Master's Thesis : Differentiable Surrogate Models to Solve Nonlinear Inverse Problems

Auteur : Vandegar, Maxime
Promoteur(s) : Louppe, Gilles
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en informatique, à finalité spécialisée en "intelligent systems"
Année académique : 2019-2020
URI/URL : http://hdl.handle.net/2268.2/9064

University of Liège - Faculty of Applied Sciences

Departement of Electrical Engineering and Computer Science

Differentiable Surrogate Models to Solve Nonlinear Inverse Problems

Graduation Studies conducted for obtaining the Master's degree in Computer Science and Engineering

by

Maxime Vandegar

Supervisor:
Prof. Gilles Louppe

Academic Year 2019-2020

Abstract

Doing inference on a model defining an implicit likelihood that is not known in closed form is called likelihood-free inference. This occurs frequently in engineering and science domains where a simulator is used as a generative model of data, but the likelihood of the generated data is not known and is intractable. Given observed data, we combine the idea of hierarchical Bayesian modeling, empirical Bayes, and neural density estimation with normalizing flow to first learn a surrogate approximation of the model likelihood and then, to learn a prior distribution over the model parameters. The learned prior and the surrogate likelihood further allow to learn a posterior distribution for each observation. This is a general approach to likelihood-free inference, and is especially useful in settings where the simulator is too costly to run at inference time. We show the applicability of our methods on a real physical problem from high energy physics (HEP).

Acknowledgments

I am sincerely grateful to Professor Gilles Louppe for introducing me to the world of likelihood-free inference and more generally, to the world of research. I am grateful for his availability and his precious comments during this work. I am forever thankful for his trust and the opportunities that he gave me.

I am grateful to Michael Kagan for hosting me at SLAC National Accelerator Laboratory and the European Organization for Nuclear Research. Michael has been much more than an internship advisor. He has helped me to take my first steps into the world of research and to make important decisions at an early stage of my career. I would like to thank him for his availability and for stimulating exchanges during this work, from beginning to end.

I am also thankful to all members of the jury for taking the time to read and evaluate this dissertation. In particular, I would like to thank Antoine Wehenkel for insightful discussions. Beyond this work, I want to take this opportunity to thank Professor Gilles Louppe and Professor Pierre Geurts for their teachings and for introducing me to machine learning and deep learning.

Thank you to Nicole Hartman for insightful discussions and for teaching me about physics. Nicole has been truly supportive and made my stay at CERN much more enjoyable. Thank you to all my friends at CERN that made my internship unforgettable.

Thank you to my family and more particularly to my parents for making everything possible.

Finally, thank you to Justine for her unconditional support.

Contents

1. Introduction

Problem Statement Many scientific and technology fields are interested in learning source data from corrupted observed data which is called solving an inverse problem. For example, topics in image restoration are interested in retrieving a clean image from a noisy one or a high resolution image from a low resolution one. Similarly, some astrophysicists are working on deblurring telescope images such as black hole images. In high energy physics, a typical example would be to retrieve true particle properties given measured properties that is, correct the corruption introduced by the measurement tools.

At the same time, many scientific and technology fields have developed computer simulators of physical processes such as robotic simulators, galaxy-galaxy lenses simulators and particle collision simulators. A simulator is a computer program that maps a vector of parameters x to a vector of parameters y, typically built using known rules of physical law. They may have free-parameters that are tuned over time so that simulations match real experiments. Usually, simulators are stochastic and thus, they may produce multiple different $y's$ given the same x. The stochasticity is due to the use of random variables generated by random number generators. These variables are called latent variables and usually represent a true underlying physical quantity. For example, while a coin thrown is often thought as stochastic, it is actually a process that can be fully determined given the initial orientation of the coin, the force applied to it and many other parameters (Tran et al., 2017a). Given the force applied to the coin, a simulator could stochastically simulate the thrown by modelling all unknowns by random variables.

While these simulators are computer programs, they may be very complex — some of them results from decades of research and engineering — and may have long run times. Simulating an event is called the forward process, it allows to observe data y given parameters x. The aim of this dissertation is to estimate solutions to the backward process that is, retrieve plausible $x's$ that could have produced the observed y. This area of research utilizes techniques such as maximum likelihood and posterior inference which aims to learn the posterior $p(x|y)$. When a simulator is used to aid this process, this is often called simulator-based inference. The stochastic simulator defines an implicit likelihood function $p(y|x)$ which is often intractable as computing it would require integration over all the latent variables that could produce y:

$$p(y|x) = \int p(y, z|x)dz. \tag{1}$$

Experts have therefore mostly developed algorithms that do not need the likelihood function in closed-form. This area of research is thus called likelihood-free inference.

Many current inference methods suffer from the fact that they need the simulator at inference and when the simulator has a long run time, most of the inference time budget is spent on running simulations. This means that in order to run the simulator as much as needed by the inference method in order to obtain the desired accuracy, inference can be arbitrary long. There also exist settings when you have raw data but do not have access to the simulator (especially in this era of Big Data) which makes the use of many inference methods impossible. Finally, in addition to the simulator, most current inference methods need a prior distribution $p(x)$ during inference, which cannot be always postulated easily. Therefore, developing inference methods that do not need the simulator at inference time, nor a predefined prior distribution is of much interest.

Research question To sum up, the research question that we will try to answer throughout this dissertation is:

How to perform inference when the simulator is too costly to run at inference time and when no likelihood function is available in closed-form?

Contributions. Given a set of training data, for instance obtained from running the simulator, we first learn a model of the simulator — called a surrogate — with a powerful differentiable statistical model called normalizing flow (Rezende and Mohamed, 2015) described in Section 2.4. This model allows us to explicitly evaluate an approximated likelihood $p(y|x)$ and being differentiable, allows to use gradient-based optimization.

Given such a surrogate, our contributions are as follows:

- **Point Estimation.** Given a corrupted observation, we retrieve plausible parameters[1] that could have generated this observation. We also show that given a set of observations, the set of individual learned parameters can produce a plausible empirical prior distribution of the parameters.

- **Density Estimation.** Given a single observation, we discuss four methods to learn a distribution of plausible parameters that could have generated the observation. This allows to quantify uncertainty over parameters, which is significantly more useful than point estimates in decision making and scientific settings as it allows an understanding of the significance of a prediction. The learned distribution can be fast sampled and differentiated. When it is modeled by a normalizing flow, its density can be evaluated as well.

- **Empirical Bayes.** Given a set of observations, we generalize one of the method from the previous point in order to learn a plausible prior distribution over the unseen source parameters. We successfully apply this method in correcting the detector effects in a real high energy physics problem.

- **Amortized Inference.** Given multiple observations, we use our results from the previous point in order to learn a prior distribution over the unseen source data. Then, for each observation, we use the learned prior and the surrogate likelihood in order to approximate the posterior.

Notations. Throughout the rest of this dissertation, vectors are represented in bold lowercase while tensors are represented in bold uppercase except for neural network and distribution parameters that are not represented in bold. When there is no ambiguity, a function $f : X \to Y$ is noted $f(\cdot)$ and when the function is parametrized by parameters ϕ that we are not interested in, we omit them. Otherwise, we represent the function by $f_\phi(\cdot)$. All norms are represented by $||\cdot||$ and the gradient operator whose components are the partial derivatives of a function $f_\phi(\cdot)$ with respect to all the parameters in ϕ is noted as $\nabla_\phi f$. Finally, the probability density function of a random variable $\mathbf{x}$ is noted $p(\mathbf{x})$.

1. The word parameters refers to a single vector of variables taken as input by the simulator. This is a single instance and should not be confused with a set of parameters.

Conventions. In the likelihood-free inference literature, the latent variables are often noted $\boldsymbol{\theta}$ while the observations are referred by $\mathbf{x}$. The probabilistic programming and linear inverse problems literature rather use $\mathbf{x}$ and $\mathbf{y}$ for the latent and observations respectively. In the Empirical Bayes literature, where the latent variables are considered as random variables modeled by a parametric distribution, a common notation is to refer to the observations by $\mathbf{y}$, the latent variables by $\mathbf{x}$ and the parameters defining the underlying distribution on $\mathbf{x}$ by θ. This is this notation that we will use throughout this dissertation.

Reproducibility. Code to reproduce the experiments and figures will be available in a GitHub repository at `www.github.com/MaximeVandegar/LikelihoodFreeEB`.

Roadmap. We introduce background concepts in Section 2 and cover related work in Section 3. Then, in Section 4 we show how we design surrogate models and in Section 5, we introduce our methods. In Section 6 we evaluate our methods on four toy problems inspired by the literature and in Section 7 we solve a scientific application in high energy physics (HEP) showing the applicability of our methods on real problems.

2. Background

2.1 Inverse Problems

Linear Inverse Problems In many scientific and technology domains, reconstructing a target signal $\mathbf{x} \in R^n$ from a corrupted observation $\mathbf{y} \in R^m$ can be represented as:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\eta} \tag{2}$$

where $\mathbf{A} \in R^{m \times n}$ is a corruption matrix and $\boldsymbol{\eta}$ possible additive noise. Usually, $m < n$ and even without noise, this is an ill-posed problem as $\mathbf{A}$ is not invertible or may be ill-conditioned. In most common problems, $\mathbf{A}$ is known. For example, in compressed sensing, it is the measurement matrix and in image denoising, it is the identity matrix.

In order to solve the corruption problem, two main approaches are commonly used. The first one is maximum a posteriori (MAP):

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}) \tag{3}$$

where $p(\mathbf{y}|\mathbf{x})$ is the likelihood — often known — of the corruption model and $p(\mathbf{x})$ is a regularizer that constrains the solution space of the problem otherwise ill-posed. A broad class of problem-dependent off-the-shelf regularizer have been introduced and studied such as the total variation (TV) norm. Other approaches also rely on learned-based regularizer. A main challenge is to find pairs of likelihood and prior function that lead to a convex optimization problem. When this is the case and when both $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x})$ are differentiable, Equation 3 can be solved recursively:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \gamma \nabla_{\mathbf{x}} (\log p(\mathbf{y}|\mathbf{x}_t) + \log p(\mathbf{x}_t)) \tag{4}$$

where γ is a learning rate.

The second approach is to solve a constrained problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{A}\mathbf{x}, \mathbf{y}), \quad s.t. \quad \mathbf{x} \in S \tag{5}$$

where $\mathcal{L}$ is an appropriate loss between regenerated values and observations — usually the minimum mean squared error (MMSE) — and $\mathcal{S}$ is a set that captures prior knowledge about $\mathbf{x}$.

Nonlinear Inverse Problems While linear inverse problems are already challenging to solve, nonlinear problems are even more challenging and as opposed to linear problems, they often results from black-box corruption processes:

$$\mathbf{y} = s(\mathbf{x}) \tag{6}$$

where s in a black-box corruption process, often stochastic. Therefore, in general, well-studied problem-dependent off-the-shelf regularizers cannot be used.

While using traditional supervised learning techniques to learn a mapping from $\mathbf{y}$ to $\mathbf{x}$ would not produce good results as the mapping is ambiguous, normalizing flows as well as a recent class of neural networks called invertible neural networks (INNs) have had particular interest in solving this class of problems as they allow to learn a stochastic mapping from $\mathbf{y}$ to $\mathbf{x}$. Some INNs allow to directly learn the posterior $p(\mathbf{x}|\mathbf{y})$ (Ardizzone et al., 2018).

2.2 Empirical Bayes

Given observed values $\mathbf{Y} = \{\mathbf{y}_n\}_{n=1}^N$ of random variables that depend on unobserved latent variables $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$, empirical Bayes aims at finding a complete-data likelihood of the model $p(\mathbf{x}, \mathbf{y}|\theta)$ parameterized by hyperparameters θ.

All the observed $\mathbf{y}'s$ are treated as if they were from the same population, which is implicitly the case in the Bayes model as the $\mathbf{x}'s$ come from the same prior distribution (Casella, 1985).

In empirical Bayes, all the information about θ is contained in the marginal likelihood distribution:

$$p(\mathbf{Y}) = \int p(\mathbf{X}, \mathbf{Y}) \mathbf{dX}. \tag{7}$$

which can be rewritten as

$$p(\mathbf{Y}|\theta) = \int p(\mathbf{X}, \mathbf{Y}|\theta) \mathbf{dX}. \tag{8}$$

Instead of quantifying uncertainty over θ:

$$p(\theta|\mathbf{Y}) = \int p(\theta, \mathbf{X}|\mathbf{Y}) \mathbf{dX} \tag{9}$$

empirical Bayes often set θ to their most likely values and therefore, is also referred as maximum marginal likelihood.

Dempster et al. (1977) proposed the well known EM-algorithm that finds the maximum likelihood θ_{MLE} of Equation 8 from observations $\mathbf{Y}$ by iteratively alterning between an expectation step and a maximization step.

The expectation steps defines $Q(\theta, \theta^{(i-1)})$ as the expected value of the complete-data log-likelihood with respect to the unknown variables $\mathbf{X}$ given the observations and the current estimate of the parameters $\theta^{(i-1)}$:

$$Q(\theta, \theta^{(i-1)}) = \mathbb{E}_{p(\mathbf{X}|\mathbf{Y}, \theta^{(i-1)})}[\log p(\mathbf{X}, \mathbf{Y}|\theta)] \tag{10a}$$

$$= \int \log p(\mathbf{X}, \mathbf{Y}|\theta) p(\mathbf{X}|\mathbf{Y}, \theta^{i-1}) \mathbf{dx} \tag{10b}$$

Then, the maximization step finds θ^i that maximizes this function:

$$\theta^i = \arg\max_{\theta} Q(\theta, \theta^{(i-1)}) \tag{11}$$

Many researchers have proved that each iteration is guaranteed to increase the marginal likelihood and thus, the algorithm is guaranteed to converge to a local maximum.

2.3 Variational Inference

As opposed to empirical Bayes, in the fully Bayesian setting, the prior is postulated to be known and the probabilistic model defines a joint $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$ — here we assume that we can evaluate the likelihood and therefore the joint. Inference consists in computing the posterior $p(\mathbf{x}|\mathbf{y})$ which can be written as:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \tag{12}$$

However, the evidence $p(\mathbf{y})$ is often not known in closed form or is intractable as it requires to marginalizing out the latent variables:

$$p(\mathbf{y}) = \int p(\mathbf{y}, \mathbf{x}) \mathbf{dx}. \tag{13}$$

Variational inference is a common machine learning framework that cast this inference problem into an optimization problem. The idea is to find a member q from a distribution family $\mathcal{Q}$ over the latent variables $\mathbf{x}$ that is the closest in Kullback-Leibler (KL) divergence to the true posterior:

$$q^*(\mathbf{x}|\mathbf{y}) = \arg\min_{q \in \mathcal{Q}} KL(q(\mathbf{x}|\mathbf{y})||p(x|\mathbf{y})) \tag{14a}$$

$$= \arg\min_{q \in \mathcal{Q}} KL(q(\mathbf{x}|\mathbf{y})||\frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}) \tag{14b}$$

$$= \arg\min_{q \in \mathcal{Q}} [KL(q(\mathbf{x}|\mathbf{y})||p(\mathbf{x})) - \mathbb{E}_q \log p(\mathbf{y}|\mathbf{x}) + \underbrace{\mathbb{E}_q \log p(\mathbf{y})}_{\log p(\mathbf{y})}] \tag{14c}$$

Equation 14c cannot be optimized directly as it still requires to compute the evidence $p(\mathbf{y})$. Nonetheless, the evidence is constant with respect to q:

$$KL(q(\mathbf{x}|\mathbf{y})||p(\mathbf{x}|\mathbf{y})) = \underbrace{-\mathbb{E}_q \log p(\mathbf{y}|\mathbf{x}) + KL(q(\mathbf{x}|\mathbf{y})||p(\mathbf{x}))}_{-ELBO(q)} + \log p(\mathbf{y}) \tag{15}$$

and therefore, we can optimize the evidence lower bound (ELBO), an alternative to the KL divergence up to a constant. Maximizing the ELBO, is equivalent to minimizing the KL divergence. The first term of the ELBO is the expected log likelihood with respect to the latent variables given the observations. It is also called the reconstruction term and encourages q to place its mass on latent variables that explain the data. The second term is a negative KL divergence between q and the prior which encourages the variational distribution to stay close to the prior.

Variational Inference and the EM Algorithm

Neal and Hinton (2000) showed interesting connections between the EM algorithm and variational inference. They show that in the EM algorithm and its variants, both the E and M steps increase the same function:

$$F(q, \theta) = \mathbb{E}_{q(\mathbf{x}|\mathbf{Y})} \log p(\mathbf{x}, \mathbf{Y}|\theta) + H(q) \tag{16}$$

where $H(q)$ is the entropy of the distribution q.

