HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

Simulation-Based Inference for Beam Parameter Inversion

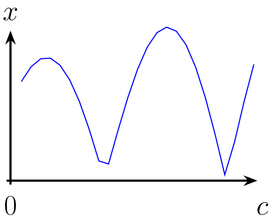
HELMHOLTZ AI

Peter Steinbach

HZDR / ML@HZDR, Dec 6, 2021

A common situation with simulations

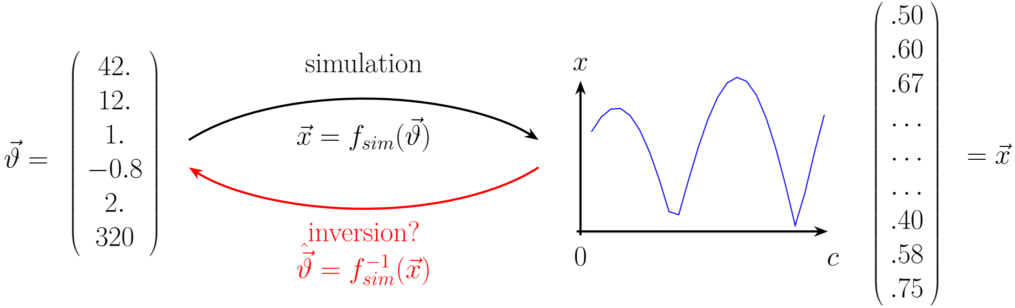
$$\vec{\vartheta} = \begin{pmatrix} 42. \\ 12. \\ 1. \\ -0.8 \\ 2. \\ 320 \end{pmatrix} \xrightarrow[\vec{x} = f_{sim}(\vec{\vartheta})]{\text{simulation}}$$



$$\begin{pmatrix} .50 \\ .60 \\ .67 \\ \cdots \\ \cdots \\ \cdots \\ .40 \\ .58 \\ .75 \end{pmatrix} = \vec{x}$$

- simulations used in many domains (physics, biology/medicine, chemistry, epidemiology, ...)
- approaches to simulations vary (mechanistic, agent based, distribution based, ...)
- simulations can be computationally challenging
- here: simulations = **forward process**

Inversion of Simulations?



- inverse process hard to do (if at all tried)
- often, only single observables “fitted”
- simulations updated based on singular observables
- considerable human tuning involved (heuristics)

Bayes Law

$$p(\vec{\vartheta}|\vec{x}) = \frac{p(\vec{x}|\vec{\vartheta}) \cdot p(\vec{\vartheta})}{\int p(\vec{x}|\vec{\vartheta})p(\vec{\vartheta})d\vec{\vartheta}}$$

Bayes Law

$$p(\vec{\vartheta}|\vec{x}) = \frac{\overbrace{p(\vec{x}|\vec{\vartheta})}^{\text{likelihood}} \cdot p(\vec{\vartheta})}{\int p(\vec{x}|\vec{\vartheta})p(\vec{\vartheta})d\vec{\vartheta}}$$

Bayes Law

$$p(\vec{\vartheta}|\vec{x}) = \frac{\overbrace{p(\vec{x}|\vec{\vartheta})}^{\text{likelihood}} \cdot \overbrace{p(\vec{\vartheta})}^{\text{prior}}}{\int p(\vec{x}|\vec{\vartheta}) p(\vec{\vartheta}) d\vec{\vartheta}}$$

Bayes Law

$$\underbrace{p(\vec{\vartheta}|\vec{x})}_{\text{posterior}} = \frac{\overbrace{p(\vec{x}|\vec{\vartheta})}^{\text{likelihood}} \cdot \overbrace{p(\vec{\vartheta})}^{\text{prior}}}{\int p(\vec{x}|\vec{\vartheta})p(\vec{\vartheta})d\vec{\vartheta}}$$

Bayes Law

$$\underbrace{p(\vec{\vartheta}|\vec{x})}_{\text{posterior}} = \frac{\overbrace{p(\vec{x}|\vec{\vartheta})}^{\text{likelihood}} \cdot \overbrace{p(\vec{\vartheta})}^{\text{prior}}}{\int p(\vec{x}|\vec{\vartheta})p(\vec{\vartheta})d\vec{\vartheta}}$$

- likelihood $p(\vec{x}|\vec{\vartheta})$ provided by (forward) simulation
- prior $p(\vec{\vartheta})$ given by how we sample the simulation parameters $\vec{\vartheta}$

