrow \quad -2 \log p(\boldsymbol{x}) \sim \chi^2_d$$

By definition, for the *cumulative* function of a 1D probability distribution on x, with ϵ being the probability mass under $x > x_0$:

$$1 - \epsilon = F(x_0)$$

So the dynamic range of the target log-p for enclosed probability mass $1 - \epsilon$:

$$\Delta_{1-\epsilon} \log p = -\frac{1}{2} F_d^{-1}(1 - \epsilon)$$

(the maximum that defines the Δ is the maximum in the training set)

Convergence criterion, revisited

We mentioned the possibility of using cheap local criteria, such as

$$\left|\mu(x^{(i)}) - \log p(x^{(i)})\right| < \epsilon$$

Recalling the dependence of the **dynamic range** of $\log p$ with dimensionality, it should be obvious that this criterion becomes **stricter** as dimensionality grows!

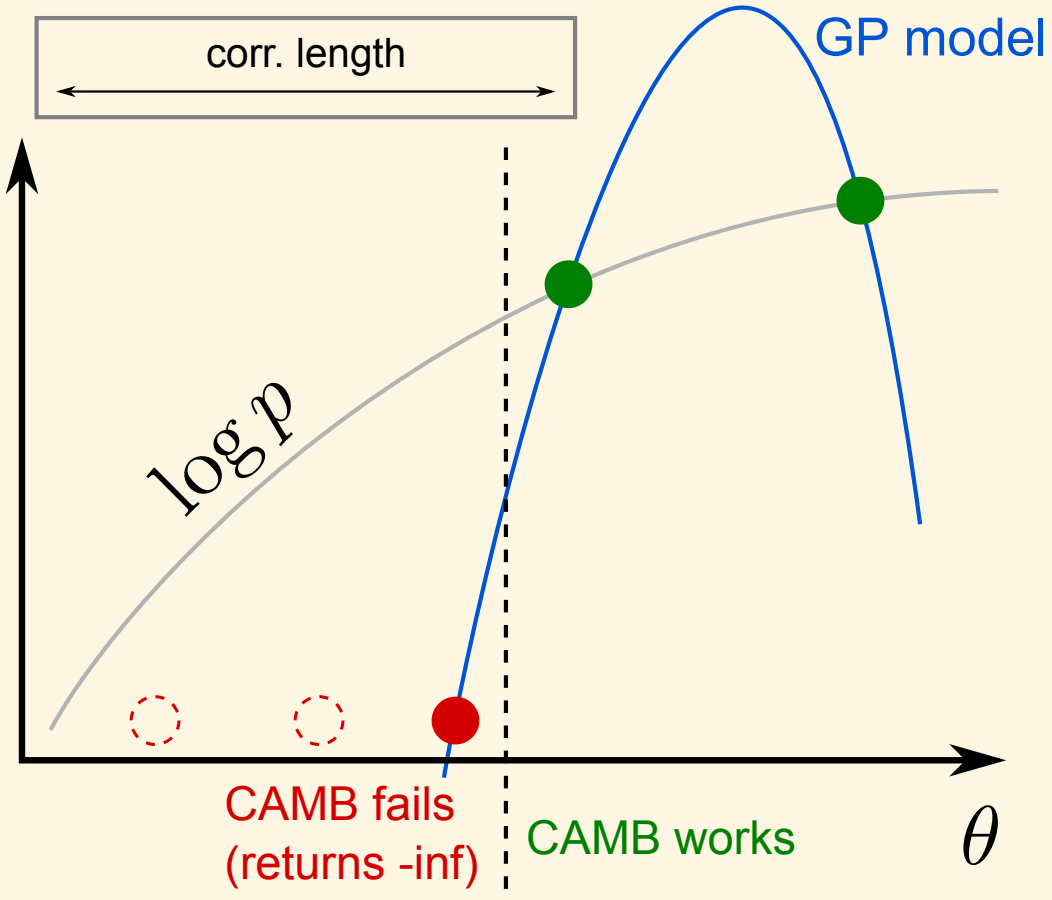
We should relax it when the dynamic range of interest is larger (larger d).

We set ϵ to a small fraction of the $\Delta \log p(68\%)$ in each dimensionality!

Reliable convergence without further fine tuning, with a very cheap criterion!