They show that if a local maximum of F occurs at q^* and θ^*, then a local maximum of the marginal likelihood $p(\mathbf{Y}|\theta)$ occurs at θ^* as well. We can therefore maximize the marginal likelihood by maximizing F which can be rewritten as:

$$F(q, \theta) = -KL(q(\mathbf{x}|\mathbf{Y})||p(\mathbf{x}|\mathbf{Y}, \theta)) + \log p(\mathbf{Y}|\theta). \tag{17}$$

2.4 Normalizing Flows

In variational inference, in order to find a variational distribution q that approximates the posterior, one should postulate a distribution family $\mathcal{Q}$ flexible enough to model $p(\mathbf{x}|\mathbf{y})$.

Furthermore, it should be computational efficient to evaluate the density $q(\mathbf{x}|\mathbf{y})$ as the optimization problems defined in Equation 14 requests its computation in the ELBO term. Normalizing flows (Rezende and Mohamed, 2015) are powerful statistical model well designed for this task among others.

The idea is to construct a bijective mapping $f : \mathbb{R}^d \to \mathbb{R}^d$ such that $\mathbf{y} = f(\mathbf{z})$ where $\mathbf{z}$ is a variable with a usually simple base density $p(\mathbf{z})$. As f is invertible:

$$\mathbf{z} = f^{-1}(\mathbf{y}) = f^{-1}(f(\mathbf{z})). \tag{18}$$

. In this setting, given $\mathbf{y}$ one can evaluate its density by simply inverting it and keeping track of the Jacobian of the transformation:

$$\ln(q(\mathbf{y})) = \ln(p(\mathbf{z})) + \ln|\det \frac{\partial f^{-1}(\mathbf{y})}{\partial \mathbf{y}}| \tag{19a}$$

$$= \ln(p(f^{-1}(\mathbf{y}))) + \ln|\det \frac{\partial f^{-1}(\mathbf{y})}{\partial \mathbf{y}}| \tag{19b}$$

where Equation 19a results from the change of variable theorem. Beyond evaluating densities, one can use the flow — once trained — to sample from the target density (represented in Figure 1):

$$\mathbf{y} = f(\mathbf{z}), \quad \mathbf{z} \sim p(\mathbf{z}). \tag{20}$$

Given a dataset of samples $\mathbf{Y} = \{\mathbf{y}_n\}_{n=1}^N$ from the target distribution $p(\mathbf{y})$, the flow can be trained by minimizing the Kullback–Leibler divergence between the flow density and the target density:

$$f = \arg\min_q \mathrm{KL}(p(\mathbf{y})||q(\mathbf{y})) \tag{21a}$$

$$= \arg\min_q \mathbb{E}_{p(\mathbf{y})}[\ln q(\mathbf{y})] + \underbrace{\cancel{H(p)}}_{Cst} \tag{21b}$$

$$= \arg\min_f \mathbb{E}_{p(\mathbf{y})}[\ln(p(f^{-1}(\mathbf{y}))) + \ln|\det \frac{\partial f^{-1}(\mathbf{y})}{\partial \mathbf{y}}|] \tag{21c}$$

$$\approx \arg\min_f \frac{1}{N}\sum_{n=1}^{N} \ln(p(f^{-1}(\mathbf{y}_n))) + \ln|\det \frac{\partial f^{-1}(\mathbf{y}_n)}{\partial \mathbf{y}_n}| \tag{21d}$$

In Equation 21a we formulate the objective. Then, we develop the KL term and rewrite the entropy of the target distribution. The entropy can be omitted because it is constant with respect to the distribution q that we optimize. In Equation 21c we develop the evaluation of the density from the normalizing flow as defined in Equation 19a. Finally, in the final step, we approximate the expectation over the target density with the dataset of samples and Monte Carlo Integration.

As long as $f(\cdot)$ is a composition of invertible functions, $f(\cdot)$ is itself invertible. Therefore, researchers have worked in developing such layers and many of them can be stacked to form highly-flexible transformations. It is worth mentioning that a naive choice of $f(\cdot)$ would imply a complexity $\mathcal{O}(d^3)$ to compute the jacobian determinant. Therefore, researchers have developed constrained function such that the computation of the Jacobian determinant is computational efficient.

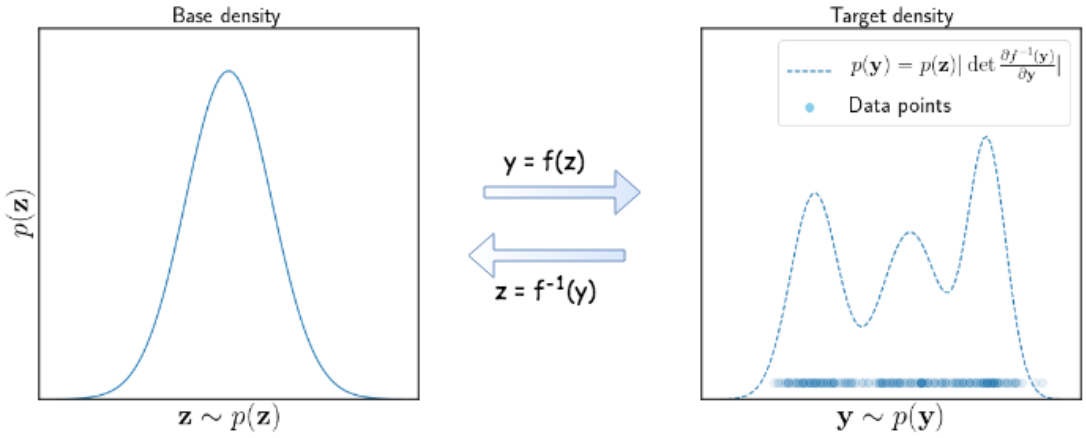


Figure 1: Representation of normalizing flows.

Real-Valued Non-Volume Preserving Transformations

Real NVPs (Dinh et al., 2017) are expressive transformations with a tractable Jacobian determinant. They use a common approach that is to design the transformation such that the Jacobian is triangular and therefore, the Jacobian determinant can be computed in $\mathcal{O}(d)$.

Given D dimensional variables $\mathbf{z}$ and $\mathbf{y}$ and $d < D$, the bijective mapping is defined as:

$$\begin{cases} \mathbf{y}_{1:d} = \mathbf{z}_{1:d} \\ \mathbf{y}_{d+1:D} = \mathbf{z}_{1:d} \odot \exp(s(\mathbf{z}_{1:d})) + t(\mathbf{z}_{1:d}) \end{cases} \tag{22a}$$

$$\Leftrightarrow \begin{cases} \mathbf{z}_{1:d} = \mathbf{y}_{1:d} \\ \mathbf{z}_{d+1:D} = (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d})) \odot \exp(-s(\mathbf{y}_{1:d})) \end{cases} \tag{22b}$$

where $\odot$ represents the element-wise product and the $\exp(\cdot)$ and addition operators are as well element-wise operators. The functions $s(\cdot)$ and $t(\cdot)$ are functions from $\mathbb{R}^d \to \mathbb{R}^{D-d}$ and can be arbitrarily complex as their Jacobians do not need to be computed to compute the Jacobian determinant of the transformation $f(.)$:

$$\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} = \begin{pmatrix} \mathbb{I}_d & 0 \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{z}_{1:d}} & \operatorname{diag}(\exp(s(\mathbf{z}_{1:d}))) \end{pmatrix} \tag{23a}$$

$$\Leftrightarrow \ln|\det \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}| = \sum_{j=0}^{D-d-1} s(\mathbf{z}_{1:d})_j \tag{23b}$$

and therefore, $s(\cdot)$ and $t(\cdot)$ are usually designed by trainable neural networks. Thus, the bijective mapping is entirely parametrized by the parameters θ of those neural networks but for clarity, we omit them in this section and note $f_\theta(\cdot)$ as $f(\cdot)$ and similarly with $f^{-1}(\cdot)$.

A similar reasoning allows to compute the log Jacobian determinant of the transformation $f^{-1}(\cdot)$ as required by Equation 19a:

$$\ln|\det \frac{\partial f^{-1}(\mathbf{y})}{\partial \mathbf{y}}| = -\sum_{j=0}^{D-d-1} s(\mathbf{y}_{1:d})_j. \tag{24}$$

Equation 24 can also be derived from the inverse function theorem:

$$\ln|\det \frac{\partial f^{-1}(\mathbf{y})}{\partial \mathbf{y}}| = \ln|\det \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}|^{-1} \tag{25a}$$

$$= -\sum_{j=0}^{D-d-1} s(\mathbf{z}_{1:d})_j \tag{25b}$$

$$= -\sum_{j=0}^{D-d-1} s(\mathbf{y}_{1:d})_j \tag{25c}$$

where the last equality comes from the fact that real NVP layers do not modify the first d components of their inputs. This of course introduces flexibility limitations and therefore, in practice variables are reversed between every layers so that there is no dimension that is left unchanged during the full transformation. In practise, d is often rounded to the closest integer to $\frac{D}{2}$.

Unconstrained Monotonic Neural Networks

Wehenkel and Louppe (2019) proposed another class of bijective mapping called Unconstrained Monotonic Neural Networks (UMNNs). These networks are based on the fact that a function is bijective as long as it is strictly monotonic or equivalently, as long as its derivative is of constant sign.

Therefore, they model a scalar bijective mapping as:

$$z = f^{-1}(y) \tag{26a}$$

$$= \int_0^y g_\phi(t)dt + \underbrace{f^{-1}(0)}_{\beta} \tag{26b}$$

where $g_\phi : \mathbb{R} \to \mathbb{R}_+$ is a strictly positive function — enforced by ELU activation increased by one — and $\beta \in \mathbb{R}$ is a scalar. The function g_ϕ can therefore be as complex as needed and its only constrained is to be positive. As with real NVPs, this function is therefore often defined by a neural network and the bijective mapping is entirely parametrized by ϕ and β. For clarity, we do not explicitly write the dependence of $f(\cdot)$ and $f^{-1}(\cdot)$ on these parameters. The integration is efficiently approximated (numerically) using Clenshaw-Curtis quadrature. UMNNs do not allow to compute $f : Z \to Y$ analytically and therefore, inversion should be done using root-finding algorithms.

The gradient with respect to y can be trivially computed:

$$\frac{d}{dy} f^{-1}(y) = g_\phi(y) \tag{27}$$

and the gradient with respect to the integrand parameters ϕ — required to train the network — can be computed with the Leibniz integral rule:

$$\nabla_\phi f^{-1}(y) = \int_0^y \nabla_\phi g_\phi(t)dt + \nabla_\phi \beta. \tag{28}$$

In order to scale to multi dimensions, UMNNs can be combined with autoregressive transformations that are widely used in normalizing flow architectures.

Similarly to real NVPs, the idea is to make the Jacobian of the transformation lower triangular. To do so, the transformation is made autoregressive such that $f^{-1}(\cdot)$ can be rewritten as a vector of d scalar functions:

$$f^{-1}(\mathbf{y}) = [f_1^{-1}(y_1), f_2^{-1}(\mathbf{y}_{1:2}), ..., f_d^{-1}(\mathbf{y}_{1:d})] \tag{29}$$

where each $f_i^{-1}(\cdot)$ is a scalar function. The Jacobian of the function $f^{-1}(\cdot)$ is indeed lower triangular and making each scalar function bijective is sufficient to make $f^{-1}(\cdot)$ bijective.

One can therefore build a normalizing flow architecture combining UMMNs and autoregressive transformations called UMNN autoregressive transformations (UMNN-MAF). Each layer in Equation 29 is represented by a UMNNs:

$$f_i^{-1}(\mathbf{x}_{1:i}) = \int_0^{x_i} g_{\phi^i}^i(t, h_{\psi^i}(\mathbf{x}_{1:i-1}))dt + \beta(h_{\psi^i}(\mathbf{x}_{1:i-1})) \tag{30}$$

where $h_{\psi^i}^i : \mathbb{R}^{i-1} \to \mathbb{R}^q$ is a q-dimensional neural embedding of the variables $\mathbf{x}_{1:i-1}$ and β is a scalar function. Equation 30 is obtained from the definition of UMNNs in equation 26 where the function $g_\phi(\cdot)$ now has additional arguments to account for the new variables $\mathbf{x}_{1:i-1}$ and similarly, the scalar β is transformed to a scalar function with arguments depending on $\mathbf{x}_{1:i-1}$. The bijective mapping is therefore parametrized by $\theta = \{\phi, \psi\}$.

To sum up, normalizing flows are powerful statistical models that allow to define a bijective mapping between a target and a base density. This allows to sample from the target distribution or evaluate its density. Each of these tasks have different complexity depending on the flow architecture and therefore, the appropriate model depends on the application at hand.

3. Related Work

Differentiable programming. The recent advances in deep learning and differentiable programming have leveraged the interest in (stochastic) gradient-based optimization. There is unfortunately a gap between scientific simulators and differentiable mappings between their inputs and outputs. Ullrich et al. (2019) filled this gap by implementing a differentiable alternative of a projection simulator in electron microscopy. However, simulators sometimes results of decades of engineering and cannot be rewritten easily. In these cases, experts rely on differentiable surrogates of the simulators. For example, Walker et al. (2019) model thousands of mouse's neuron responses by a convolutional neural network (CNN) that they latter use to find sensory stimuli that drive neurons optimally. In order to optimize the free-parameters of a simulator, Shirobokov et al. (2020) learn a differentiable surrogate during the optimization process in order to enable gradient-based optimization. Closely related to our approach, Lueckmann et al. (2018) learn a differentiable surrogate of the simulator that allows to evaluate the likelihood $p(\mathbf{y}|\mathbf{x})$. Then, given a prior distribution $p(\mathbf{x})$, one can do inference by evaluating $p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \propto p(\mathbf{x}|\mathbf{y})$. As opposed to them, we do not assume a known prior distribution. With an empirical Bayes approach, we learn a prior distribution during the inference process that can be directly used to quantify uncertainty over the parameters of interest or plugged together with the likelihood in a Markov Chain Monte Carlo (MCMC) sampler.

Likelihood-free inference. Likelihood-free inference study how to learn the likelihood ratio $r(\mathbf{y}_{obs}; \mathbf{x}_0, \mathbf{x}_1) = \frac{p(\mathbf{y}_{obs}|\mathbf{x}_0)}{p(\mathbf{y}_{obs}|\mathbf{x}_1)}$ or the posterior $p(\mathbf{x}|\mathbf{y_{obs}})$ given an observation $\mathbf{y_{obs}}$. As opposed to empirical Bayes that learns a distribution over the unseen data $\mathbf{x}$ given multiple observations, in the likelihood-free inference literature, $\mathbf{y}_{obs}$ is often a single observation.

The likelihood ratio is a powerful test statistics that allows to compute the goodness of a parameter against a reference parameter. For example, Brehmer et al. (2018) learn a likelihood ratio that can be used to constrain theory parameters in collider experiments. The likelihood ratio can also be used to solve a maximum likelihood estimation problem (Louppe, 2018). In our work, maximum likelihood estimation can be solved directly with the learned surrogate likelihood.

Since Rubin (1984), Approximate Bayesian computation (ABC) methods have been largely used to solve the problem of estimating the posterior in simulator-based inference. They run the simulator repeatedly with different parameters and only accept parameters that produce $\mathbf{y}'s$ close to the true observed data $\mathbf{y_{obs}}$. Then, the accepted $\mathbf{x}'s$ form an empirical posterior $p(\mathbf{x}|\mathbf{y_{obs}})$. Another well-known approach to approximate the posterior is to create model of the likelihood by kernel or histograms, usually in 1-D (Diggle and Gratton, 1984). Then, inference can be done with standard tools as if the likelihood was known. Both approaches suffer from the curse of dimensionality [Bishop (2006) & Bellman and Collection (1961)] and therefore rely on domain-knowledge summary statistics. Similarly to the latter approach, we build a model of the likelihood function. However, we take advantage of the revolutions in deep learning that allow to work with high-dimensional data and to keep the correlation between dimensions in the likelihood model.

With the recent advances in deep learning, sequential neural network-based methods have emerged requiring much fewer simulations than traditional ABC methods and much less domain knowledge. These methods are sequential in the sense that they alternate between an acquisition step guided by active learning where they run the simulator and an optimization step. Lueckmann et al. (2018) and Papamakarios et al. (2019) sequentially learn a density estimator of the simulator likelihood which allows once trained, to sample

from the posterior by MCMC. Hermans et al. (2019) learn a density ratio proportional to the likelihood which can be plugged into MCMC samplers while Papamakarios and Murray (2016), Lueckmann et al. (2018) and Greenberg et al. (2019) directly learn a density estimator of the posterior. While the density ratio method of Hermans et al. (2019) and the posterior-based method of Greenberg et al. (2019) had been compared and considered distinct in the literature, Durkan et al. (2020) recently showed that they are actually instances of a more general framework called contrastive learning. As opposed to sequential methods, our work assume that the simulator is costly to run (and therefore avoided at inference time) and that we do not have a known prior distribution. We maximize the marginal likelihood of observed data through the surrogate likelihood in order to learn a prior in the support allowed by the observed point. The learned prior and the likelihood can then be used to approximate or sample independent data from the posterior.

For a broader review of simulator-based inference, Cranmer et al. (2020) provide an overview of the rapidly evolving field as well as how machine learning revolutionizes it.