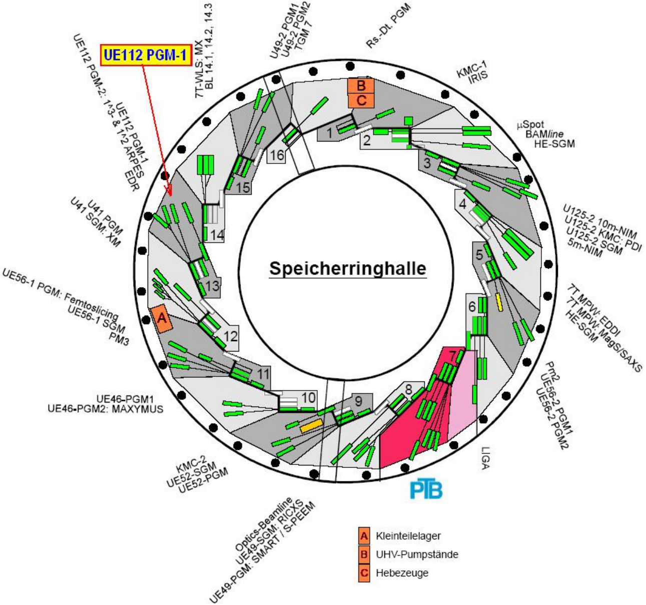
Bayes Law

$$\underbrace{p(\vec{\vartheta}|\vec{x})}_{\text{posterior}} = \frac{\overbrace{p(\vec{x}|\vec{\vartheta})}^{\text{likelihood}} \cdot \overbrace{p(\vec{\vartheta})}^{\text{prior}}}{\int p(\vec{x}|\vec{\vartheta}) p(\vec{\vartheta}) d\vec{\vartheta}}$$

- likelihood $p(\vec{x}|\vec{\vartheta})$ provided by (forward) simulation
- prior $p(\vec{\vartheta})$ given by how we sample the simulation parameters $\vec{\vartheta}$

Goal: predict posterior to the best of our abilities!

Why do I care?

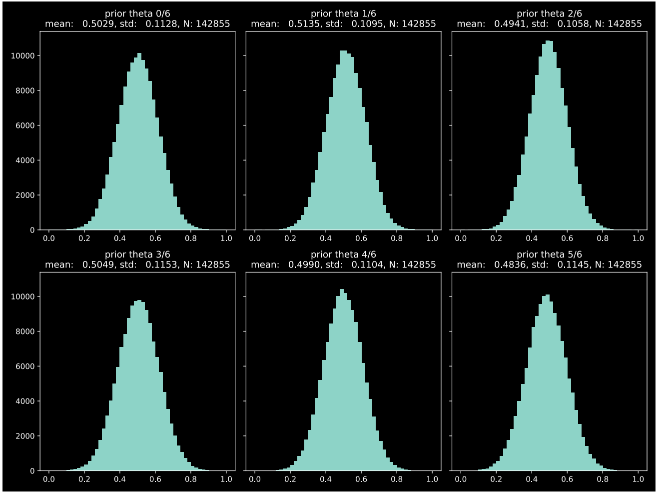


Scientific Question

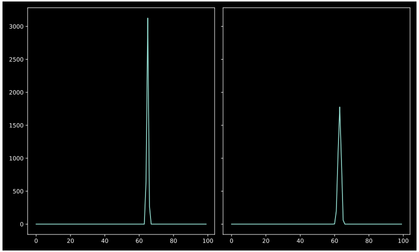
Given a beam profile, what were the beamline optics parameters that likely produced it?

- beamline UE112 PGM-1 at BESSY
- beam characteristics fixed at electron storage ring outlet
- forward simulations by rayUI

My data from a surrogate



(forward) simulation parameters $\vec{\vartheta} \in \mathbb{R}^6$



knife-edge scans to quantify beam quality at experimental station, $\vec{x} \in \mathbb{R}^{200}$

(surrogate: multivariate normal of 2 dimensions plus 2 extra variables)