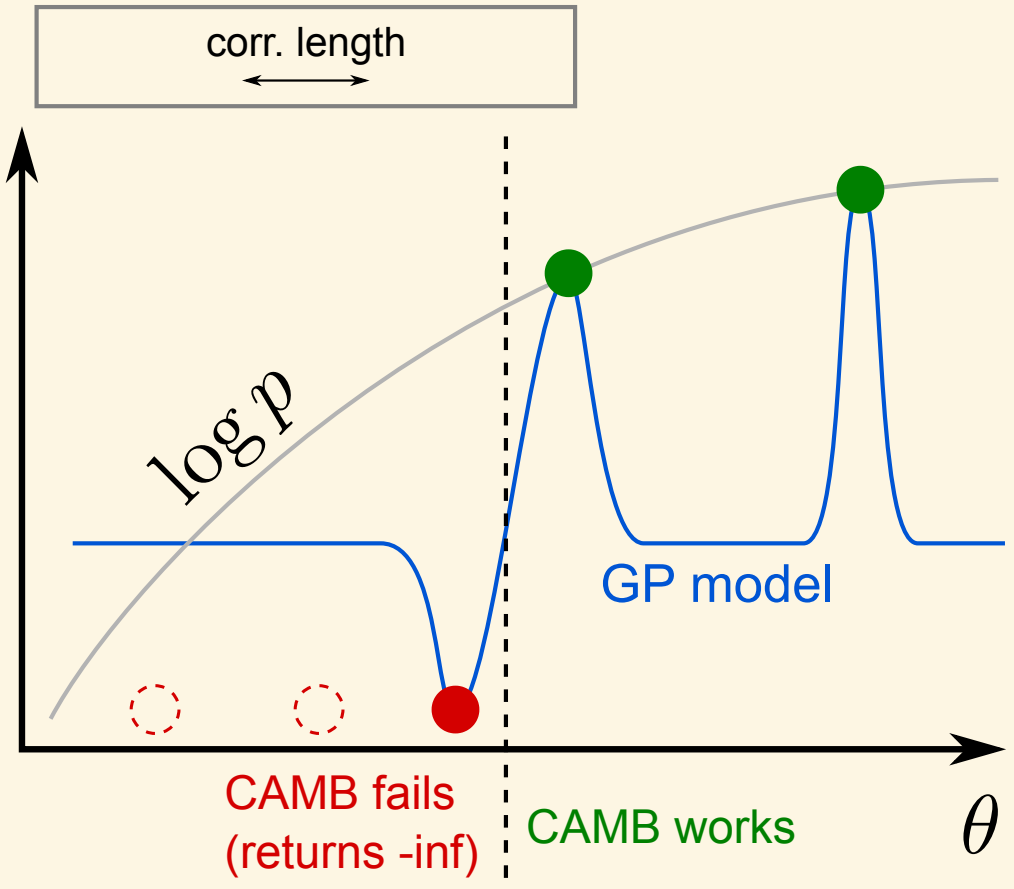
(In practice we have another term in the RHS, $\left|\mu(x^{(i)}) - y_{\max}\right| \cdot \epsilon_{\text{rel}}$, but it has no dim. scaling)

Infinities and extreme values



Likely to either **overshoot** (long corr. length)

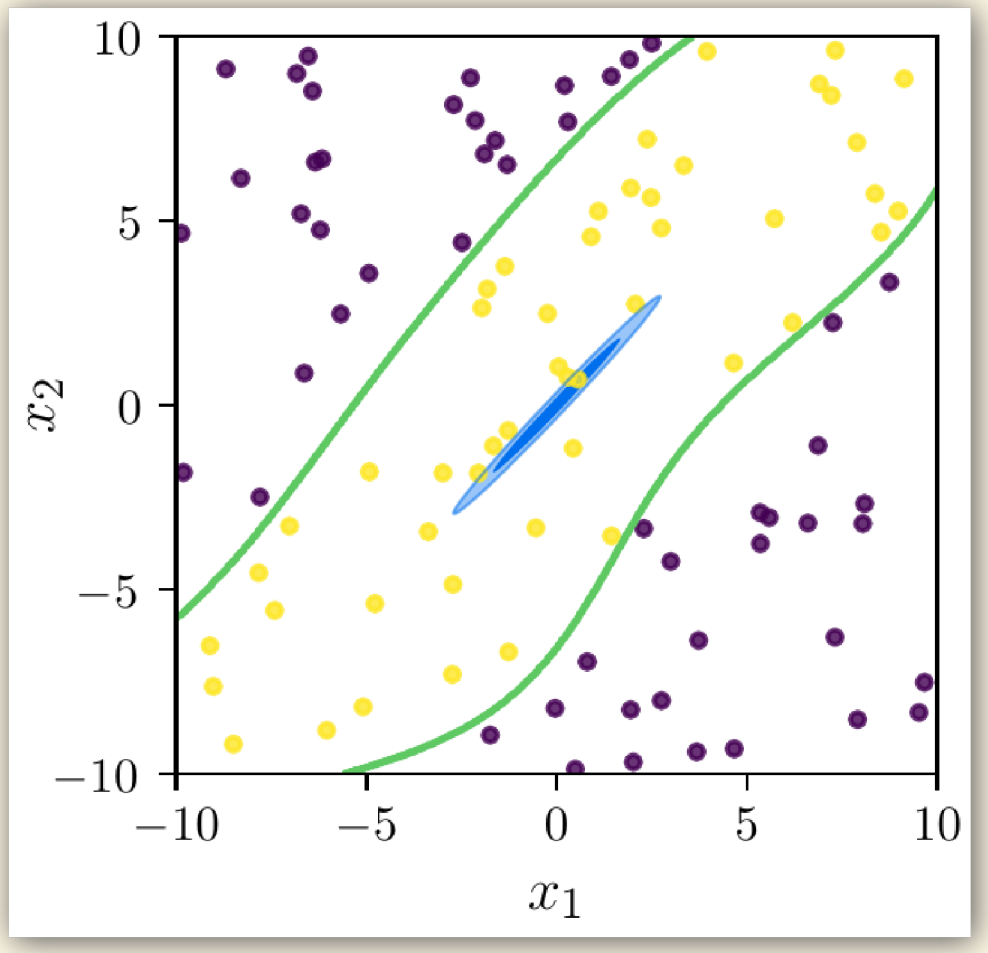
Infinities and extreme values



Likely to either overshoot (long corr. length) or **overfit** (short corr. length).

Regularising the bad points doesn't help too much (and is finicky)

SVM, a cheap classifier to learn which regions to ignore



Our solution to this problem is to simultaneously exclude these *infinities* from the GP, and to use them to divide the parameter space into a *finite* and an *infinite* region using a support vector machine (SVM) classifier [61, 62].[8] A SVM defines a hyperplane which maximizes the separation between samples with locations $\boldsymbol{x}_i$ belonging to one of two classes $y \in \{-1, 1\}$. By defining the distance between points through a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ the separating hyperplane is drawn in a higher-dimensional space which is connected to the sample space by a non-linear transformation. This effectively transforms the separating hyperplane into more complex hypersurfaces which are better suited to the classification problem at hand.