Empirical Bayes. Hong and Cheng (2018) detect careless or unmotivated responders in research surveys. A first step in their method is to consider the responders as random effects and learn the parameters of the underlying responder distribution. To do so, they use domain knowledge to build a complete-data log-likelihood function that they can plug in the well-known EM-algorithm (Dempster et al., 1977). Wu et al. (2018) and Krishnan et al. (2019) learn a prior over the weights of Bayesian Neural Networks by approximating empirical Bayes (Robbins, 1956) to allow variational inference. In our work, we build a hierarchical model where the observations are independent on the population parameters given the latent variables $\mathbf{x}$. Given a surrogate that allows to evaluate an approximation of the likelihood function, this allows to learn a neural network-based prior distribution of the underlying population parameters by maximum marginal likelihood, with minimal assumptions on the form of the prior.

Closely related to our work is Louppe and Cranmer (2019) that deal with the fact that the simulator is non-differentiable by learning a prior distribution over the unseen variables such that the marginal likelihood of reconstructed data — defined by the simulator and the trainable prior — match the empirical distribution of observed data in a minimax problem. This can be viewed as a form of empirical Bayes where the prior is optimized based on the observed data. In our work, we build an estimator of the likelihood which allows to directly maximize the marginal likelihood and thus, we do not need the simulator at inference.

4. Surrogate Models

In this section we describe how we use normalizing flows to build a stochastic model of the simulator from a dataset $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^N$ of pairs of uncorrupted and corrupted data.

Given such a dataset and that the end goal is to do inference, rather than learning a model of the simulator — i.e. a stochastic mapping from $\mathbf{x}$ to $\mathbf{y}$ — one could be interested in directly learning a stochastic mapping from $\mathbf{y}$ to $\mathbf{x}$ (Kruse et al., 2019; Ardizzone et al., 2019). Yet, for many simulators and experiment setups, this might not be the best way to tackle the problem.

First, the stochastic mapping from $\mathbf{x}$ to $\mathbf{y}$ results from a true underlying process and given a flexible model enough, can be approximated precisely well. On the other hand, the stochastic mapping from $\mathbf{y}$ to $\mathbf{x}$ is ambiguous and is usually much more difficult to learn.

Second, suppose that the uncorrupted data in $\mathcal{D}$ have been generated from a proposal prior $\tilde{p}(\mathbf{x})$. Then, if someone learns a stochastic mapping from $\mathbf{y}$ to $\mathbf{x}$ such that the mapping approximates the posterior, the learned distribution is:

$$\tilde{p}(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})}{\tilde{p}(\mathbf{y})} \tag{31}$$

where $\tilde{p}(\mathbf{y})$ is the evidence $\int p(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})\mathbf{dx}$.
Suppose now that we want to do inference on data drawn from another prior $p(\mathbf{x})$ — for example a tighter prior or a prior concentrated in a region of interest. Then, the target posterior is:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} \tag{32}$$

where $p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\mathbf{dx}$. So, the stochastic mapping learned beforehand does not approximate the target posterior but rather approximates a proposal posterior and the two are linked by:

$$\tilde{p}(\mathbf{x}|\mathbf{y}) = p(\mathbf{x}|\mathbf{y})\frac{\tilde{p}(\mathbf{x})p(\mathbf{y})}{p(\mathbf{x})\tilde{p(\mathbf{y})}} \tag{33}$$

where the last equality is obtained by combining equations 31 and 32.

This kind of situation is not uncommon in practise. For example, one can be interested in doing inference on a stochastic detector whose inputs are themselves results of simulations. If the input data generator is tuned over time or run with other arguments producing data with a different distribution, a previously trained model of the posterior distribution of the detector would not approximate the target distribution.

On the other hand, directly learning a mapping from $\mathbf{x}$ to $\mathbf{y}$ — learning the likelihood — does not suffer from this issue (Papamakarios et al., 2019). Indeed, for large N, training a density estimator $q_\theta(\mathbf{y}|\mathbf{x})$, approximation of $p(\mathbf{y}|\mathbf{x})$ by maximizing the total log likelihood $\sum_{n=1}^N \log q_\phi(\mathbf{y}_n|\mathbf{x}_n)$ — where the data have been drawn from $\tilde{p}(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})$ — is approximately equivalent to maximizing:

$$\mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})} \log q_\phi(\mathbf{y}|\mathbf{x}) = \mathbb{E}_{\tilde{p}(\mathbf{x})}\mathbb{E}_{p(\mathbf{y}|\mathbf{x})} \log q_\phi(\mathbf{y}|\mathbf{x}) \tag{34a}$$

$$= \mathbb{E}_{\tilde{p}(\mathbf{x})} \int \log q_\phi(\mathbf{y}|\mathbf{x}) p(\mathbf{y}|\mathbf{x}) \mathbf{dy} \tag{34b}$$

$$= \mathbb{E}_{\tilde{p}(\mathbf{x})} \int (\log q_\phi(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{y}|\mathbf{x})) p(\mathbf{y}|\mathbf{x}) \mathbf{dy} \tag{34c}$$

$$= \mathbb{E}_{\tilde{p}(\mathbf{x})} \left[\int (\log q_\phi(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{y}|\mathbf{x})) p(\mathbf{y}|\mathbf{x}) \mathbf{dy} + H(p) \right] \tag{34d}$$

$$= -\mathbb{E}_{\tilde{p}(\mathbf{x})} \left[KL(p(\mathbf{y}|\mathbf{x}) || q_\phi(\mathbf{y}|\mathbf{x})) \right] + \cancel{H(p)}. \tag{34e}$$

In the first line we factorize the joint distribution. Then, in the next line we expand the expectation definition and in Equation 34c we introduce the log-likelihood which will later allow us to write the Kullback–Leibler divergence. In Equation 34d we develop the integrand and rewrite the entropy of the likelihood distribution. Finally, in the last line we rewrite the integral using the KL divergence definition and develop the expectation over $\tilde{p}(\mathbf{x})$ taking into account that $H(p)$ is a constant with respect to $\tilde{p}(\mathbf{x})$.

The above quantity is maximized when $q_\phi(\mathbf{x}|\mathbf{y}) = p(\mathbf{x}|\mathbf{y})$ in the support of $\tilde{p}(\mathbf{x})$. This means that as long as the proposal prior $\tilde{p}(\mathbf{x})$ do not exclude parts of the prior space, no matter the shape of the proposal, the learned distribution approximates the true likelihood everywhere. This suggests that one could build a dataset with a broad proposal of any shape and use it to train a surrogate to do inference in different subdomains. Thus, after an upfront expensive simulation phase, inference is amortized as the surrogate can be used as a proxy of the simulator everywhere in the support of the proposal prior.

Now, let us focus on how we use normalizing flows to build a model of the simulator or in other words, learn a stochastic mapping from $\mathbf{x}$ to $\mathbf{y}$ maximizing Equation 34. Note that the latter equation and the objective introduced to train normalizing flows (Equation 21b) are the same up to the conditioning variables $\mathbf{x}$.

Real NVP layers can be adapted to produce conditional layers. As the scale network $s(.)$ and translation network $t(.)$ can be any function in $\mathbb{R}^d$, we concatenate the conditioning data $\mathbf{x}$ to their inputs. This allows to define a bijective mapping $f(.;\mathbf{x})$ between $\mathbf{z}$ and $\mathbf{y}$ conditioned on $\mathbf{x}$:

$$\begin{cases} \mathbf{y}_{1:d} = \mathbf{z}_{1:d} \\ \mathbf{y}_{d+1:D} = \mathbf{z}_{1:d} \odot \exp(s(\mathbf{z}_{1:d};\mathbf{x})) + t(\mathbf{z}_{1:d};\mathbf{x}) \end{cases} \tag{35a}$$

$$\Leftrightarrow \begin{cases} \mathbf{z}_{1:d} = \mathbf{y}_{1:d} \\ \mathbf{z}_{d+1:D} = (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d};\mathbf{x})) \odot \exp(-s(\mathbf{y}_{1:d};\mathbf{x})). \end{cases} \tag{35b}$$

This allows to sample from $p(\mathbf{y}|\mathbf{x})$ or evaluate its density (Figure 2). It is also possible to condition the base density $p(\mathbf{z})$ on $\mathbf{x}$ even though it is not a common practice.

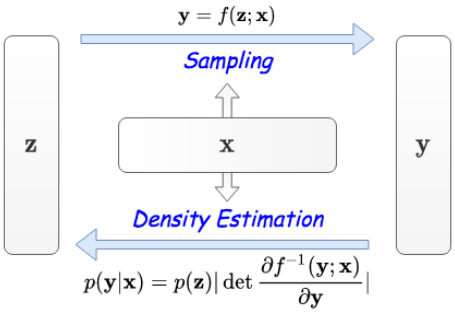


Figure 2: Representation of conditional normalizing flows.

UMNN-MAF layers can also be adapted to produce conditioning layers. This is done by concatenating the conditioning variable $\mathbf{x}$ to the inputs of the embedding network from Equation 30:

$$f_i^{-1}(\mathbf{y}_{1:i}) = \int_0^{y_i} g^i_{\phi^i}(t, h_{\psi^i}(\mathbf{y}_{1:i-1}; \mathbf{x}))dt + \beta(h_{\psi^i}(\mathbf{y}_{1:i-1}; \mathbf{x})). \tag{36}$$

In the rest of this dissertation, we assume that we have a trained surrogate model of the simulator. Therefore, we assume that we can evaluate the likelihood $p(\mathbf{y}|\mathbf{x})$ (or at least, approximate it) and sample from the surrogate. We note the surrogate likelihood $\tilde{p}(\mathbf{y}|\mathbf{x})$ but for clarity, in the rest of this dissertation, when it is clear that the likelihood is evaluated by the surrogate, we simply write $p(\mathbf{y}|\mathbf{x})$. We use the notation $S(\cdot; \mathbf{x})$ for the function $S : Z \to Y$ that approximates a stochastic simulation with parameters $\mathbf{x}$.

Interestingly, the dimensions of the variables that we are doing inference on and the variables that we observe do not need to be the same which is a powerful asset. For example, $\mathbf{x}$ can be a high resolution image that we perform optimization on while $\mathbf{y}$ can be its observed low-resolution counterpart. Or $\mathbf{y}$ can be a grey image that we are trying to color by optimizing $\mathbf{x}$, a three channel image. Among others, this is specifically interesting in scientific imaging where we are trying to retrieve a $\mathbb{R}^{d+1}$ structure from $\mathbb{R}^d$ observations, such as a 3-D protein structure from 2-D microscope observations (Ullrich et al., 2019).

5. Inference

This section is divided in four parts. In Part 5.1, we focus on point estimation and then, in Part 5.2 we introduce uncertainty quantification. In Part 5.3, we switch to multiple observations and learn a prior distribution over the unseen parameters. Finally, in Part 5.4, given multiple observations, we first learn a prior as described in Part 5.3 and then, we use it with the surrogate likelihood to do inference on each observed variable.

The main novelty of our work rely on how we use the surrogate in the optimization processes. We take advantage of normalizing flow surrogates that allows us to evaluate an approximation of the simulator likelihood. We then make an implicit model assumption. We assume that given parameters $\mathbf{x}$, the observed data $\mathbf{y}$ is independent on the random variable θ, the hyperparameters of the distribution over $\mathbf{x}$. This is depicted in Figure 3. We do not learn a distribution on θ but we rather try to find its most likely estimate as it is commonly done in empirical Bayes. Then, given this assumption, we combine generative modelling, Monte Carlo integration, the reparameterization trick, and standard statistics methods to design powerful inference algorithms.

Figure 3: (**Left**) Bayesian model with global variables θ. (**Right**) Hierarchical model where $\mathbf{y}$ is independent on θ given $\mathbf{x}$.

In Part 5.1, we simply highlight that the surrogate can be used directly to solve a maximum likelihood estimation (MLE) or constrained MLE problem.

In Part 5.2, we introduced four methods that go beyond point estimation and learn a distribution over the unseen variables. Given our assumptions, they require a differentiable surrogate of the simulator that allows to evaluate the likelihood — the last requirement is not needed for one of the approach. They do not need the true simulator nor domain knowledge at inference. Apart from one method, they train a model of the distribution of interest whose only constraint is that it should be sampled from efficiently. They do not need to evaluate its density during training. This suggest computationally efficient methods that should be applicable on high-dimensional problems as they do not require to compute the log Jacobian determinant at each training step.

In Part 2.2, we generalize one of the method proposed in the previous point to multiple observations.

In Part 5.4, we highlight how four standard inference methods can be used to learn a posterior for a given observation given the surrogate likelihood and a prior learned from multiple observations (Part 2.2). The inference process is fast and amortized as after an upfront expensive data acquisition and surrogate training phase, a prior is learned only once and then, these quantities are fixed and can be used to do inference.

5.1 Point Estimation

As highlighted in section 2.1, a common approach to solve inverse problems is to solve the maximum a posteriori (MAP) estimation problem. This requires the likelihood function and a prior distribution. In our experiments, we study significantly different simulators for which we build a surrogate that allows to evaluate the likelihood but we assume few prior knowledge and as opposed to well-studied problem, we do not have access to off-the-shelf regularizers.

We will therefore focus on maximum likelihood estimation (MLE) which is also a common approach to solve inverse problems. Note that solving the MLE problem is substantially different than solving the MAP estimation problem. In the first case, the question answered is: *For which* $\mathbf{x}$*, the observation* $\mathbf{y}$ *is the most likely to be produced?* while in the latter case, the question is rather: *Given this observation, which* $\mathbf{x}$ *is the most likely to have produced it?* (Renaud Foy, 2002).

Maximum Likelihood Estimation

Experimental results show that even if the surrogate likelihoods are in general non convex and that no regularizers are used, we are able to properly retrieve plausible $\mathbf{x}'s$ that may have generated observed $\mathbf{y}'s$. To do so, we optimize by gradient ascent:

$$\mathbf{x_{t+1}} = \mathbf{x}_t + \gamma \nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}_t) \tag{37}$$

where γ is a learning rate and $\mathbf{x}_0$ is randomly initialized.

Constrained Maximum Likelihood Estimation

Despite good empirical results when solving the maximum likelihood problem, MLE alone is often unable to properly solve inverse problems. Therefore, a common approach is to rather solve the constrained maximum likelihood estimation problem:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) \quad s.t. \quad \mathbf{x} \in S \tag{38}$$

where S is a set that captures prior knowledge about $\mathbf{x}$, such as positiveness.

Constrained MLE is not a too strong assumption. For example, in science and technology it is common to have variables that are naturally constrained. Namely, we know that the mass of a particle should be positive or that pixel values are bounded by the application dynamic range.

To ensure such constraints, we use the following reparametrization:

$$\mathbf{x} = h(\mathbf{z}) \tag{39}$$

where $h(\cdot)$ is a constrained function (possibly parametrized) and $\mathbf{z}$ is the new optimized variable. The constrained problem 38 therefore becomes:

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} \log p(\mathbf{y}|h(\mathbf{z})) \tag{40a}$$

$$\mathbf{x}^* = h(\mathbf{z}^*). \tag{40b}$$

Note that $h(\cdot)$ should be differentiable to allow backpropagation. For instance, to ensure positiveness, $h(\cdot)$ can be the rectified linear unit. If we want to constraint $\mathbf{x}$ such that

$\mathbf{x} \in [\alpha, \beta]$, we can design $h(\cdot)$ such that:

$$\mathbf{x} = h(\mathbf{z}) \tag{41a}$$
$$= (\alpha - \beta)\text{Sigmoid}(\mathbf{z}) + \beta \tag{41b}$$

In image reconstruction, Ulyanov et al. (2017) interestingly show that when the function $h(\cdot)$ is a deep convolutional network, the network itself acts as a regularizer that constrains $\mathbf{x}$ in the space of natural-like images.

When the prior is uniform and that the constrained function ensure that the solution lies within the prior space, solving the MLE problem is equivalent to solving the MAP estimation problem. At the same time, anyone who is solving the problem defined in Equation 5 with a mean-squared error loss:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} ||\mathbf{A}\mathbf{x} - \mathbf{y}||^2, \quad s.t \quad \mathbf{x} \in \mathcal{S} \tag{42}$$

is implicitly solving a constrained maximum likelihood problem — or a maximum likelihood problem if the solution is not constrained. Therefore, when using a mean-squared error, constrained maximum likelihood is a generalization of Equation 5 and is well justified. This is shown in Appendix A.

Empirical Prior From Single Estimates

Now, rather than having a single observation, let us assume that we have observed a large number of events $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$. One's goal could be to form estimates $\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_N$ of the $\mathbf{x}'s$ that have generated the observations or to learn a distribution over the population $\mathbf{x}$.

A first naive approach would be to learn for each observation $\mathbf{y}_i$ its most likely estimate $\hat{\mathbf{x}}_i$ such that $p(\mathbf{y}_i|\hat{\mathbf{x}}_i)$ is maximized. Efron (2019) shows that one can effectively do better by considering the observations as a whole rather than treating them independently. We will discuss this problem in Section 5.3.