Conditional Invertible Neural Networks

arxiv:1907.02392, Code on github

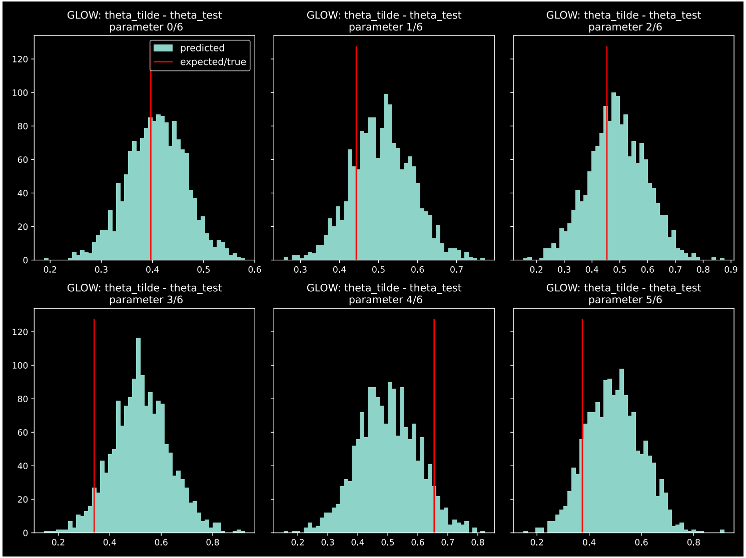
GUIDED IMAGE GENERATION WITH CONDITIONAL INVERTIBLE NEURAL NETWORKS

Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, Ullrich Köthe
Visual Learning Lab Heidelberg

ABSTRACT

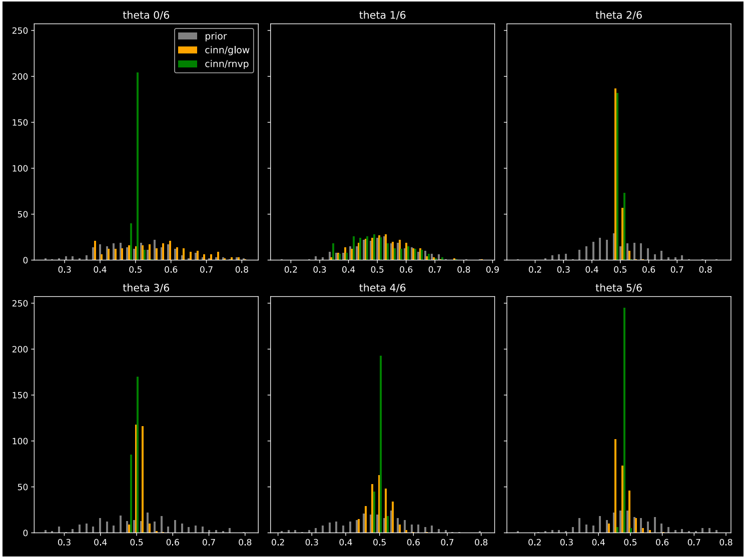
In this work, we address the task of natural image generation guided by a conditioning input. We introduce a new architecture called conditional invertible neural network (cINN). The cINN combines the purely generative INN model with an unconstrained feed-forward network, which efficiently preprocesses the conditioning input into useful features. All parameters of the cINN are jointly optimized with a stable, maximum likelihood-based training procedure. By construction, the cINN does not experience mode collapse and generates diverse samples, in contrast to e.g. cGANs. At the same time our model produces sharp images since no reconstruction loss is required, in contrast to e.g. VAEs. We demonstrate these properties for the tasks of MNIST digit generation and image colorization. Furthermore, we take advantage of our bi-directional cINN architecture to explore and manipulate emergent properties of the latent space, such as changing

cINN Inference on the validation set



- cINN provides posterior that can be sampled
- extract Maximum a posteriori estimation (MAP) estimate by mean/median

cINN MAPs on the validation set



- training:
  - 30 epochs only
  - fixed arch: 8 layers
  - 256 units per dense layer
- inference: 256 draws per validation sample, MAP by mean
- good: posterior stays within prior support
- to improve: posterior misses out for some dimensions

What is going on?

core assumption(s)

- network f_{cinn} is sufficiently expressive
- as $N_{\text{simulations}} \rightarrow \infty$, network allows mapping of $\vec{x}$ onto $p(\vec{\vartheta}|\vec{x})$
- training dataset imposes the entire “truth” through implicit prior $p_{\text{simulation}}(\vec{\vartheta})$

What is going on?

core assumption(s)

- network f_{cinn} is sufficiently expressive
- as $N_{\text{simulations}} \rightarrow \infty$, network allows mapping of $\vec{x}$ onto $p(\vec{\vartheta}|\vec{x})$
- training dataset imposes the entire "truth" through implicit prior $p_{\text{simulation}}(\vec{\vartheta})$

Let's reconsider

- goal: infer $\vec{\vartheta}|\vec{x_o}$ on observation $\vec{x_o}$
- **but:** the global learned posterior may not be too informative at $p(\vec{\vartheta}|\vec{x_o})$ (as we fixed the prior)

What is going on?

core assumption(s)

- network f_{cinn} is sufficiently expressive
- as $N_{\text{simulations}} \to \infty$, network allows mapping of $\vec{x}$ onto $p(\vec{\vartheta}|\vec{x})$
- training dataset imposes the entire “truth” through implicit prior $p_{\text{simulation}}(\vec{\vartheta})$