The categorical predictions $\hat{y}(\boldsymbol{x})$ of the SVM are then given by

$$\hat{y}(\boldsymbol{x}) = \operatorname{sgn}\left(b + \sum_{i=1}^{N_s} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})\right) \tag{3.3}$$

where the hyperparameters b and α_i are optimized in the training procedure.

SVM, a cheap classifier to learn which regions to ignore (II)

Calibration needs knowledge about the expected dynamical range of the target function, i.e. *how small is too small and w.r.t. what?*

We go back again to our Gaussian example, where we can calculate the dynamical range, and simply choose a threshold in terms of credibility ϵ.

We have found that the ϵ equivalent to 20 1D standard deviations works well, and the correct scaling is guaranteed.

(The SVM will not only ignore $-\infty$ points, but also very-low-value ones. That's OK, and will keep the model cheaper and more numerically robust.)

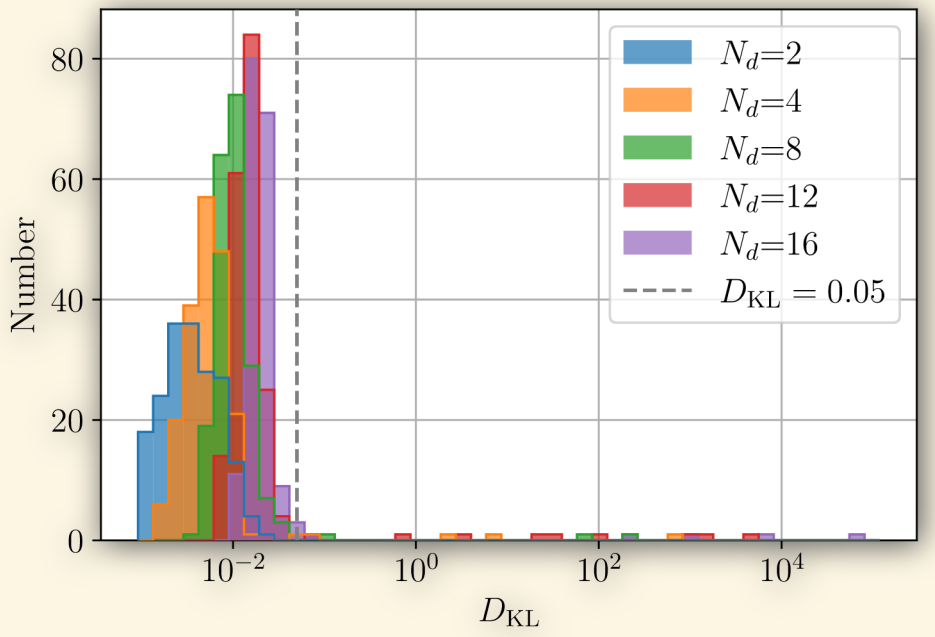
Summary of our advances in modelling

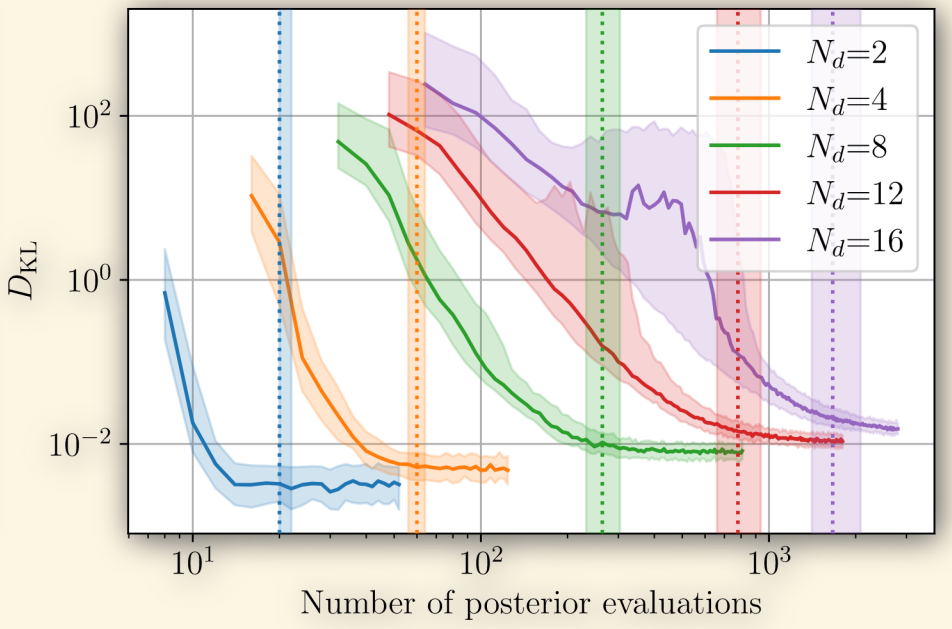
Using the dimensionality scaling of Gaussians as an ansatz, we have:

- added dimensionality scaling to our **acquisition function** to guarantee good exploration-vs-exploitation balance
- added dimensionality scaling to our **cheap convergence criterion** to make it as robust as a global one
- proposed a **SVM classifier** for outliers with a dimensionally-scaling threshold

Altogether, this produces more reliable convergence with fewer evaluations of the true model ($\sim \mathcal{O}(10)$), and removes the need for fine tuning, without making the model less flexible!