We experimentally show that given a large number of observations, we can learn individual estimates $\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_N$ of the true uncorrupted data and that these estimates can be binned to form a plausible empirical prior distribution over $\mathbf{x}$. This process can be efficiently parallelizable and is fast. Therefore, if someone is interesting in learning a distribution over the unseen variables $\mathbf{x}$, the proposed method can be used to gather prior knowledge before a more complex optimization loop.

5.2 Density Estimation

Given a single observation $\mathbf{y}_{obs}$ we aim at finding a distribution over the latent variables $\mathbf{x}$ that may have produced this event. As we can approximate the likelihood through the learned surrogate, if we had a prior over $\mathbf{x}$, we could sample from the posterior with Markov Chain Monte Carlo (MCMC) or solve a variational inference problem to learn the posterior. Therefore, in this section we will not operate in a fully Bayesian setting but rather introduce four methods that can learn a distribution over $\mathbf{x}$ without a known prior distribution. First, we show how we use Empirical Bayes to learn a prior distribution over the latent variables in the support allowed by the observation. Then, we show how to constrain this distribution such that it stays close to the posterior. Third we show how the EM algorithm an be used given our assumptions. Finally, we show how to learn a distribution such that it generates parameters whose simulator's outputs are close to the observed data given a defined loss.

N=1 Empirical Bayes

As postulated in Section 2.2, empirical Bayes aims at finding population parameters that maximize the marginal likelihood. As opposed to equation Equation 8, we do not operate with a dataset of observations $\mathbf{Y}$ but with a single observation $\mathbf{y}_{obs}$:

$$p(\mathbf{y}_{obs}|\theta) = \int p(\mathbf{x}, \mathbf{y}_{obs}|\theta)\mathbf{dx}. \tag{43}$$

As already mentioned, in order to maximize Equation 43, we build a hierarchical model and assume that $\mathbf{y}$ is independent on θ given $\mathbf{x}$ (figure 3). Then, as opposed to a fully Bayesian analysis, we do not try to learn a distribution on θ but we find its most likely values as it is commonly done in empirical Bayes. So, Equation 43 is maximized directly with respect to θ rather than being integrated (again with respect to θ). The optimization problem is therefore:

$$\theta^* = \arg\max_\theta \log p(\mathbf{y}_{obs}|\theta) \tag{44a}$$

$$= \arg\max_\theta \log \int p(\mathbf{y}_{obs}|\mathbf{x}, \theta)p(\mathbf{x}|\theta)\mathbf{dx} \tag{44b}$$

$$= \arg\max_\theta \log \int p(\mathbf{y}_{obs}|\mathbf{x})p(\mathbf{x}|\theta)\mathbf{dx}. \tag{44c}$$

The first line maximizes the marginal likelihood with respect to the prior hyperparameters. Then, the second line comes from marginalizing $\mathbf{x}$ and rewriting the joint probability distribution using the definition of conditional probabilities. Finally, the last line uses the hierarchical model assumption.

Our aim is to model $p(\mathbf{x}|\theta)$ by a neural-density estimator $q_\theta(\mathbf{x})$ — a normalizing flow — and optimize θ, the parameters of the normalizing flow. The flow should allow us to differentially sample $\mathbf{x}'s$ from a base density $p(\boldsymbol{\epsilon})$ as $\mathbf{x} = f(\boldsymbol{\epsilon})$, $\epsilon \sim p(\boldsymbol{\epsilon})$ and to approximate the density $p(\mathbf{x}|\theta)$ by $q_\theta(\mathbf{x})$. The first condition is a strong requirement that should be met in order to train the model as will be explained below. However, during the optimization process, it is not required to evaluate the density $q_\theta(\mathbf{x})$ and therefore, if the end-goal is only to learn a distribution that can be sampled from but not evaluated, any generative model that allows to differentially sample $\mathbf{x}$ from a base density can be used. We train the flow by setting its parameters θ to the most likely values of Equation 44. We therefore aim to solve by gradient ascent:

$$\theta^* = \arg\max_\theta \mathcal{L}(\theta) \tag{45}$$

with $\mathcal{L}(\theta)$:

$$\mathcal{L}(\theta) = \log \int p(\mathbf{y}_{obs}|\mathbf{x})q_\theta(\mathbf{x})\mathbf{dx} \tag{46a}$$

$$= \log \mathbb{E}_{q_\theta(\mathbf{x})} p(\mathbf{y}_{obs}|\mathbf{x}) \tag{46b}$$

$$= \log \mathbb{E}_{p(\boldsymbol{\epsilon})} p(\mathbf{y}_{obs}|f_\theta(\boldsymbol{\epsilon})) \tag{46c}$$

$$\approx \log \frac{1}{N} \sum_{\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})} p(\mathbf{y}_{obs}|f_\theta(\epsilon)) \tag{46d}$$

The first line results from Equation 44 where the prior has been approximated by the density estimator $q_\theta(x)$. In the second line we rewrite the integral as an expectation and in the

third line we apply the law of the unconscious statistician (LOTUS). Finally, the last line approximates the expectation (46c) with Monte Carlo integration.

Maximizing Equation 46 by gradient ascent requires the gradient signal from the surrogate model. There is a whole literature about Monte Carlo gradient estimation and for example, one can compute the gradient of $\mathcal{L}(\theta)$ without having to backpropagate through the surrogate — called a score function gradient estimator (Mohamed et al., 2019). There are many competing approaches in Monte Carlo gradient estimation and they mainly differ in their bias and variance properties as well as their computational requirements. Using such estimators may allow us to have less noisy gradients but is left for future work.

In our experiments, we applied the log-sum-exp trick to $\mathcal{L}(\theta)$ and used PyTorch (Paszke et al., 2019) to compute gradients by automatic differentiation. The log-sum-exp is a convex function (Boyd and Vandenberghe, 2004) that allows to compute a sum of exponentials in a numerical stable way and is defined as:

$$\begin{aligned}\text{logSumExp}(x_1, ..., x_n) &= \log\left[\exp(x_1) + ... + \exp(x_n)\right] && \text{(47a)}\\ &= x^* + \log\left[\exp(x_1 - x^*) + ... + \exp(x_n - x^*)\right] && \text{(47b)}\end{aligned}$$

where x^* is the maximum value in $\{x_1, ..., x_n\}$. Intuitively, when x^* is substantially higher than the other variables, it contributes to most of the value of Equation 47 and the value of the equation is close to the value of x^*. Therefore, rewriting the equation allows to avoid numerical error on this most interesting variable and numerical errors in the second part of Equation 47b are less important. Or for example, when all variables are of constant sign and highly positive or highly negative, subtracting them by x^* allows to shift all the variables towards zero which allows more stable computations.

Therefore, Equation 46d is computed as:

$$\begin{aligned}\log\frac{1}{N}\sum_{\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})} p(\mathbf{y}_{obs}|f_\theta(\epsilon)) &= \log\sum_{\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})} p(\mathbf{y}_{obs}|f_\theta(\epsilon)) - \log N && \text{(48a)}\\ &= \log\sum_{\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})} \exp\log p(\mathbf{y}_{obs}|f_\theta(\epsilon)) - \log N && \text{(48b)}\\ &= \text{logSumExp}_{\boldsymbol{\epsilon}}\log p(\mathbf{y}_{obs}|f_\theta(\epsilon)) - \cancel{\log N} && \text{(48c)}\end{aligned}$$

where the first line comes from applying the logarithm product and power rules, the second line rewrites the likelihood using an identity operator that will allow us to write the log-sum-exp trick. Finally, in the last line, we rewrite the equation using the notation logSumExp to refer to the log-sum-exp trick operator. As $\log N$ is constant with respect to θ, it can be omitted when doing gradient optimization.

Before concluding with this method, let us focus on the defined loss. Given Equation 46b, the objective is maximized when the distribution $q_\theta(\mathbf{x})$ is a sum of dirac functions where the likelihood is maximum. Or in other words, when the learned distribution only generates points where the likelihood is maximum. When the likelihood function has finite number of global maximum, this means that the problem is optimized when the learned distribution degenerates to the most likely estimates. This is not specifically a bad behaviour and depends on what is expected. For example it can be an alternative to maximum likelihood estimation in order to retrieve different modes. There exists problems, where there are many equally likely $\mathbf{x}'s$ that could have generated an observation, and therefore, the method proposed here allows to learn a distribution over the unseen parameters. It is

worth mentioning that the generative model used to model the distribution may act as a regularizer and avoid mode collapse even when there is a single most likely estimate.

Expected Log Likelihood

As explained, the hereabove defined method tends to collapse to the most likely estimates of the problem - when there are finite. We show that it is possible to avoid such behaviour for example by constraining the distribution $q_\theta(\cdot)$ by maximizing:

$$\mathcal{L}(\theta) = \mathbb{E}_{q_\theta(\mathbf{x})} \log p(\mathbf{y}_{obs}|\mathbf{x}). \tag{49}$$

As the logarithm is a concave function, losses defined in Equation 46 and Equation 49 are linked by Jensen's inequality:

$$\mathbb{E}_{q_\theta(\mathbf{x})} \log p(\mathbf{y}_{obs}|\mathbf{x}) \leq \log \mathbb{E}_{q_\theta(\mathbf{x})} p(\mathbf{y}_{obs}|\mathbf{x}). \tag{50}$$

So, maximizing the expected log likelihood with respect to the variational prior distribution is maximizing a lower bound on the maximum marginal likelihood loss. Let us study the interval between these two quantities. By the Bayes rule, we know that:

$$\log p(\mathbf{y}_{obs}|\theta) = \log p(\mathbf{y}_{obs}|\mathbf{x}) + \log q_\theta(\mathbf{x}) - \log \tilde{p}(\mathbf{x}|\mathbf{y}_{obs}, \theta) \tag{51}$$

where $p(\mathbf{y}_{obs}|\theta) = \int p(\mathbf{y}_{obs}|\mathbf{x}) q_\theta(x) \mathbf{dx}$, $\tilde{p}(\mathbf{x}|\mathbf{y}_{obs}, \theta) = \frac{p(\mathbf{y}_{obs}|\mathbf{x}) q_\theta(\mathbf{x})}{p(\mathbf{y}_{obs}|\theta)}$ and the dependence of θ in the likelihood function has been removed due to the hierarchical model assumption. If we take the expectation over $q_\theta(\mathbf{x})$ on each side of Equation 51:

$$\log p(\mathbf{y}_{obs}|\theta) = \mathbb{E}_{q_\theta(\mathbf{x})} \log p(\mathbf{y}_{obs}|\mathbf{x}) + \mathrm{KL}(q_\theta(x)||\tilde{p}(\mathbf{x}|\mathbf{y}_{obs}, \theta)) \tag{52}$$

or equivalently:

$$\mathbb{E}_{q_\theta(\mathbf{x})} \log p(\mathbf{y}_{obs}|\mathbf{x}) = \log p(\mathbf{y}_{obs}|\theta) - \mathrm{KL}(q_\theta(x)||\tilde{p}(\mathbf{x}|\mathbf{y}_{obs}, \theta)). \tag{53}$$

In other words, the difference between the two losses is the Kullback–Leibler divergence between the variational prior and the posterior distribution. The first term of the loss defined in Equation 49 is the maximum marginal likelihood and can be compared to a reconstruction term that encourages q to put its mass where the likelihood is maximized. The second term is the negative Kullback–Leibler divergence between the learned distribution and the posterior which constrains the solution to remain close to the posterior. This allows to learn broader distributions over $\mathbf{x}$ and was confirmed in experimental studies.

EM algorithm

Given the hierarchical model assumption and taking advantage of the fact that we can compute the likelihood, we can as well use the EM algorithm. Under the hierarchical model

assumption, the expectation step defined in Equation 10 can be rewritten as:

$$
\begin{aligned}
Q(\theta, \theta^{(i-1)}) &= \int \log\left[p(\mathbf{X}, \mathbf{Y}|\theta)\right] p(\mathbf{X}|\mathbf{Y}, \theta^{i-1}) \mathbf{dX} && \text{(54a)} \\
&= \int \log\left[p(\mathbf{x}, \mathbf{y}_{obs}|\theta)\right] p(\mathbf{x}|\mathbf{y}_{obs}, \theta^{i-1}) \mathbf{dx} && \text{(54b)} \\
&= \int \log\left[p(\mathbf{x}, \mathbf{y}_{obs}|\theta)\right] \frac{p(\mathbf{y}_{obs}|\mathbf{x}) q_{\theta^{i-1}}(\mathbf{x})}{p(\mathbf{y}_{obs}|\theta^{i-1})} \mathbf{dx} && \text{(54c)} \\
&\propto \int \log\left[p(\mathbf{x}, \mathbf{y}_{obs}|\theta)\right] p(\mathbf{y}_{obs}|\mathbf{x}) q_{\theta^{i-1}}(\mathbf{x}) \mathbf{dx} && \text{(54d)} \\
&= \mathbb{E}_{q_{\theta^{i-1}}(\mathbf{x})} \log\left[p(\mathbf{x}, \mathbf{y}_{obs}|\theta)\right] p(\mathbf{y}_{obs}|\mathbf{x}) && \text{(54e)} \\
&= \mathbb{E}_{q_{\theta^{i-1}}(\mathbf{x})} \log\left[p(\mathbf{y}_{obs}|\mathbf{x}) q_{\theta}(\mathbf{x})\right] p(\mathbf{y}_{obs}|\mathbf{x}) && \text{(54f)}
\end{aligned}
$$

In the first line, we use the definition of $Q(\theta, \theta^{(i-1)})$. In the second line, we highlight that there is a single observation $\mathbf{y}_{obs}$. Then, in the third line we use the Bayes rule, model the prior distribution by a variational prior and and use the hierarchical model assumption in order to remove the dependency on θ in the likelihood term. In Equation 54d, as the purpose of the expectation step is to define a function that will be optimized with respect to θ, the evidence can be removed as it is positive and constant with respect to $\mathbf{x}$ and θ. Then, in Equation 54e we rewrite the integral as an expectation and finally, in the last line, we rewrite the joint distribution using the hierarchical model assumption.

The maximization step is unchanged and aims at finding θ that maximizes Equation 54. We search this value by gradient ascent.

Concretely, in the expectation step, we generate a dataset $\mathcal{D}$ of latent variables from the current estimate of $q_{\theta^{i-1}}$. Then, in the maximization step, we optimize $Q(\theta, \theta^{i-1})$ by gradient ascent:

$$
\begin{aligned}
\theta^i &= \theta^{i-1} + \gamma \nabla_\theta Q(\theta, \theta^{i-1}) && \text{(55a)} \\
&= \theta^{i-1} + \gamma \sum_{\mathbf{x} \in \mathcal{D}} \nabla_\theta \log(p(\mathbf{y}_{obs}|\mathbf{x}) q_\theta(\mathbf{x})) p(\mathbf{y}_{obs}|\mathbf{x}) && \text{(55b)}
\end{aligned}
$$

where γ is a learning rate. This procedure is depicted in Algorithm 1. The algorithm follows the same structure as sequential neural network-based algorithms in likelihood-free inference. That is, it is sequentially composed of step that generates a dataset and then, the parameters of the density estimator are optimized with respect to the generated dataset. In our case, we only generate a dataset of uncorrupted data while sequential methods generate pairs of uncorrupted and corrupted data by running the simulator.

In the EM algorithm, each iteration is guaranteed to increase the marginal likelihood and thus, the algorithm is guaranteed to converge to a local maximum. However, we approximate the maximization step by gradient descent rather than computing it analytically as it is commonly done. Therefore, there is no guarantees that each iteration will indeed increase the marginal likelihood. Nonetheless, experimental results show a clear increase between each step before convergence.

Algorithm 1 EM algorithm

Inputs: Surrogate model $p(\mathbf{y}|\mathbf{x})$, Density estimator $q_\theta(\mathbf{x})$, Observation $\mathbf{y}_{obs}$, Number of iterations N, Number of points to estimate the expectation M

Outputs: Trained density estimator $q_\theta(\mathbf{x})$

for $n = 1 : N$ **do**
  $\mathcal{D} = \{\}$
  sample $\mathbf{x}_{1:M} \sim q_\theta(\mathbf{x})$
  add $\mathbf{x}_{1:M}$ into $\mathcal{D}$
  while not converged
    $\mathcal{L}(\theta) \leftarrow -\sum_{\mathbf{x}\in\mathcal{D}} \log(p(\mathbf{y}_{obs}|\mathbf{x})q_\theta(\mathbf{x}))p(\mathbf{y}_{obs}|\mathbf{x})$
    $\theta \leftarrow$ **Optimizer**$(\theta, \mathcal{L}(\theta))$
  end while
end for

Handcrafted Loss

Finally, we can use a handcrafted loss. Here, the idea is to learn a distribution that generates data such that when corrupted, there are close to the observation given a defined distance metric:

$$\mathcal{L}(\theta) = \mathbb{E}_{p_\theta(\mathbf{z},\mathbf{x})}\mathcal{C}(\mathbf{y}_{obs}, \mathcal{S}(\mathbf{z}, \mathbf{x})) \tag{56a}$$
$$= \mathbb{E}_{p(\mathbf{z})}\mathbb{E}_{q_\theta(\mathbf{x})}\mathcal{C}(\mathbf{y}_{obs}, \mathcal{S}(\mathbf{z}, \mathbf{x})) \tag{56b}$$
$$= \mathbb{E}_{p(\mathbf{z})}\mathbb{E}_{p(\boldsymbol{\epsilon})}\mathcal{C}(\mathbf{y}_{obs}, \mathcal{S}(\mathbf{z}, f_\theta(\boldsymbol{\epsilon}))) \tag{56c}$$

where $\mathcal{C}$ is an appropriate cost function between the observation and the regenerated data such as the least-squared error. The first line defines the loss as the expectation of a cost function. The expectation is taken over the joint of the variables that drive the stochasticity of the surrogate simulator and the distribution over $\mathbf{x}$ that we try to lean. The second lines comes from the independence of these variables and we apply the law of the unconscious statisticians in the last line.