Let’s reconsider

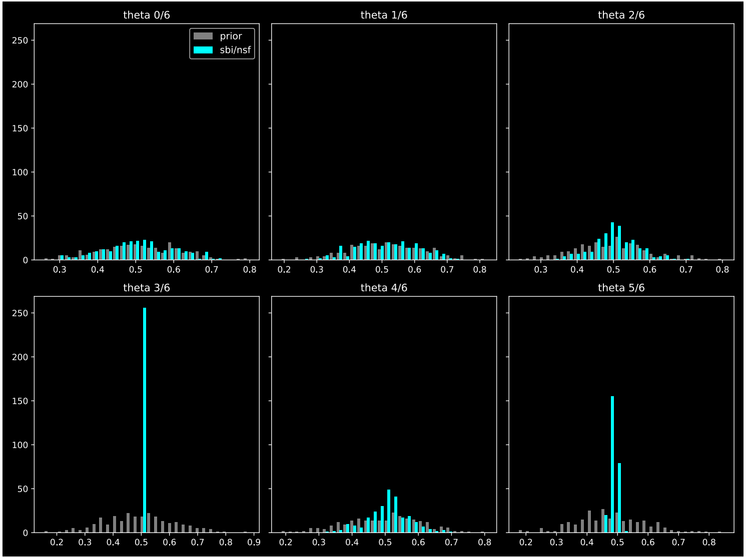
- goal: infer $\vec{\vartheta}|\vec{x_o}$ on observation $\vec{x_o}$
- **but:** the global learned posterior may not be too informative at $p(\vec{\vartheta}|\vec{x_o})$ (as we fixed the prior)

sequential neural density estimation (SNPE) with

www.mackelab.org/sbi

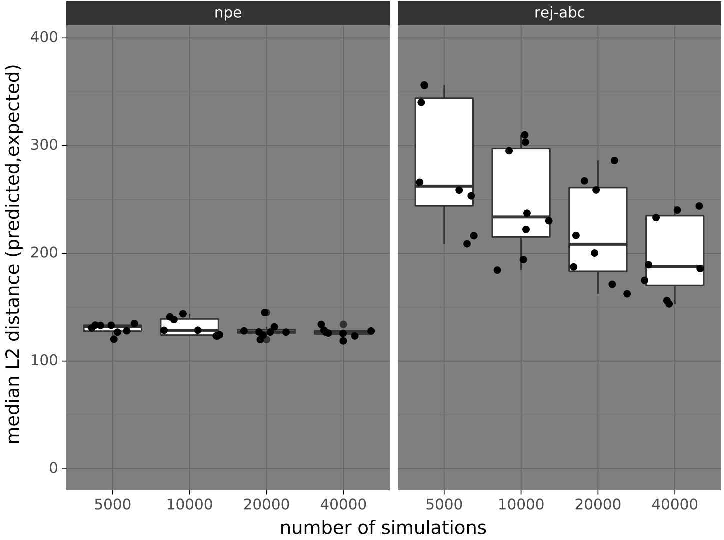
(Sequential) neural density estimation

NPE: `sbi` MAPs on the validation set



- training:
  - >100 epochs only
  - fixed arch: 8 layers, neural spine flow Durkan et al, 2019
  - 256 units per layer
- training: only one round (NPE) to produce amortized posterior
- inference: 256 draws per validation sample, MAP by mean
- good: posterior stays within prior support
- to improve: posterior misses out for two dimensions (expected)

presently: comparison to other methods

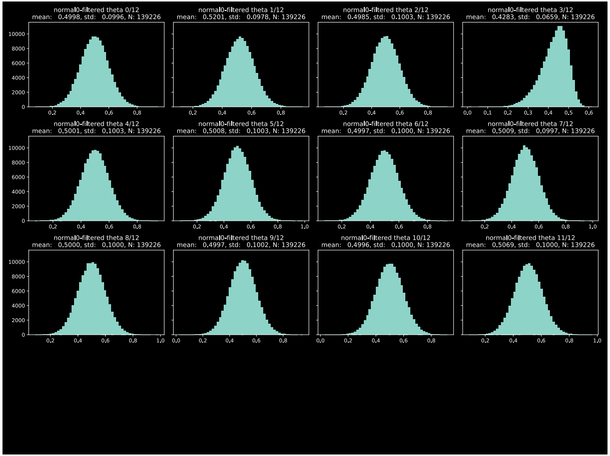


- using surrogate simulation of a beamline (projected multivariate normal)
- integrating into `sbibm`
- plan: compare to more SBI based approaches (sequential and non-sequential)