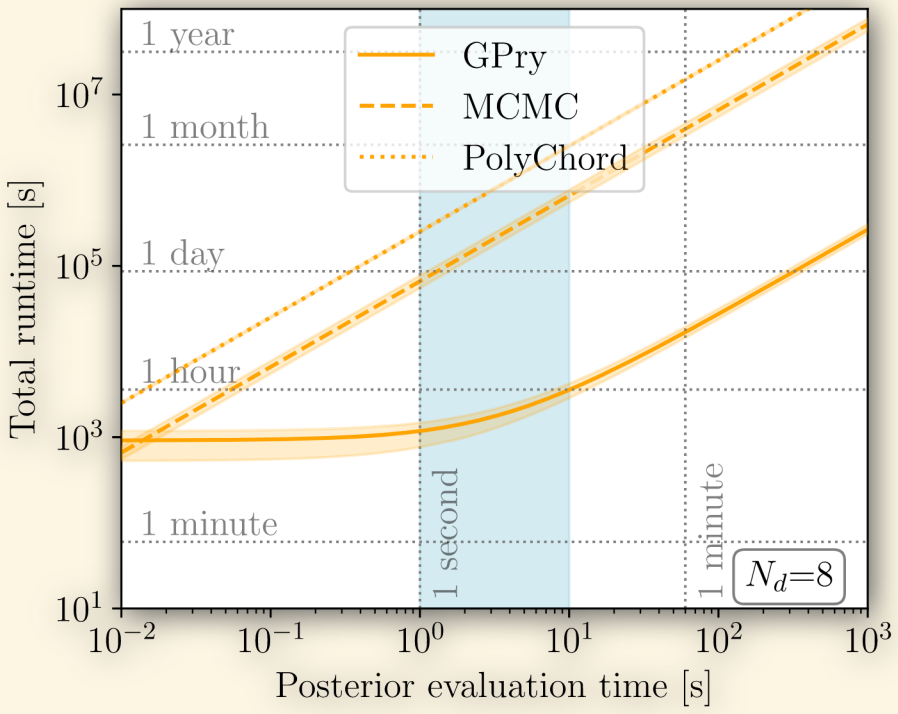
Examples/convergence experiments: Gaussians