This loss can be useful if we have a differentiable surrogate or directly a differentiable simulator but that we cannot evaluate the likelihood. However, this is substantially different than solving the three other problems as we will rather find a distribution over $\mathbf{x}$ that maximizes a cost function between regenerated $\mathbf{y}'s$ and the observation and therefore, we will not learn a prior distribution.

5.3 Empirical Bayes

In this section, we assume that we have observed a large number of i.i.d. events $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$ and we are interested in learning an underlying population distribution over $\mathbf{x}$. Therefore, as opposed to Section 5.1, we do not try to learn the hidden variables $\mathbf{x}_i$ for each observation. We focus on the later problem in Section 5.4 where the distribution learned in this section allows to amortize the inference process.

Similarly to Section 5.2, we model the distribution over $\mathbf{x}$, $p(\mathbf{x}|\theta)$ by a neural-density estimator $q_\theta(\mathbf{x})$ and optimize its parameters θ. The density estimator should allow us to

differentially sample $\mathbf{x}'s$ from a base density $p(\boldsymbol{\epsilon})$ as $\mathbf{x} = f(\boldsymbol{\epsilon})$, $\epsilon \sim p(\boldsymbol{\epsilon})$ and to approximate the density $p(\mathbf{x}|\theta)$ by $q_\theta(\mathbf{x})$ but only the former condition is required by the optimization process. So, the distribution over $\mathbf{x}$ could be technically modeled by any differentiable generative model.

In order to learn the parameters θ, we maximize the marginal likelihood of observed data with the hierarchical model assumption made in Section 5.2:

$$\max_\theta p(\mathbf{Y}|\theta) = \max_\theta \prod_j^N p(\mathbf{y}_j|\theta) \tag{57a}$$

$$= \max_\theta \prod_j^N \int p(\mathbf{y}_j|\mathbf{x}) q_\theta(\mathbf{x}) \mathbf{dx} \tag{57b}$$

$$= \max_\theta \prod_j^N \mathbb{E}_{q_\theta(\mathbf{x})} p(\mathbf{y}_j|\mathbf{x}) \tag{57c}$$

$$= \max_\theta \prod_j^N \mathbb{E}_{p(\boldsymbol{\epsilon})} p(\mathbf{y}_j|f_\theta(\boldsymbol{\epsilon})) \tag{57d}$$

$$\approx \max_\theta \prod_j^N \frac{1}{M} \sum_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})}^M p(\mathbf{y}_j|f_\theta(\boldsymbol{\epsilon})). \tag{57e}$$

In the first line, we rewrite the marginal likelihood of the observed data taking into account that the data are independent and identically distributed. In the second line, we rewrite the individual marginal likelihoods and remove the conditioning on θ in the likelihood $p(\mathbf{y}|\mathbf{x})$ given the hierarchical model assumption. Then, in the third line, we rewrite the integral as an expectation and in the fourth line, we use the LOTUS theorem. Finally, in the last line, we approximate the integral with Monte Carlo integration.

As it is often done in statistics, we do not maximize the marginal likelihood directly but rather maximize the logarithm of the marginal likelihood which is numerically more stable. We also use the log-sum-exp trick and therefore maximize:

$$\max_\theta \log(p(\mathbf{Y}|\theta)) \approx \max_\theta \log \left[\prod_j^N \frac{1}{M} \sum_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})}^M p(\mathbf{y}_j|f_\theta(\boldsymbol{\epsilon})) \right] \tag{58a}$$

$$= \max_\theta \sum_j^N \log \left[\frac{1}{M} \sum_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})}^M \exp(\log(p(\mathbf{y}_j|f_\theta(\boldsymbol{\epsilon})))) \right] \tag{58b}$$

$$= \max_\theta \sum_j^N \log \left[\sum_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})}^M \exp(\log(p(\mathbf{y}_j|f_\theta(\boldsymbol{\epsilon})))) \right] - N \log M \tag{58c}$$

$$= \max_\theta \sum_j^N \operatorname{logSumExp}_{\boldsymbol{\epsilon}} (\log(p(\mathbf{y}_j|f_\theta(\boldsymbol{\epsilon})))) - \cancel{N \log M}. \tag{58d}$$

In the first line, we take the logarithm of Equation 57e. Then, we use the logarithm product rule and introduce an identity term that will allow us to use the log-sum-exp trick. In the third line, we use again the logarithm product rule and remove $\log M$ from the sum

over j. Finally, in the last equation, we rewrite the log-sum-exp operator and $N \log M$ can be omitted as it is constant with respect to θ.

In the limit, for a large number of observations N, maximizing the data marginal likelihood is equivalent to minimizing the Kullback–Leibler divergence between the target distribution $p(\mathbf{y})$ and the learned marginal likelihood $q(\mathbf{y}|\theta) = \int \tilde{p}(\mathbf{y}|\mathbf{x}) q_\theta(\mathbf{x}) \mathbf{dx}$[2]:

$$\begin{aligned} \mathrm{KL}(p(\mathbf{y})||q(\mathbf{y}|\theta)) &= \mathbb{E}_{p(\mathbf{y})} \left[\log(p(\mathbf{y}) - \log q(\mathbf{y}|\theta))\right] && \text{(59a)} \\ &= -\mathbb{E}_{p(\mathbf{y})} \left[\log q(\mathbf{y}|\theta)\right] + H(p) && \text{(59b)} \\ &\approx -\frac{1}{N} \sum_j^N \log q(\mathbf{y}_j|\theta) + \cancel{H(p)}. && \text{(59c)} \end{aligned}$$

In the first line, we use the definition of the Kullback–Leibler divergence. Then, in the second line, we develop the expectation and rewrite the entropy of the target distribution. Finally, in the third line, we approximate the expectation with Monte Carlo integration from the observed dataset $\mathbf{Y}$ and the entropy term can be omitted as it is constant.

Minimizing Equation 59c is equivalent to maximizing Equation 57a (up to a constant $\frac{1}{N}$) as the logarithm is a monotonic function. Therefore, Equation 58 is optimized when the Kullback–Leibler divergence is minimized or equivalently, when the learned marginal likelihood equals the target evidence distribution. This means that in theory, we are guaranteed to learn a distribution $q_\theta(\mathbf{x})$ such that when corrupted by the surrogate model, reproduces the target distribution $p(\mathbf{y})$:

$$\begin{aligned} p(\mathbf{y}) &= q_\theta(\mathbf{y}|\theta) && \text{(60a)} \\ &= \int \tilde{p}(\mathbf{y}|\mathbf{x}) q_\theta(\mathbf{x}) \mathbf{dx}. && \text{(60b)} \end{aligned}$$

where the first line results from Equation 59 that is minimized when $p(\mathbf{y}) = q_\theta(\mathbf{y}|\theta)$. Then, we develop the definition of the learned marginal likelihood by marginalizing over $\mathbf{x}$.

Therefore, the problem is optimized when the learned distribution $q_\theta(\mathbf{x})$ is such that equation 60b holds, or in other words, such that when corrupted by the surrogate model, reproduce the distribution $p(\mathbf{y})$. Even if the surrogate model was a perfect approximation of the likelihood function $p(\mathbf{y}|\mathbf{x})$, depending on its shape, there may exist different solutions to the problem. If the surrogate model approximates perfectly well the likelihood function, the exact distribution $p(\mathbf{x})$ over unseen data will belongs to the set of solutions that optimize the problem. However, there is no guarantee to converge to this solution among the others. In that matter, proper regularization should be introduced.

5.4 Amortized Inference

In this section, we assume that we have observed a large number of i.i.d. events $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$ and that we are interested in learning the hidden variables $\mathbf{x}_i$ for each observation.

The surrogate model allows us to evaluate an approximation of the likelihood function and as explained in Section 5.3, we can learn a prior distribution over the unseen variables.

2. Until now, the true likelihood $p(\mathbf{y}|\mathbf{x})$ and its approximation $\tilde{p}(\mathbf{y}|\mathbf{x})$ have been used interchangeably, for clarity. Here, we explicitly state the difference between them because it impacts the learned marginal distribution $q(\mathbf{y}|\theta)$.

In this section, we highlight how these two quantities can be used to do inference and learn a posterior for each observed variable. Each of the proposed methods have their own strengths and drawbacks, and a full comparison between them is beyond the scope of this dissertation. Therefore, we only superficially state some strengths and drawbacks given our setting.

Rejection sampling

In all generality, the only requirements to learn a prior as explained in Section 5.3 is that the prior model can be easily sampled from. We show here how given such a prior and the an approximation of the likelihood one can sample from the posterior without additional requirements. Given an observation $\mathbf{y}$, rejection sampling iteratively sample a data point $\mathbf{x}$ from the prior as well as a random variable u from a uniform distribution $\mathcal{U}(0,1)$ and accept the data point if:

$$u < \frac{p(\mathbf{y}|\mathbf{x})}{M} \tag{61}$$

where M is a constant that should meet the requirement:

$$M > p(\mathbf{y}|\mathbf{x}), \quad \forall \mathbf{x}. \tag{62}$$

Choosing the constant M is therefore the most critical part of this method. This can be done by sampling a large number of $\mathbf{x}'s$ and evaluating empirically the maximum value of the likelihood.

The algorithm can be efficiently parallelizable. The first step consists in sampling many $\mathbf{x}'s$, which can be done in constant time on graphics processing unit (GPU) given enough memory. Then, the likelihood of each $\mathbf{x}$ should be evaluated which can also be done in parallel in constant time and M is defined as the maximum value of these likelihoods. To finish, a random variable can be sampled and associated to each $\mathbf{x}$, and only points that meet the requirement defined in Equation 61 are kept. The last two steps are also efficiently parallelizable.

Importance Sampling

As in the previous point, importance sampling only requires to sample from the prior and not to evaluate its density. By sampling from the prior distribution and weighting samples by the likelihood, one can infer properties about the posterior, for example by plotting the weighted distribution. It is also possible to compute the expected value of the posterior distribution from the weighted samples. Indeed,

$$\mathbb{E}_{p(\mathbf{x}|\mathbf{y})}[\mathbf{x}] = \int \mathbf{x} p(\mathbf{x}|\mathbf{y}) \mathbf{dx}. \tag{63a}$$

$$= \int \frac{1}{Z} \mathbf{x} p(\mathbf{x}) p(\mathbf{y}|\mathbf{x}) \mathbf{dx}. \tag{63b}$$

$$= \mathbb{E}_{p(\mathbf{x})}[\frac{1}{Z} \mathbf{x} p(\mathbf{y}|\mathbf{x})] \tag{63c}$$

where in the first line, we rewrite the definition of the expectation. Then, in the second line, we use the Bayes rule and rewrite $p(\mathbf{y})$ as Z. Finally, in the last line we rewrite the expectation with respect to $p(\mathbf{x})$.

Therefore, if we knew the normalizing constant Z, we could approximate the expected value of the posterior distribution by sampling data points from the prior distribution and

weighting them by $\frac{p(\mathbf{y}|\mathbf{x})}{Z}$. Fortunately, it can be showed that the normalizing constant can be approximated by the mean of the non-normalized weight, that is by the mean of the sample likelihoods.

Importance sampling is interesting here as it can also be highly parallelizable. Given enough GPU memory, a large number of data points can be sampled in constant time. Then, evaluating their likelihood can also be done in parallel in constant time.

Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) allows to sample from a distribution when all we can do is evaluate its density (or the density up to a constant factor)[3].

Therefore, if we can evaluate the likelihood and the prior, we can evaluate the posterior up to a constant term and sample form it with a MCMC sampler. In order to evaluate the prior, a density estimator of the distribution over the unseen data should be trained as described in Section 5.3 but now, it is no longer sufficient to model the prior distribution with any generative model. The generative model should allow to evaluate the density $q_\theta(\mathbf{x})$. Depending on the model chosen, the prior density may also be differentiable and as the likelihood function defined by the surrogate is differentiable as well, one can use MCMC samplers that require to compute the derivatives with respect to the latent variables such as Hamiltonian Monte Carlo.

Unfortunately, MCMC samplers cannot be used without careful checks and proper tuning. Moreover, these algorithms are not easily parallelizables and therefore can be slow to run, especially when the densities are evaluated by Neural Networks.

Variational inference

Finally, in order to go beyond sampling or estimating the properties of the posterior, one can use variational inference in order to directly learn a density estimator of the posterior. This allows, once trained to directly sample from the posterior or evaluate the density of data points. If the learned model of the prior allows to differentially evaluate the density $q_\theta(\mathbf{x})$, it is possible to maximize the evidence lower bound defined in Equation 15 to fit a density estimator on the posterior distribution.

3. For a clear review about MCMC, van Ravenzwaaij et al. (2016) provide a simple introduction to Markov Chain Monte Carlo.

6. Results

In this section, we illustrate the methods defined hereabove. First, we introduce four benchmark simulators inspired by the literature. Then, we show how the surrogate model can be used alone to a solve standard maximum likelihood estimation problem. Afterwards, we go a step further and learn a prior density over the unseen source data, first when there is a single observation and then, when the number of observations increases. Finally, we show how the learned prior and the surrogate model can be used together to learn a posterior distribution over unseen source data.

Before presenting our results, let us introduce a few terms.

Proposal prior. In order to train the surrogate model, pairs of source data and corrupted data should be generated. Once the surrogate is trained, our methods require minimal domain knowledge. However, choosing a distribution from which training source data should be sampled is a crucial step and may require domain knowledge. We refer to this distribution as the proposal prior.

Regenerated data. When we learn a point estimate $\hat{\mathbf{x}}$, we may want to use the simulator $\mathcal{M}$ to run simulations with $\hat{\mathbf{x}}$ as parameters to see how regenerated data compare with a target observation. Or in other words, we call regenerated data, samples from the distribution $p(\mathbf{y}|\hat{\mathbf{x}})$. When we learn a density $\tilde{p}(\mathbf{x})$ over $\mathbf{x}$ rather than a point estimate, we called regenerated data, samples from the distribution:

$$p(\hat{\mathbf{y}}) = \int p(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})\mathbf{dx}. \tag{64}$$

Reconstruction error. Given generated data, we may want to compute a distance between these data points and a target observation $\mathbf{y}_{obs}$. Therefore, when a point estimate $\mathbf{x}^*$ is learned, unless stated otherwise, we define the reconstruction error as:

$$E_{rr} = \mathbb{E}_{p(\mathbf{y}|\mathbf{x}^*)}||\mathbf{y} - \mathbf{y}_{obs}||_2^2. \tag{65}$$

When we learn a density over $\mathbf{x}$ rather than a point estimate, unless stated otherwise, the reconstruction error is defined as:

$$E_{rr} = \mathbb{E}_{p(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})}||\mathbf{y} - \mathbf{y}_{obs}||_2^2. \tag{66}$$

Distance to ground truth parameters. When doing inference on an observed variable $\mathbf{y}_{obs}$ generated by ground truth parameters $\mathbf{x}^*$, we may want to compute the distance between the learned point estimate $\hat{\mathbf{x}}$ and $\mathbf{x}^*$. unless stated otherwise, we use the euclidean norm: When a distribution over $\mathbf{x}$ is learned and that we want to assess how close samples from this distribution are to the ground truth parameter, we define the distance to ground truth parameter the distance:

$$d = \mathbb{E}_{\tilde{p}(\mathbf{x})}||\mathbf{x} - \mathbf{x}^*||_2^2. \tag{67}$$

aim at computing the distance between a learned point estimate $\hat{x}$ and the ground truth parameter $\mathbf{x}^*$ that have generated the observation $\mathbf{y}_{obs}$

6.1 Experimental Setup

Simple Gaussian

To start with, we introduce a simple 2-D gaussian stochastic simulator. Given parameters $\mathbf{x} \in \mathbb{R}^2$, the simulator generates $\mathbf{y} \in \mathbb{R}^2$ according to:

$$\mathbf{m} = \mathbf{x} \tag{68a}$$

$$\mathbf{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{68b}$$

$$\mathbf{y} = \mathcal{N}(\mathbf{m}, \mathbf{\Sigma}). \tag{68c}$$

The prior $p(\mathbf{x})$ is a 2-D multivariate gaussian with means $\mathbf{m} = [3, 7]^\top$ and an identity covariance matrix.