Summary

- simulation-based inference (`sbi`) can be a tool for inversion (either NPE with cINN or SNPE)
- `sbi` is a light weight and yet flexible framework
- neural spline flow used in SNPE appears more flexible
- quality control & comparison:
  - currently: how do other methods compare (Lueckmann et al, 2021)
  - sample based metrics (Naeem et al, 2020)
  - simulation based calibration (Talts et al, 2018)

the real world



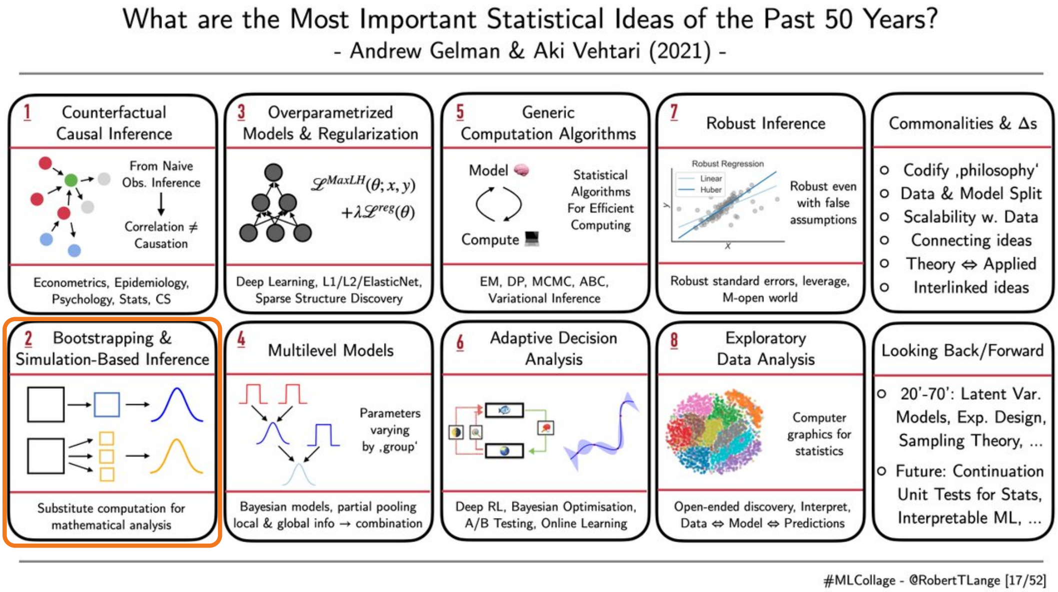
reproducibility, openness & team work = key!

(all results from above were from toy simulations)

Further Reading

- nice blog post with pytorch code samples
- “Normalizing Flows for Probabilistic Modeling and Inference,” G. Papamakarios et al, arXiv:1912.02762, 2019.
- “Normalizing Flows: An Introduction and Review of Current Methods”, I. Kobyzev et al, arXiv:1908.09257, 2019.
- “Glow: Generative Flow with Invertible 1x1 Convolutions”, Kingma et al, arXiv:1807.03039, 2018.

Backup



Simulation-based inference discovered recently

(Cranmer et al, 2020, Gelman & Vehtari, 2021)

SNPE

Algorithm 1 APT with per-round proposal updates

Input: simulator with (implicit) density $p(x|\theta)$, data x_o, prior $p(\theta)$, density family q_ψ, neural network $F(x, \phi)$, simulations per round N, number of rounds R.

$\tilde{p}_1(\theta) := p(\theta)$
for $r = 1$ **to** R **do**
  for $j = 1$ **to** N **do**
    Sample $\theta_{r,j} \sim \tilde{p}_r(\theta)$
    Simulate $x_{r,j} \sim p(x|\theta_{r,j})$
  end for
  $$\phi \leftarrow \operatorname*{argmin}_\phi \sum_{i=1}^{r} \sum_{j=1}^{N} -\log \tilde{q}_{x_{i,j},\phi}(\theta_{i,j}) \qquad \text{using (2)}$$
  $\tilde{p}_{r+1}(\theta) := q_{F(x_o,\phi)}(\theta)$
end for
return $q_{F(x_o,\phi)}(\theta)$

- sequentially update prior to form a proposal prior $\tilde{p}(\vartheta)$ (and a $\tilde{p}(\vartheta|\vec{x})$)
- loss function has to be adapted
- **cINNs can be used as conditional density estimator**
- Greenberg et al, 2019:
  "Learning with such 'atomic' proposals has an intuitive interpretation: we are training the network to solve multiple choice test problems, of the format "which of these ϑ's generated this x?""