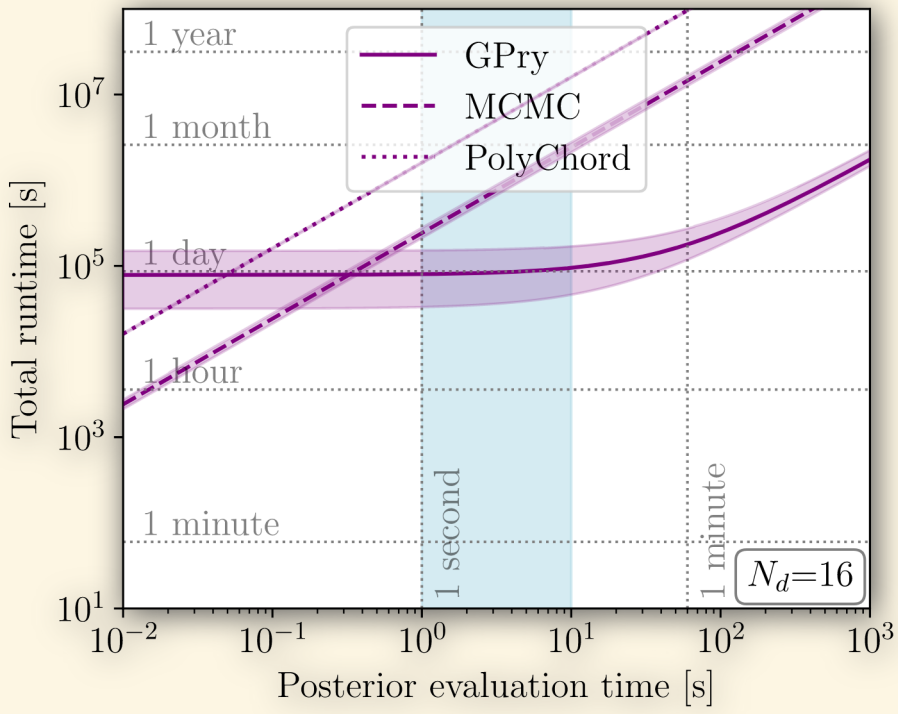




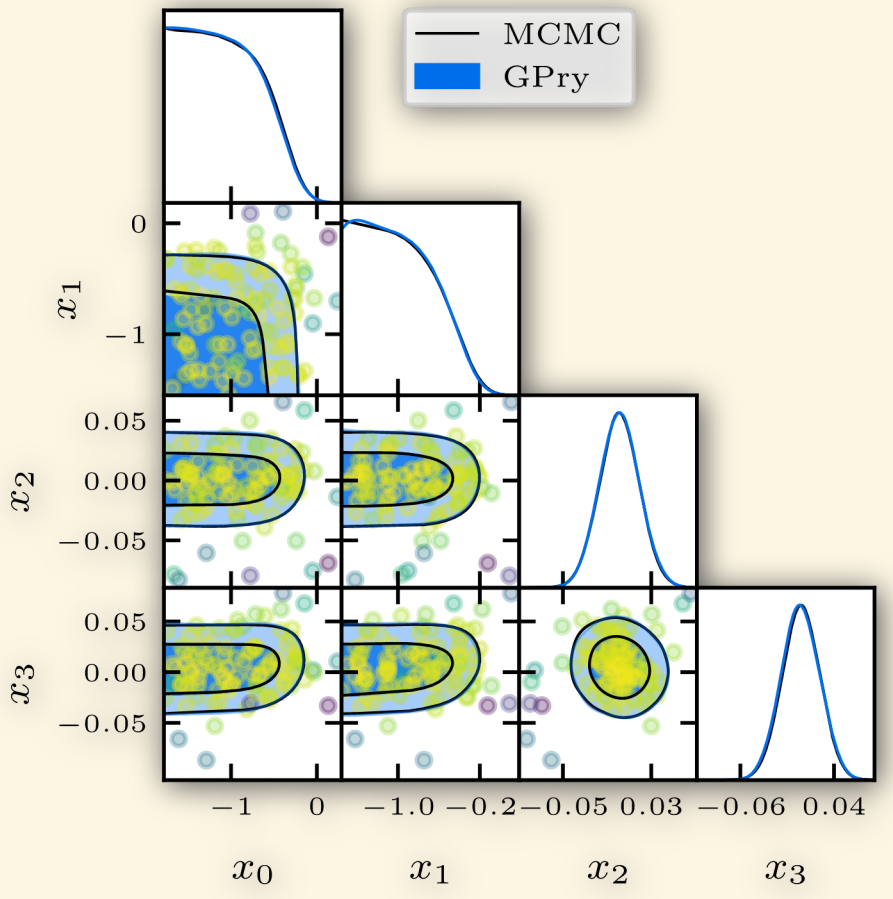
Examples/convergence experiments: Gaussians

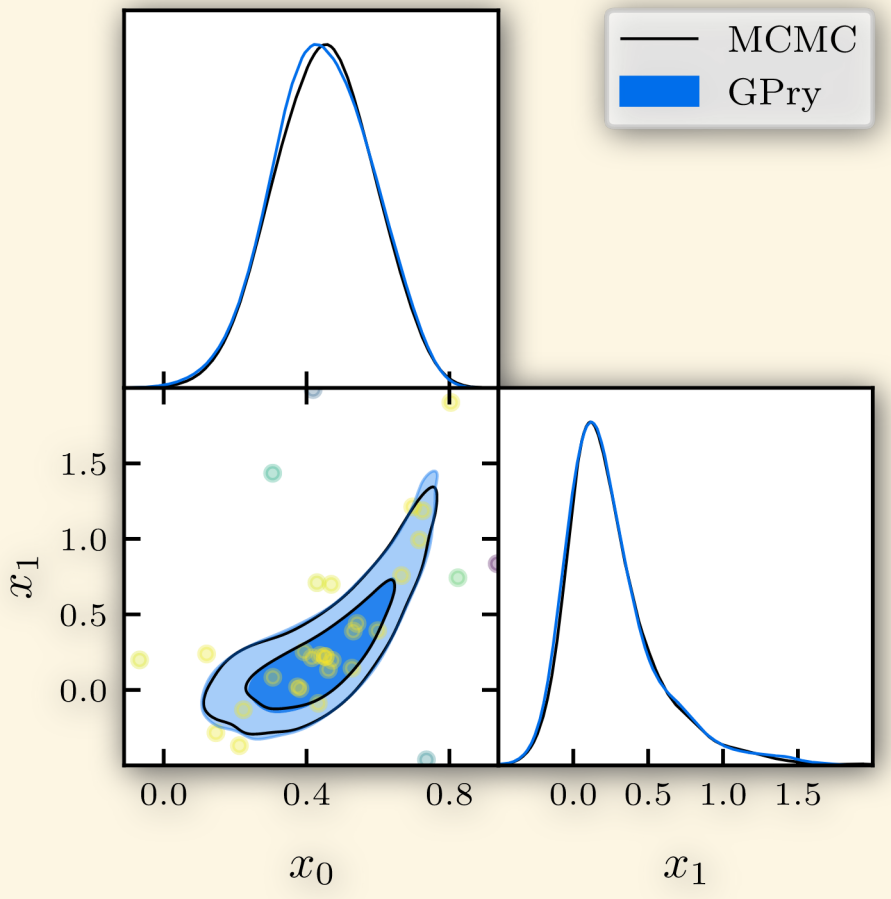
Cost comparison:



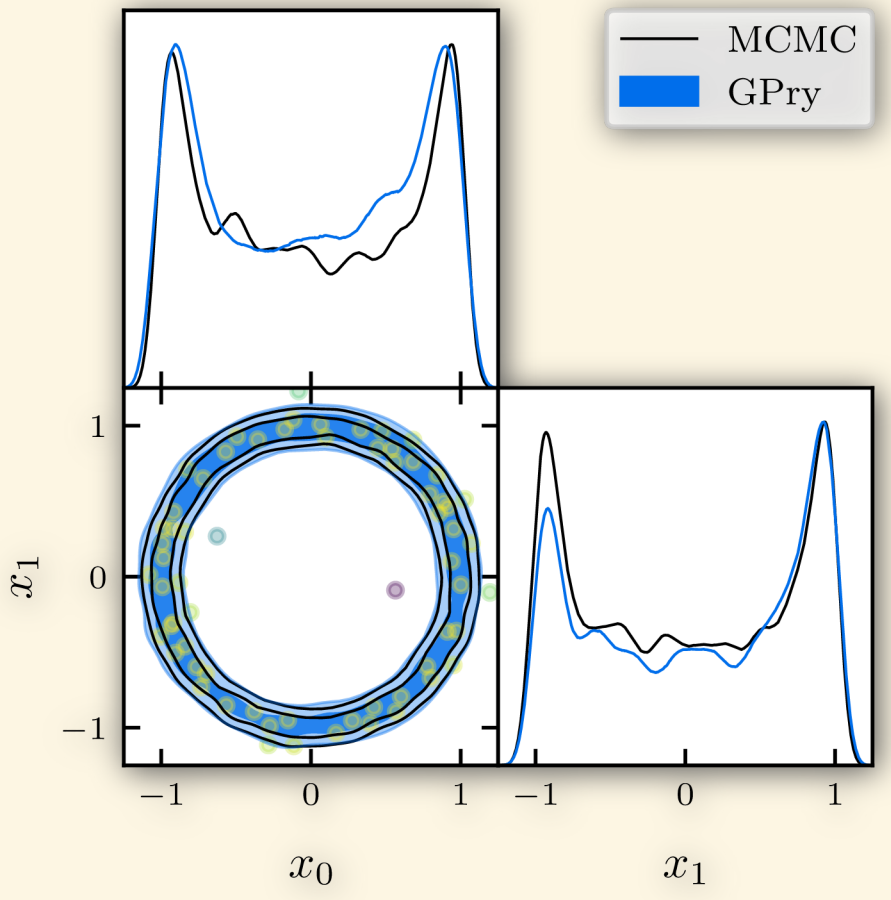


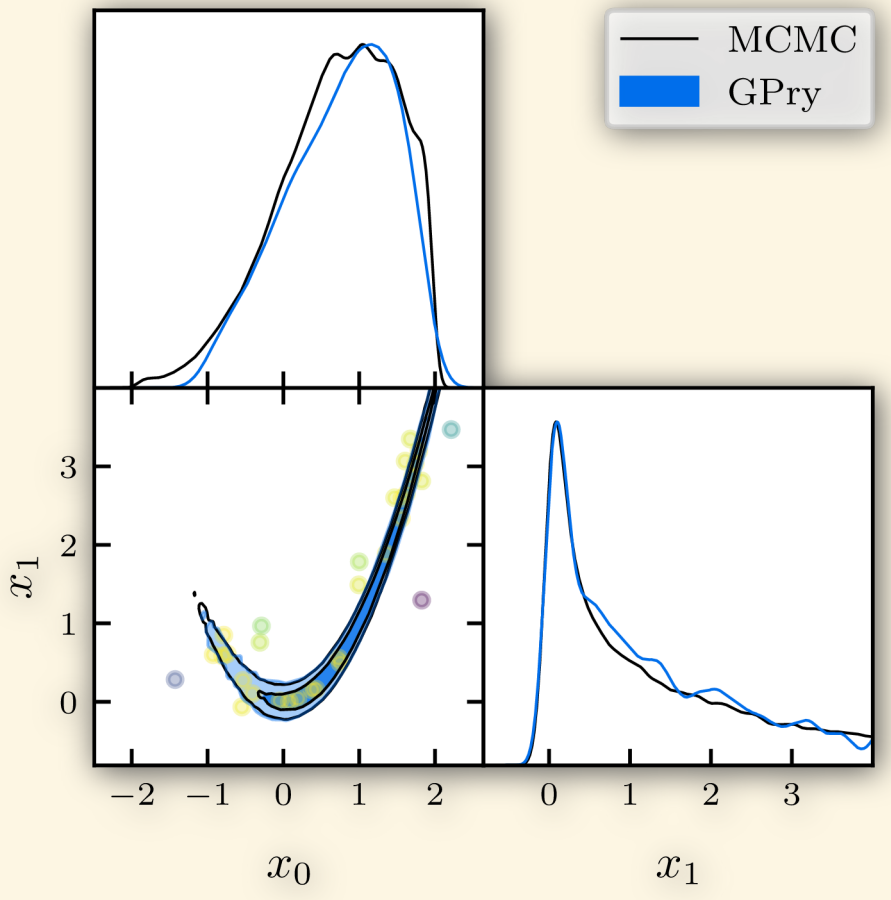
Examples: non-Gaussians