Two-Moons

Next, we use the two-moons simulator introduced in Greenberg et al. (2019). Given parameters $\mathbf{x} \in \mathbb{R}^2$, the simulator generates $\mathbf{y} \in \mathbb{R}^2$ according to:

$$a \sim \mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2}) \tag{69a}$$

$$r \sim \mathcal{N}(0.1, 0.01^2) \tag{69b}$$

$$\mathbf{p} = [r\cos(a) + 0.25, r\sin(a)]^\top \tag{69c}$$

$$\mathbf{y} = \mathbf{p} + [-\frac{|x_1 + x_2|}{\sqrt{2}}, \frac{-x_1 + x_2}{\sqrt{2}}]^\top. \tag{69d}$$

The prior $p(\mathbf{x})$ is uniform between $[-1, 1]$ for each x_i.

Simple Likelihood and Complex Posterior

The next simulator (Papamakarios et al., 2019) has a simple likelihood but complex posterior (SLCP). Given parameters $\mathbf{x} \in \mathbb{R}^5$, the simulator generates $\mathbf{y} \in \mathbb{R}^8$ according to:

$$\mathbf{m}_\mathbf{x} = [x_1, x_2]^\top \tag{70a}$$

$$s_1 = x_3^2 \tag{70b}$$

$$s_2 = x_4^2 \tag{70c}$$

$$\rho = \tanh(x_5) \tag{70d}$$

$$\mathbf{\Sigma} = \begin{bmatrix} s_1^2 & \rho s_1 s_2 \\ \rho s_1 s_2 & s_2^2 \end{bmatrix} \tag{70e}$$

$$\mathbf{x}_j \sim \mathcal{N}(\mathbf{m}_\mathbf{x}, \mathbf{S}_\mathbf{x}), \quad j = 1, ..., 4 \tag{70f}$$

$$\mathbf{x} = [\mathbf{x}_1^\top, ..., \mathbf{x}_4^\top]^\top. \tag{70g}$$

The prior $p(\mathbf{x})$ is uniform between $[-3, 3]$ for each x_i. Such a prior includes nonlinearities in the posterior. The posterior is further complexified due the symmetries introduced by the squared operations in Equations 70b and 70c.

Inverse Kinematics

Ardizzone et al. (2018) introduced a problem where $\mathbf{x} \in \mathbb{R}^4$ but which can still be easily visualisable in 2-D. They model an articulated arm that can move vertically along a rail

and that can rotates at three joints. Given parameters $\mathbf{x}$, the arm's end point $\mathbf{y} \in \mathbb{R}^2$ is defined as:

$$y_1 = x_1 + l_1 \sin(x_2) + l_2 \sin(x_2 + x_3) + l_3 \sin(x_2 + x_3 + x_4) \tag{71a}$$
$$y_2 = l_1 \cos(x_2) + l_2 \cos(x_2 + x_3) + l_3 \cos(x_2 + x_3 + x_4) \tag{71b}$$

with arm lengths $l_1 = l_2 = 0.5, l_3 = 1.0$.

As the forward model defined in equation 71 is deterministic and that we are interested in stochastic simulators, we add noise at each rotating joint. Noise is sampled from a normal distribution $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = 0.00017$ rad $\equiv 0.01°$. This gives a repeatability[4] of about 1.7 mm which is quite high for industrial applications and means that joint noise in industrial robots are even smaller.

The prior $p(\mathbf{x})$ follows a gaussian $\mathcal{N}(0, \sigma_i^2)$ for each $\mathbf{x}_i$ with $\sigma_1 = 0.25$ and $\sigma_2 = \sigma_3 = \sigma_4 = 0.5 \equiv 28.65°$. This prior favors arm configurations as represented in Figure 4.

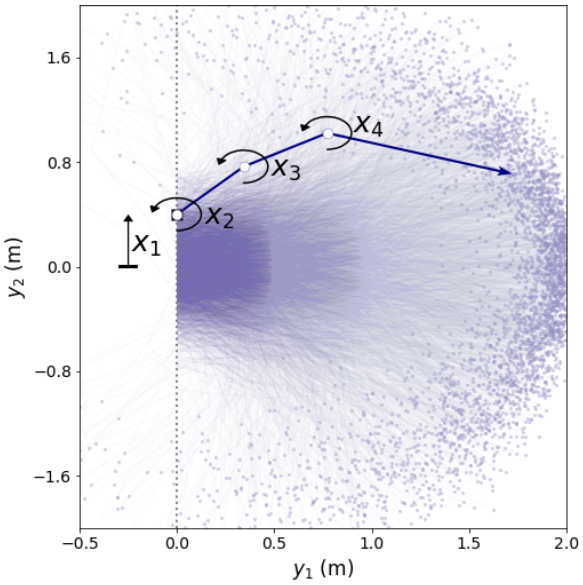


Figure 4: **Inverse kinematics.** An articulated arm is mounted on a rail. The position on the rail is determined by the variable x_1. Then, $x_{2:4}$ are the joint angles. The background shows different arm configurations.

6.2 Surrogate Models

All toy simulators are modeled by real NVP transformations. All surrogates are made of 4 NVP layers where the networks $s(\cdot)$ and $t(\cdot)$ are multilayer perceptron (MLPs) with three layers of 50 units each with ReLU between every two layers. Batch normalization between MLP layers as well as between NVP layers as suggested in Papamakarios et al. (2017) did not improve results. Therefore, no batch normalization was used. As the output of the network $s(\cdot)$ is exponentiated in Equation 35 this may lead to unstable behaviours during training. In the initial NVPs paper, $s(\cdot)$ is squashed by a hyperbolic tangent function

4. In robotics, repeatability defines how well a robot can achieve the same task. Repeatability has been measured by ANSI/RIA R15.05-1 standard as explained in Dagalakis (2007).

multiplied by a trainable parameter. We rather use soft clamping of scale coefficients as introduced in Ardizzone et al. (2019):

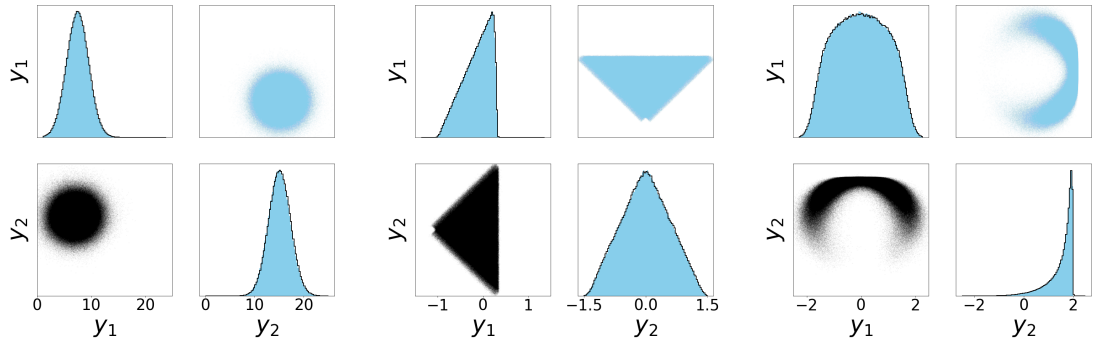
$$s_{clamp} = \frac{2\alpha}{\pi} \arctan(\frac{s}{\alpha}) \tag{72}$$

which gives $s_{clamp} \approx s$ for $s \ll |\alpha|$ and $s_{clamp} \approx \pm\alpha$ for $|s| \gg \alpha$.

We performed a grid search over hyperparameters and found $\alpha = 1.9$ to be a good value for most architectures, as in Ardizzone et al. (2019). Therefore, we fixed α to 1.9 for all models.

As it is commonly done, each NVP layer only modifies half the dimension of its inputs and between every two layers, dimension are reversed so that no dimension is left unchanged.

Finally, each surrogate was trained with a training dataset of only 15,000 samples. A validation set of 5,000 samples was used to avoid overfitting. Datasets were generated by producing pairs of uncorrupted (sampled from a proposal prior) and corrupted (obtained by running the simulator) data. The used proposal prior are the priors defined in Section 6.1. Optimization was run for 300 epochs over the training dataset and the Adam optimizer was used with default parameters and weight decay set to 5×10^{-5}. Figures 5 and 6 show the (empirical) learned distributions against the target distribution $p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\mathbf{dx}$ on the toy datasets. The target distributions have been generated after training, by sampling parameters from the proposal prior $p(\mathbf{x})$ and corrupting them with the simulator. The learned distributions are obtained by corrupting the same parameters but this time, with the surrogate.



(a) Simple gaussian. (b) Two-moons. (c) Inverse kinematics.

Figure 5: For different simulators, given a set of parameters $\mathbf{x}$ sampled a proposal prior $p(\mathbf{x})$, the empirical distributions of corrupted data obtained by running the physical simulator (in blue) against running the trained surrogate (in black) with parameters $\mathbf{x}$ are shown. The diagonal show the 1-D marginal distributions while others are pairwise scatter plots.

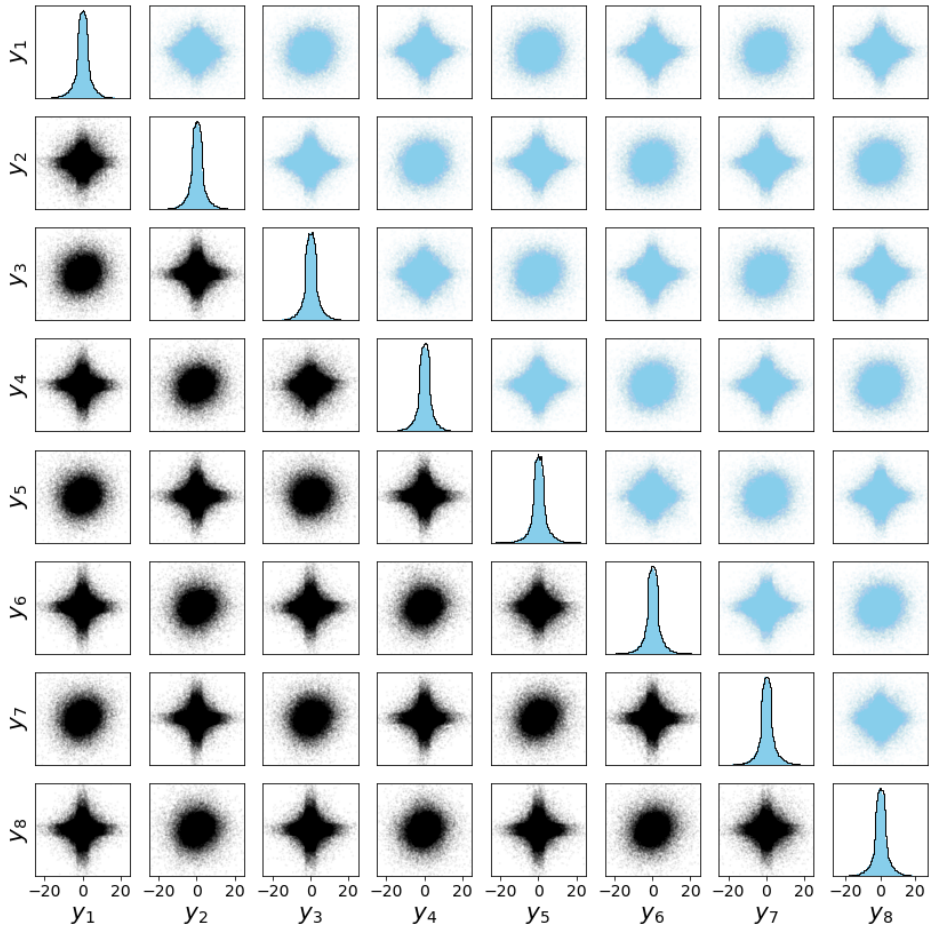


Figure 6: Simple likelihood and complex posterior (SLCP) simulator. Given a set of parameters $\mathbf{x}$ sampled a proposal prior $p(\mathbf{x})$, the empirical distribution of corrupted data obtained by running the physical simulator (in blue) against running the trained surrogate (in black) with parameters $\mathbf{x}$ is shown. The diagonal show the 1-D marginal distributions while others are pairwise scatter plots .

6.3 Point Estimation

In this section we study how to retrieve a point estimate that may have generated an observation. To do so, given a single observation $\mathbf{y}_{obs}$, we aim at finding $\hat{\mathbf{x}}$ that solves:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \log p(\mathbf{y}_{obs}|\mathbf{x}). \tag{73}$$

We solve this problem iteratively with the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization algorithm[5] (Liu and Nocedal, 1989). It is a quasi-Newton method that approximates the inverse of the Hessian matrix with limited memory.

Simulators	Ground truth parameters $\mathbf{x}^*$
Simple gaussian	$[4.9, 6.5]^\top$
Two-moons	$[0.0, 0.7]^\top$
SLCP	$[0.7, -2.9, -1.0, -0.9, 0.6]^\top$
Inverse kinematics	$[0.1, -0.4, 0.5, -0.1]^\top$

Table 1: For different simulators, ground truth parameters $\mathbf{x}^*$ used to run the stochastic simulator and generate a single observation $\mathbf{y}_{obs}$.

Experimental Setup Throughout this dissertation, we assume minimal knowledge about the corruption process and the distribution over parameters $\mathbf{x}$. Yet, when solved iteratively by gradient ascent, the optimization process described in Equation 73 needs a starting point $\mathbf{x}_0$. Choice have been made to sample $\mathbf{x}_0$ from a zero-mean, unit-variance isotropic gaussian distribution. This heavily samples points out of the proposal prior space used to train the surrogate model for the two-moons, inverse kinematics and simple gaussian toy problems. For the gaussian toy problem, all points are sampled out of the space used to train the surrogate. Nonetheless, $\mathbf{x}'s$ are pushed towards the expected solution even if they are initialized in a out-of-training region. These results suggest that the surrogate model has nicely generalized and learned the likelihood function beyond the domain from the training dataset. Despite these good empirical results, Equation 34 only guarantees that given enough training sample, the learned likelihood matches the true likelihood in the support of the proposal prior. In order to avoid unexpected behaviors, one can therefore solve a constrained maximum likelihood problem where $\mathbf{x}$ is constrained to stay within this proposal.
For each simulator, we generate a single observation $\mathbf{y}_{obs}$ by running the stochastic simulator with the ground truth parameters defined in Table 1. Then, we repeatedly solve the optimization problem defined in Equation 73 with multiple random starting points to assess the consistency of the algorithm. The optimization is solved with the L-BFGS optimizer with default parameters and learning rate set to 10^{-1} for the simple gaussian, two-moons and inverse kinematics simulators. For the SLCP simulator, the learning rate is set to 10^{-2}. The algorithm is run for 400 epochs.

Reconstruction Error In Table 2, it can be observed that for the two-moons and inverse kinematics problems, the reconstruction error is very small. On the contrary, the reconstruction error for the other simulators is much higher but this is not necessarily bad. Indeed, this is mainly due to the simulator itself that is stochastic and has a broad likelihood. Figure 7 shows the regenerated data against the target. When pushed to the simulator, the learned $\hat{\mathbf{x}}'s$ produce $\mathbf{y}'s$ that are concentrated at the observed target point $\mathbf{y}_{obs}$. The fact that regenerated data are spread out is due to the simulator's stochasticity.

Distance to ground truth parameters Table 2 also shows the distance between the optimized $\hat{\mathbf{x}}'s$ and the ground-truth parameters over multiple runs. For most problems, the distance to the true parameters is not that small but it is not necessarily a problem. Indeed,

5. We used the public implementation from PyTorch.

	Reconstruction Error $\mathbb{E}_{p(\mathbf{y}\|\mathbf{x}^*)}\|\|\mathbf{y}-\mathbf{y}_{obs}\|\|_2^2$.	Distance to ground truth parameters $\|\|\mathbf{y}_{obs}-\mathbf{x}^*\|\|^2$	Distance to true MLE $\|\|\mathbf{y}_{obs}-\mathbf{x}_{MLE}\|\|^2$
Simple Gaussian	1.2544 ± 0.6644	0.4049 ± 0.0005	0.0531 ± 0.0006
Two-Moons	0.0888 ± 0.0561	$0.0458 \pm 0.0088^*$	/
SLCP	2.9131 ± 2.4723	$1.6190 \pm 0.6437^*$	$0.8563 \pm 0.8616^*$
Inverse Kinematics	0.0034 ± 0.0018	0.8026 ± 0.3555	/

Table 2: Reconstruction error between an observation and data regenerated by running simulations with the (learned) most likely parameter estimates as input. We also show the distance between the learned MLEs and the true parameters that have generated the observation as well as the distance to the true MLE — when it can computed. Asterisks indicate that a norm that takes the simulator symmetries into account has been used rather than the euclidean norm. We report the average metrics over 100 runs, error bars equal to the standard deviation.