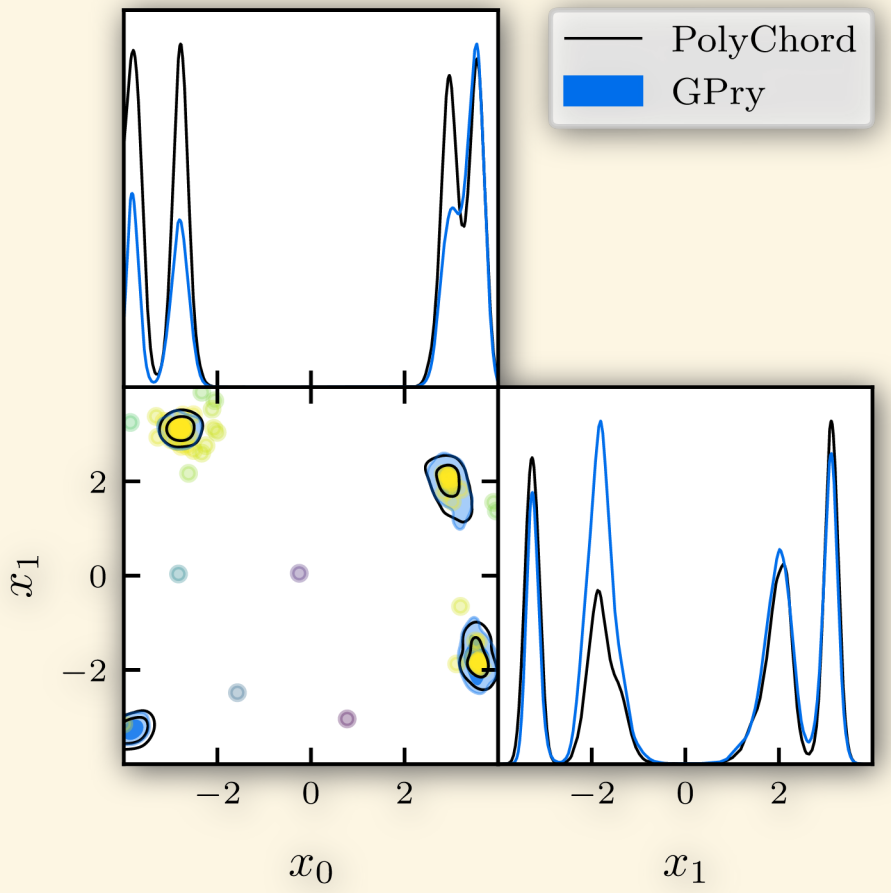


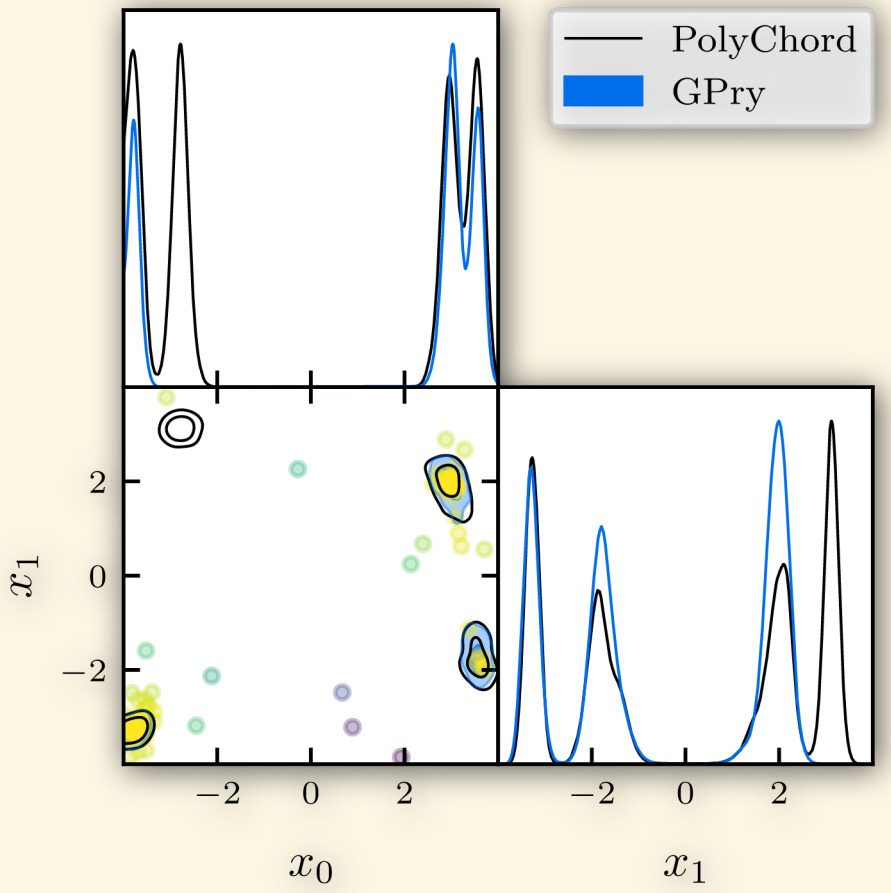
Examples: non-Gaussians (II)