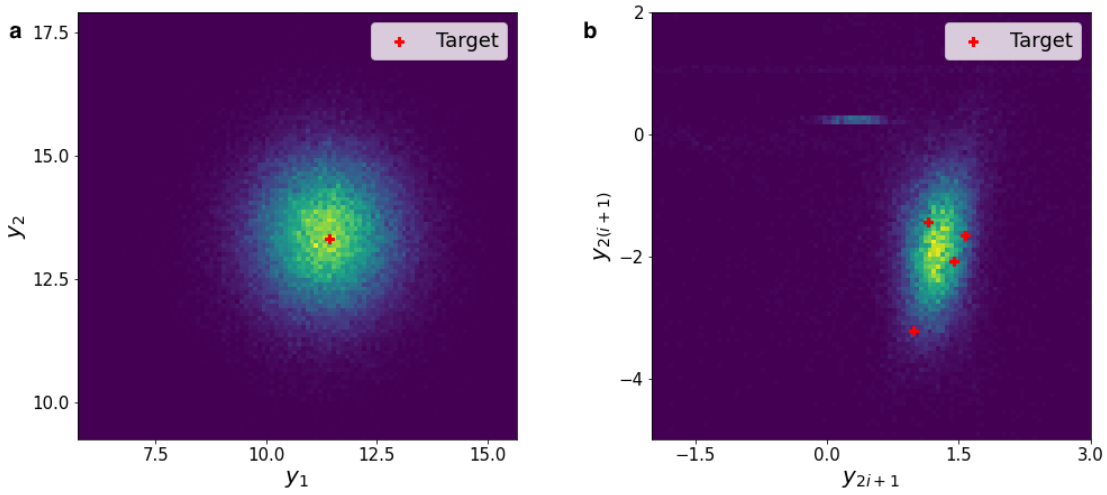


Figure 7: Target observation against reconstructed data obtained by running the simulator with learned point estimates as input. The point estimates are learned by solving a maximum likelihood problem with different starting points. **a** Simple gaussian simulator. **b** SLCP simulator. For clarity, we show the target as four 2-D variables as it is the case from the problem definition. Nonetheless, the simulator is seen as a black-box and the target is treated everywhere as an 8-D variable and it should be considered as such. For both simulators, regenerated data are in strong agreement with the observation.

the stochastic simulator may have generated an observation $\mathbf{y}_{obs}$ with a low likelihood and therefore, there are better $\mathbf{x}'s$ than the ground truth that explain the observation. At the same time, for a given $\mathbf{y}_{obs}$, there may be many equally likely $\mathbf{x}'s$ that can generate it. Therefore, the optimization algorithm may not always converge to the ground truth parameters. This can be clearly seen in Figure 8 where all optimized $\hat{\mathbf{x}}'s$ consistently reproduce data close to the observation, but many different $\hat{\mathbf{x}}'s$ are learned.

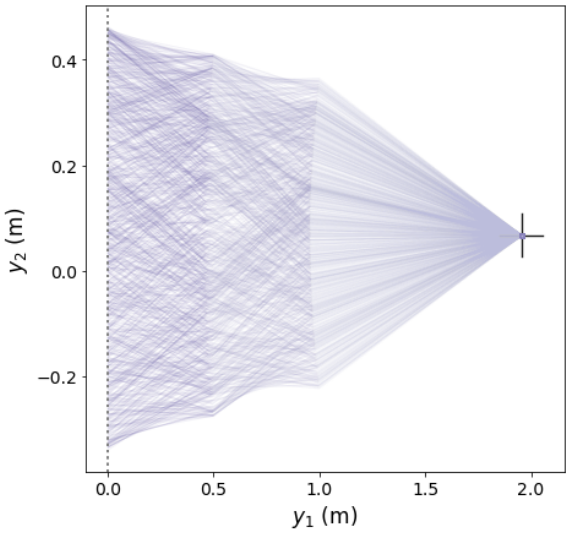


Figure 8: Articulated arm configurations learned by solving a maximum likelihood problem with different initializations. The cross indicates the target observation $\mathbf{y}_{obs}$. The algorithm do not always converge to the same solution but consistently learn an arm configuration that allows to reach the target.

Distance to ground truth MLE Finally, in Table 2 we also report the distance between the optimized solutions and the true most likely estimate (when it can be computed). On the simple gaussian problem, we consistently retrieve the true MLE. For the SLCP problem, about $\sim 2\%$ of the optimized data were outliers[6] and have been removed. This is due to large losses that induce large update steps and thus, $\mathbf{x}'s$ that have large values. Then, the algorithm diverge and do not recover. This may be due to a poorly tuned L-BFGS optimizer with respect to this problem. Here, in order to compute the distance between learned and exact MLEs, we use a norm that takes the symmetries of the problem into account. The norm is defined as:

$$d = \sqrt{(x_1 - x_1^*)^2 + (x_2 - x_2^*)^2 + (|x_3| - |x_3^*|)^2 + (|x_4| - |x_4^*|)^2 + (\tanh(x_5) - \tanh(x_5^*))^2}. \tag{74}$$

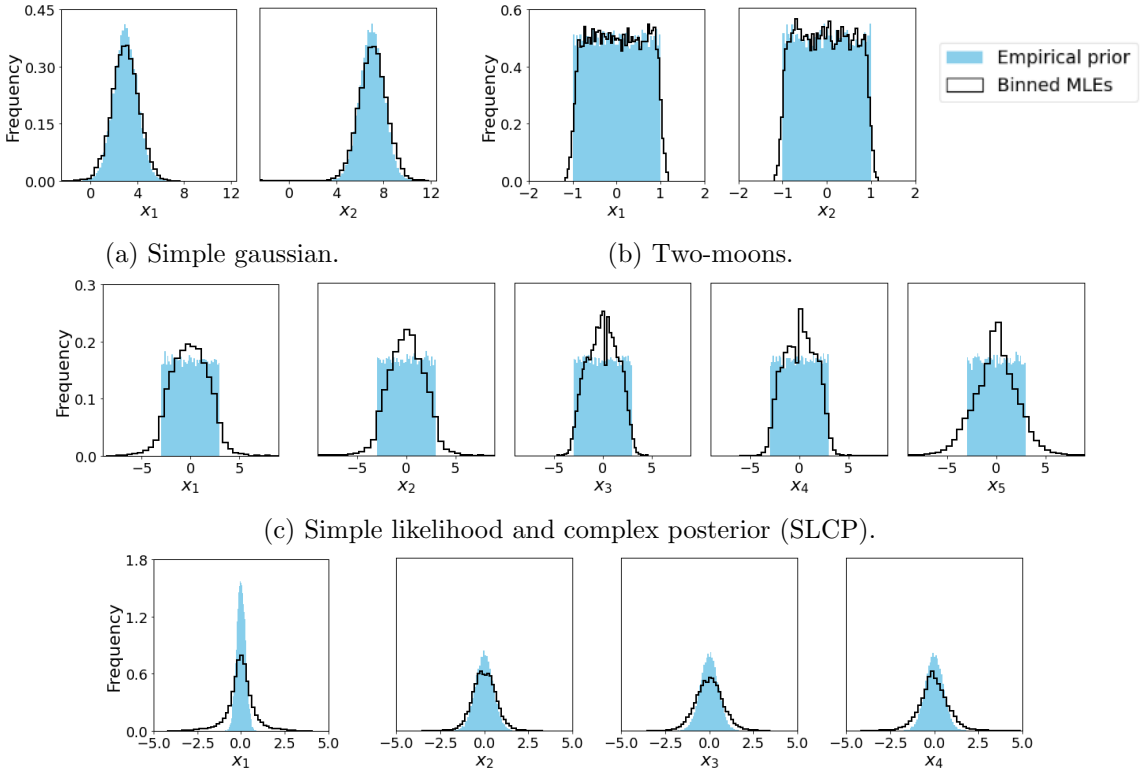
This norm equals zero if the retrieved $\hat{\mathbf{x}}$ is one of the most likely estimate — there are four of them in the SLCP problem due to the squared operations in Equations 70b and 70c. Here, the distance to the MLE and the variance are not so small which means that the optimization algorithm does not always converges to the same point. Nonetheless, results are decent and optimized $\hat{\mathbf{x}}'s$ produce data close to the target (Figure 7). A better tuning of the L-BFGS optimizer may lead to even better results.

To sum up, using the surrogate likelihood, we are able to consistently retrieve $\mathbf{x}'s$ that explain the observation and that regenerate data close to it. No regularization was used and even if $\mathbf{x}$ is sometimes outside of the proposal space used to train the surrogate model, the surrogate has well generalized and allows to guide the optimization process to one of the expected most likely estimates.

6. We define outliers as data points that when optimized, are beyond the proposal prior distribution used to train the surrogate.

Empirical Prior From Single Estimates

In this section we assume that we have observed N corrupted data $\{\mathbf{y}_1, ..., \mathbf{y}_N\}$. We retrieve the most likely estimates of each observations and bin the retrieved $\hat{\mathbf{x}}'s$ to empirically approximate the prior. Binning is done in 1-D for each dimension by defining equally likely bins in the range of the values taken by the data. Binning is only done at the end of the optimization process and thus, we could bin in multiple dimensions as well. Of course, one can do better and consider all the observed data as a whole (we treat this problem in Section 6.5) rather than treating them independently. Nonetheless, this can be embedded in a more complex algorithm for example, to initialize the prior before optimization.



(a) Simple gaussian.

(b) Two-moons.

(c) Simple likelihood and complex posterior (SLCP).

(d) Inverse kinematics.

Figure 9: Empirical prior obtained by binning (unseen) parameters against their binned most likely estimates (MLEs). The later are obtained by solving a maximum likelihood problem for each observation.

Experimental Setup For all toy problems, we use 25,000 observations. Experiments showed that using more observations do not lead to a better empirical prior. As in the previous section, for each observation $\mathbf{y}_i$, we start the algorithm by initializing the starting point $\mathbf{x}_{i,0}$ with a sample from a zero-mean unit-variance isotropic gaussian distribution. The L-BFGS implementation from PyTorch is not parallelizable and may takes several minutes to solve the MLE problem. Therefore, it does not scale to large numbers of observations and therefore, we switched to the Adam optimizer to parallelize the process. There is no stochasticity in the optimization and Adam was chosen for its learning rate adaptability. Therefore, a good tuning of another standard gradient descent algorithm or using a second order method may further improve results. The Adam optimizer was used with default

parameters and a learning rate set to 10^{-3} for the simple gaussian and two-moons simulators, 10^{-4} for the inverse kinematics simulator and 5×10^{-4} for the SLCP simulator. The optimization problem was run for 25,000 epochs.

Figure 9 shows the learned 1-D learned empirical distribution against the true empirical prior. Figure 9a and 9b show that the learned empirical distribution is already a good approximation of the unseen empirical prior on the simple gaussian and two-moons problems. Figure 9c shows that the binned MLEs on the higher dimensional problem with an easy likelihood but complex posterior do not match the empirical prior as well as for the other problems. This is nonetheless a good first approximation that can be used to pretrain a variational prior, for example. Figure 9d shows results on the inverse kinematics simulator. The learned and expected distributions do not closely match but the learned distributions are gaussian distributions as expected. The empirical standard deviation retrieved over the first variable is smaller than the ones of the other variable, as it is the case on the distribution that has generated source data.

6.4 Density Estimation

In this section, we go beyond point estimation and aim at learning a distribution over the source data $\mathbf{x}$ that may have generated a single observation $\mathbf{y}_{obs}$. Four different approaches to solve this problem have been defined in Section 5.2 and all of them perform fairly similarly in terms of reconstruction error and distance to the ground truth parameters. While the empirical Bayes method and EM algorithm defined in Section 5.2 often have difficulties to capture all the modes and sometimes collapse to a single point, the expected log likelihood loss often learns all the modes and broader distributions due to the implicit KL constraint in the loss definition[7]. Therefore, throughout this section, we focus on this loss unless stated otherwise. Thus, we model the prior distribution $p(\mathbf{x})$ by a density estimator $q_\theta(\mathbf{x})$ and optimize iteratively the parameters θ of the density estimator by gradient descent on the following loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{q_\theta(\mathbf{x})} \log p(\mathbf{y}_{obs}|\mathbf{x}). \tag{75}$$

The expectation is approximated by Monte Carlo integration. We used 256 integration step for all simulators. When this is optimized on GPU, using more integration steps do not necessarily increase the computational time. We used the Adam optimizer with a learning rate set to 10^{-3}. The density estimator are modelled with NVP layers for the simple gaussian, two-moons and inverse kinematics simulators. Three NVP layers were used with the networks $s(\cdot)$ and $t(\cdot)$ modeled by MLPs with 3 layers of 16 units each and with RELU between every two layers. For doing inference on the SLCP simulator, UMNN-MAFs were used. We used the public implementation[8] from Wehenkel and Louppe (2019). We used a single layer with the embedding and integrand networks made of 4 layers of 100 units. The optimization process is trained for 5,000 epochs.

We also compare our method to two sequential-neural based methods. This is mostly for illustration purpose because these methods are learning a posterior distribution in a Bayesian setting, which is substantially different to what we are doing. Our aim is to

7. In this study, we do not put much attention on the handcrafted loss approach as it requires domain knowledge to design a proper cost function between the observation and regenerated corrupted data. Furthermore, the cost function chosen makes implicit assumptions on the simulator likelihood function. In this study, we have the luxury of being able to evaluate an approximation of the likelihood function and therefore, we take advantage of it.
8. https://github.com/AWehenkel/UMNN

highlight that without calling the simulator at inference, we achieve similar performance in terms of reconstruction error. This also gives an order of magnitude about the number of simulations needed by these methods at each inference step. [9] We compare our results to Sequential Neural Likelihood (SNL) proposed in Papamakarios et al. (2019) and SNPE-B (Lueckmann et al., 2017). The first method sequentially learns an approximation of the likelihood function that can be plugged with a prior distribution into a MCMC sampler to sample from the posterior. The second method directly learns an approximation of the posterior, also in a sequential algorithm. Results and figures are produced with the public implementation[10] of Papamakarios et al. (2019). We run the methods with the default hyperparameters presented in the method papers. For low-dimensional toy problems, we use 250 simulations at each round rather that 1,000.

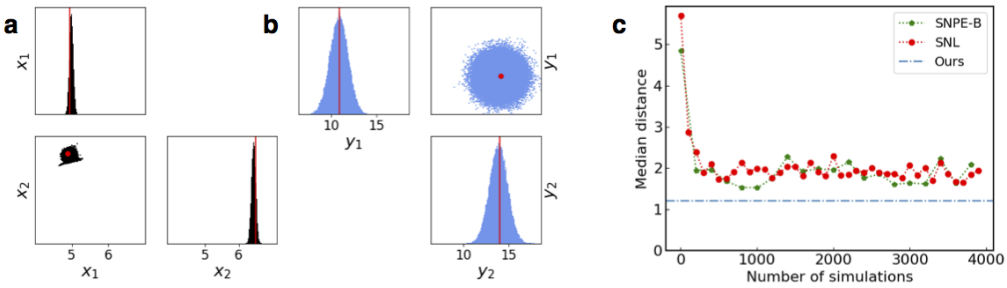


Figure 10: **Simple gaussian.** **a** Learned prior distribution (black) against true parameters (red). **b** Observation (red) against data (blue) regenerated by running simulations with parameters drawn from the learned distribution. **c** Median distance between regenerated data and the observation as a function of the number of calls to the simulator. Our method does not call the simulator at inference.

Figure 10 shows the learned distribution, the regenerated corrupted distribution and a comparison between our results, SNL and SNPE-B on the simple gaussian simulator. The learned distribution does not collapse to the most likely estimate and contains the ground truth parameters. Yet, it is not always the case as the ground truth parameters may generate the observation $\mathbf{y}_{obs}$ with a low likelihood. As we are not learning a posterior distribution, we often learn a tighter distribution that puts most of its mass on parameters for which the observation has a high likelihood. Regenerated data are in strong agreement with the observation. In terms of reconstruction error, our results are better than sequential methods only because we learn a distribution that is often tighter than the posterior distribution learned by sequential methods. As the stochastic simulator has a high variance, our regenerated data concentrate much closer to the observed point and therefore, we have better results in terms of reconstruction error. All results are consistent over multiple runs.

Similarly, we show the learned distribution, the regenerated corrupted distribution and a comparison between our results, SNL and SNPE-B on the two-moons simulator in Figure 11. Due to the absolute values in Equation 69d, for a given observation, there are two moon-shaped manifolds of parameters that may have produce it. It can be observed in Figure

9. For each observation, these methods train a local density estimator of the likelihood function or the posterior distribution, guided by active learning. Therefore, the process should be repeated for each observation but the local estimator can be reused from one observation to another. Thus, the number of calls to the simulator is not constant and tends to decreases as more observations are observed.

10. https://github.com/gpapamak/snl

11 that the learned distribution has well learned the two modes as well as their expected shape. Here as well, reconstructed data are in strong agreement with the observation. Finally, in this problem the distribution that we learn is close to the posterior distribution and therefore, the reconstruction error obtained with our method matches the ones from sequential methods. All results are consistent over multiple runs.