Bad stuff: multimodality

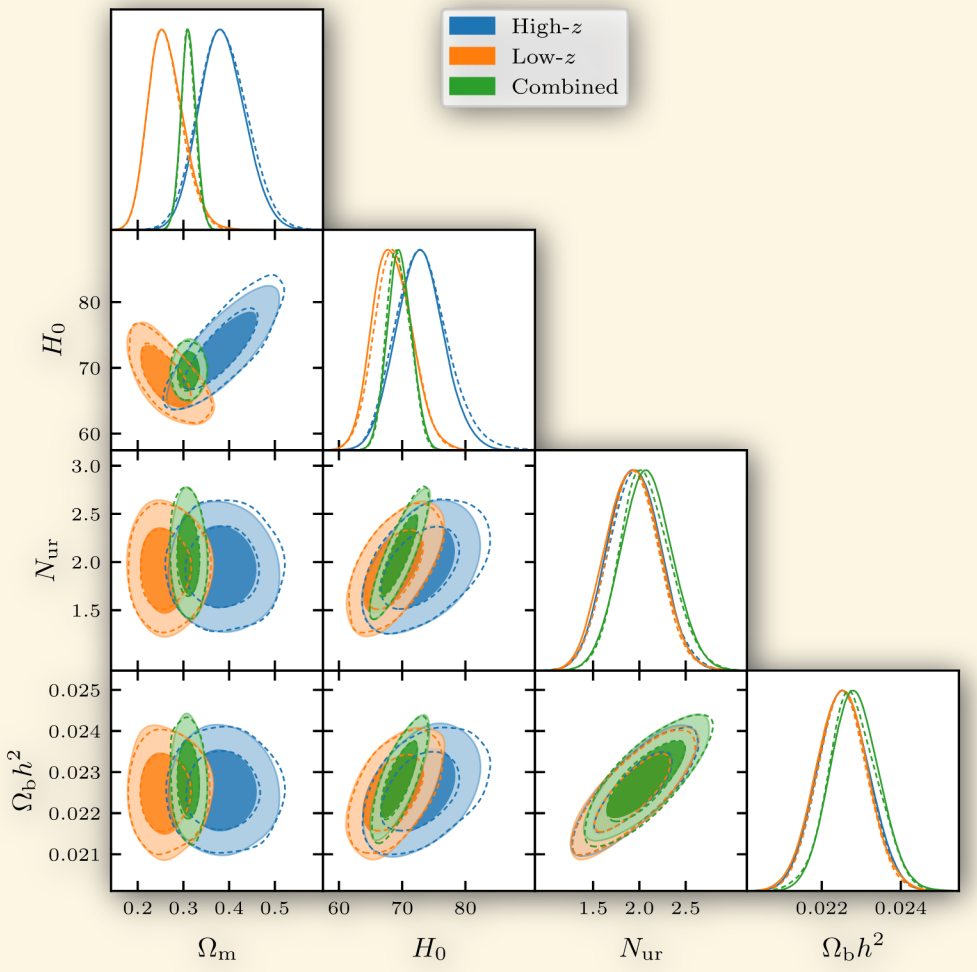




We expect to fix this with a less greedy acquisition procedure (WIP)

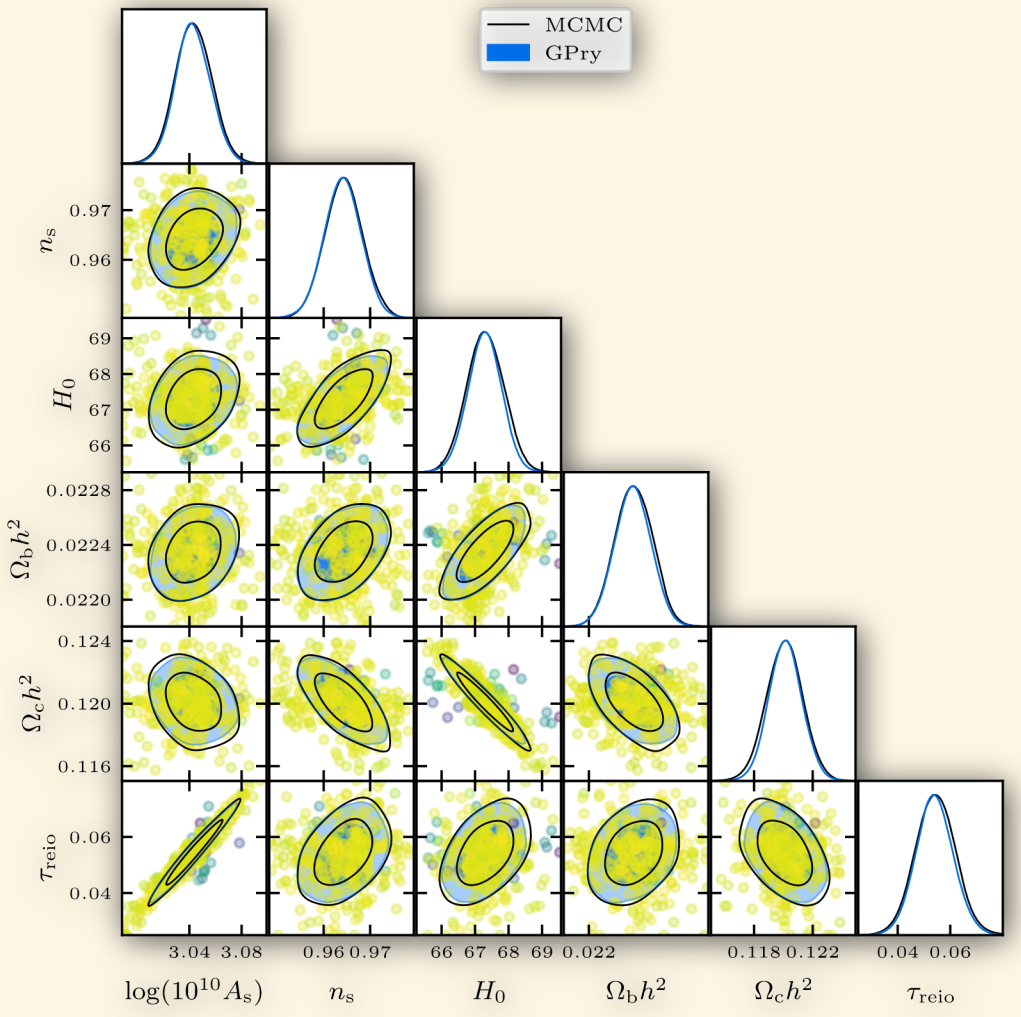
Cosmology:

3-parameter ΛCDM + $N_{\rm eff}$ with BBN + BAO (low- and high-z)



Cosmology:

6-parameter ΛCDM with Planck Lite 2018



Conclusions & outlook

- $\mathcal{O}(>100)$ fewer evaluations, and overhead that makes it way faster than traditional MC for posterior evaluation costs of 1/10 seconds for $d < 8$, 1 second for $d < 16$...
- So far we focused on modelling, but there are a lot of interesting avenues to pursue (integration with VBMC by L. Acerbi et al., clever strategies for active learning and hyperparameter fitting...)
- Working in application to inference on GW sources (R. Buscicchio, G. Nardini)
- Current limitations:
  - Strong non-Gaussianity in high dim (in general hard for approximate methods) – looking at normalising flows.
  - Multimodality (more informative initialisation, no SVM, clustering and multiple GP models...)
  - Costs scale badly with dimensionality, so for $d > 20$ we are working to extend it (GPU/approximate GP libraries).

Play with it!

https://pypi.org/project/gpry/ – https://gpry.readthedocs.io/en/latest/