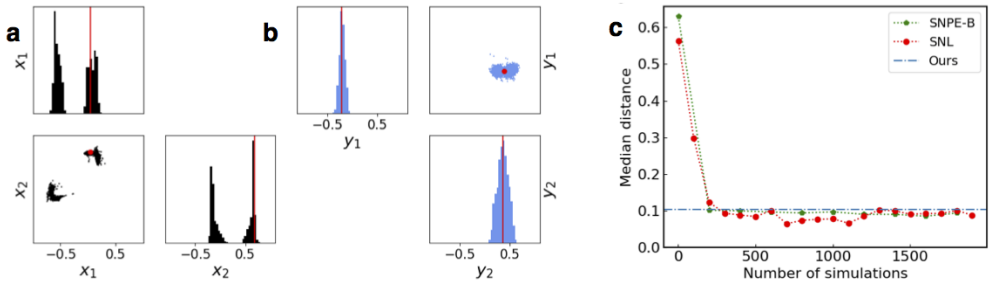


Figure 11: **Two-moons.** **a** Learned prior distribution (black) against true parameters (red). **b** Observation (red) against data (blue) regenerated by running simulations with parameters drawn from the learned distribution. **c** Median distance between regenerated data and the observation as a function of the number of calls to the simulator. Our method does not call the simulator at inference.

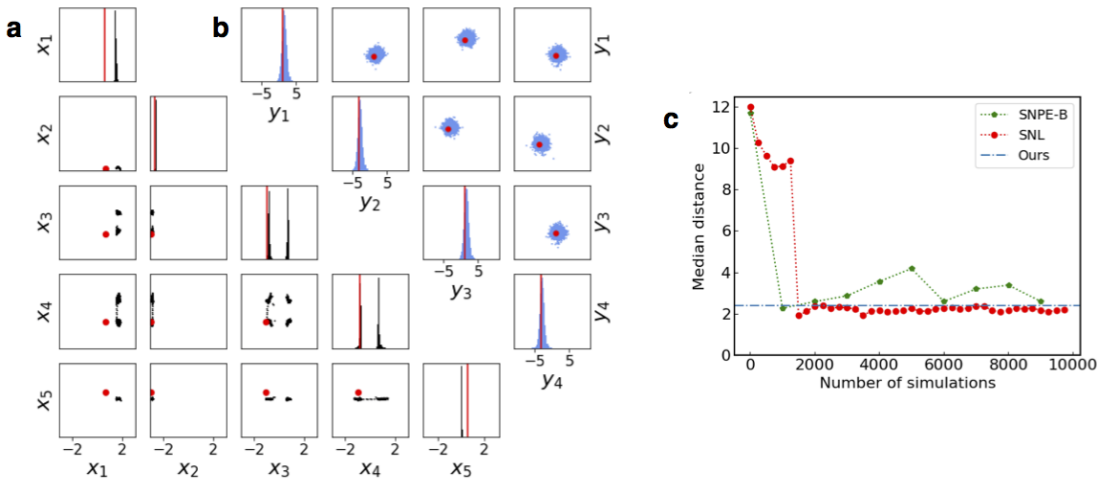


Figure 12: **SLCP.** **a** Learned prior distribution (black) against true parameters (red). **b** Observation (red) against data (blue) regenerated by running simulations with parameters drawn from the learned distribution — only the first four dimensions are represented. **c** Median distance between regenerated data and the observation as a function of the number of calls to the simulator. Our method does not call the simulator at inference.

Figure 12 is a clear example of a learned distribution that is too tight to contain the ground truth parameters. Nonetheless, the learned distribution learns all the modes induced by the squared operations in Equations 70b and 70c. As with the other toy problems, regenerated data are in strong agreement with the observation. This simulator being higher dimensional than the others, sequential methods need more simulations in order to converge.

Thus, the upfront simulation cost requested by the methods proposed in this dissertation in order to train the surrogate model is quickly amortized.

For the inverse kinematics problem, Figure 13 shows that the learned distribution produce a broad set of arm configurations in order to reach a target point with high accuracy. The regenerated points are highly focused around the target and the average (computed over three runs) reconstructed error is 0.0076 m. These results therefore compete with the best architecture presented in Kruse et al. (2019) where the authors benchmark different neural network architectures to solve inverse problem. Yet, the simulator used in this dissertation is not the same as the one in Kruse et al. (2019) as we introduced stochasticity in the joints, albeit almost negligible.

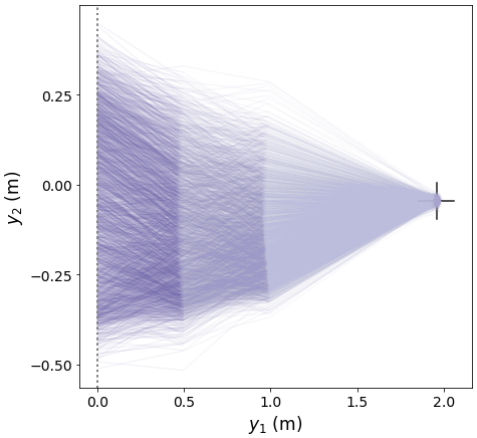


Figure 13: **Inverse kinematics.** Learned prior over the arm configurations given a target observation represented by the cross. The learned distribution do not collapse to a single mode and captures a large set of configurations while reaching the target point with high accuracy.

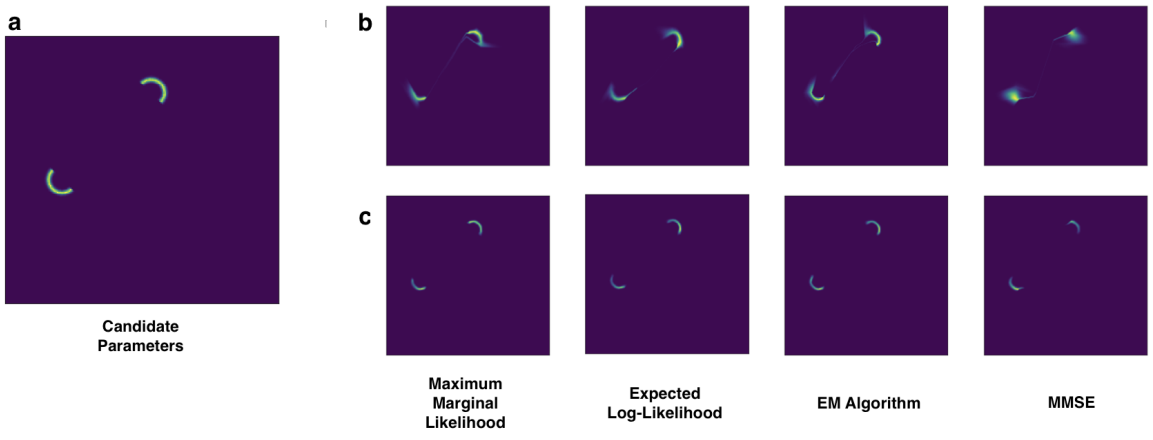


Figure 14: **Two-moons.** Illustration of importance sampling. **a** The manifold of parameters that could generate the observation. **b** Learned distribution for each loss. **c** Learned distributions reweighted by the surrogate likelihood.

In Figure 14 we illustrate importance sampling. Given the learned prior distribution, as we can evaluate the surrogate likelihood, we can reweight the learned distribution by the likelihood. This allows to get properties about the associated posterior distribution. This process is fast and as can be seen in Figure 14, allows to fine tune the learned distribution and only keep parameters that could have generated the observation by assigning them a much higher weight than others. For the expected likelihood and minimum mean squared error (MMSE) losses, results are consistent over multiple runs. For the other methods, one of the two modes is sometimes missed.

6.5 Empirical Bayes

In this section, we study empirical Bayes in a model-free setting in order to learn a distribution over source data. As opposed to traditional methods, we do not assume knowledge about the likelihood function in closed form and we rather approximate the log marginal likelihood by Monte Carlo integration as explained in Section 5.3. That is, we model the prior by a distribution $q_\theta(\mathbf{x})$ and train its parameters by doing gradient descent on stochastic minibatches of observations, with the loss defined in Equation 58. This section is divided in two parts. In the first one, we assume knowledge about the distribution family to which the prior belongs and we optimize the hyperparameters θ of this distribution so that it defines a plausible prior over unseen source data. Then, we go a step further and assume minimal knowledge about the distribution family to which the prior belongs. We only assume that it can be modeled by a neural network density estimator and that it is supported almost everywhere by the proposal prior used to train the surrogate.

Knowledge About the Prior Distribution Family

In this section we assume knowledge about distribution family $\mathcal{Q}$ to which the prior belongs and we try to find the parameters θ such that the candidate $p_\theta(\cdot) \in \mathcal{Q}$ approximates well the distribution over unseen source data. To do so, we learn θ by solving Equation 58.

As the prior distribution $p_\theta(\mathbf{x})$ should be differentiable, we use the reparametrization trick to parameterize it from a known base distribution. When the prior belongs to the family of multivariate gaussians, we learn the parameters $\theta = \{\boldsymbol{\mu}, \mathbf{A}\}$ and sample from $p_\theta(\mathbf{x})$ as:

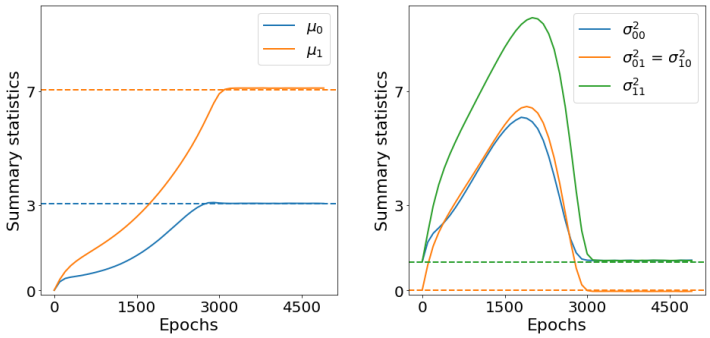
$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\mathbf{T}) \Leftrightarrow \mathbf{x} = \boldsymbol{\mu} + \mathbf{A}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \tag{76}$$

where $\boldsymbol{\Sigma}$ is the covariance of the base density (a multivariate gaussian as well), usually the identity matrix. When the prior belongs to a uniform family, we learn the parameters $\theta = \{\alpha, \beta\}$ and sample from $p_\theta(x)$ as:

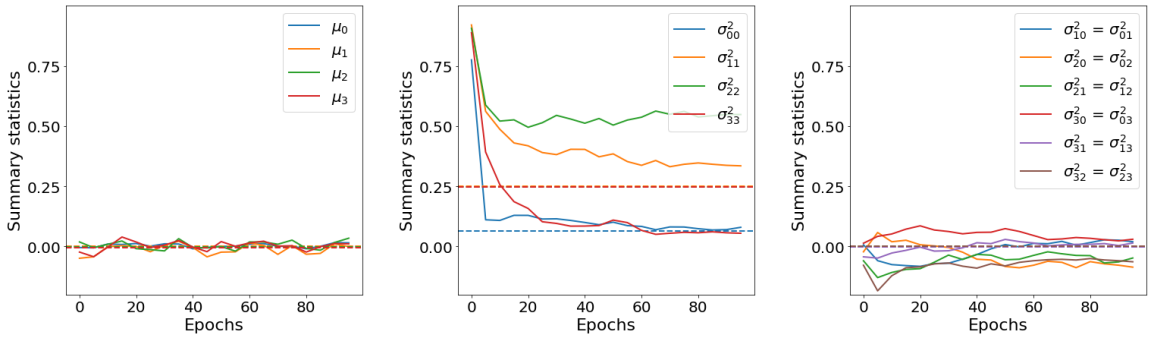
$$x \sim \mathcal{U}(\alpha, \beta) \Leftrightarrow x = (\beta - \alpha)\epsilon + \beta, \quad \epsilon \sim \mathcal{U}(0, 1). \tag{77}$$

In Figure 15, we focus on toy datasets for which the prior is gaussian and plot the evolution of the prior summary statistics during training. That is, we plot the mean $\boldsymbol{\mu}$ and the covariance $\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\mathbf{T}$ as training evolves. We optimize Equation 58 with Adam optimizer with default parameters and stochastic minibatches from the observed data. For the simple gaussian problem, we used 1,000 observed data and we use 512 MC integration steps. Figure 15a shows that around 3,000 epochs, the learned summary statistics converge exactly to the summary statistics of the distribution used to generate source data. Results are consistent over multiple runs. The base density from Equation 76 was chosen to be an isotropic

gaussian and $\boldsymbol{\mu}$ was initialized to $\mathbf{0}$ and $\mathbf{A}$ to the identity matrix. Therefore, at the beginning of training samples are generated completely out of the proposal prior space used to train the surrogate ($\mathcal{N}([3,7]^\top, \mathbb{I})$). Again, this shows that the density estimator has well generalized even out of the proposal prior.



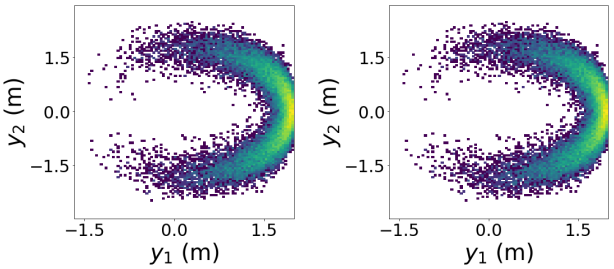
(a) Simple gaussian.



(b) Inverse kinematics.

Figure 15: Evolution of the learned (solid lines) prior summary statistics against the ones of unseen source data (dashed lines) during training.

Figure 15b shows the learned summary statistics during training for the inverse kinematics problem. As the underlying problem is more complex and is higher dimensional, 25,000 observations have been used. For estimating the marginal likelihood integral, 2^{10} MC integration steps are used. It can be observed that the learned summary statistics do not converge as nicely to the summary statistics of the distribution used to generate source data. This is not necessarily bad. Figure 16 shows the learned marginal likelihood density $q(\mathbf{y}|\theta) = \int p(\mathbf{y}|\mathbf{x})q_\theta(\mathbf{x})\mathbf{dx}$ (obtained by sampling parameters from the learned distribution and corrupting them with the simulator) against the empirical distribution of observed data $p(\mathbf{y})$. It can be observed that they closely match — the represented configurations are not induced by the simulator itself but are clearly due to the chosen prior over the source data. Table 3 shows quantitative summary statistics between the two densities over multiple runs. This highlights that the learned marginal likelihood has been well learned and converges towards the expected density but due to the symmetries in the problem definition (Equation 71), we do not necessarily converge to the prior used to generate source data. Even in a perfect setting, with a perfect surrogate and a large number of observations, if the likelihood function has symmetries, without proper regularization, no better solution can be achieved.



(a) Learned distribution. (b) Expected distribution.

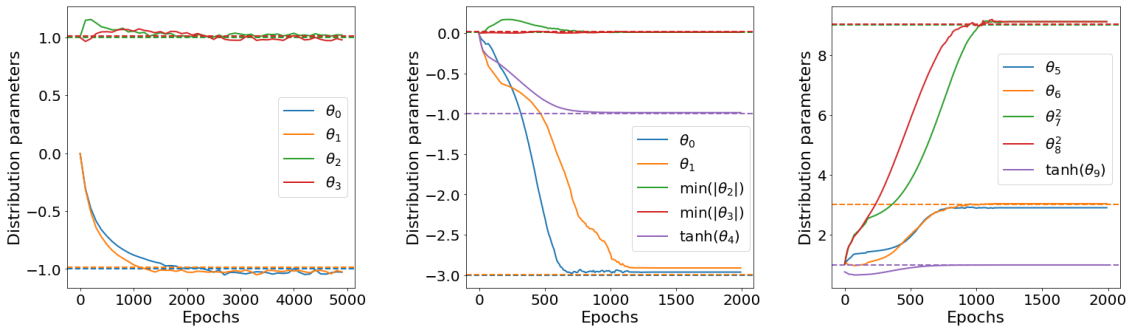
Figure 16: **Inverse Kinematics.** Expected and learned marginal likelihood densities in log-space.

	True statistics	Learned statistics
Means	0.01	0.01 ± 0.01
	1.51	1.44 ± 0.02
Log Variances	0.74	0.78 ± 0.03
	0.24	0.30 ± 0.01

Table 3: **Inverse Kinematics.** Expected and learned marginal likelihood summary statistics over three runs.

Figure 17 shows the evolution of the learned distribution parameters against the ones used to generate source data as training evolves. Figure 17a shows the lower bounds θ_0 and θ_1 and the higher bounds θ_2 and θ_3 of the learned uniform distribution against the ones that were used to generate source data. For this low dimensional problem, 1,000 observations have been used and 512 MC steps have been used to approximate the marginal likelihood. After approximately 2,000 epochs, the parameters converge to the ones that were used to generate source data. Results are consistent over multiple runs.

For the problem with an easy likelihood but complex posterior (SLCP), we do not show directly the lower bounds $\{\theta_0, \theta_1, \theta_2, \theta_3, \theta_4\}$ and higher bounds $\{\theta_5, \theta_6, \theta_7, \theta_8, \theta_9\}$ but the bounds after a change of variable that takes the symmetries from the simulator definition (Equation 70) into account. Indeed, as in the inverse kinematics problem, we do not consistently converge to the parameters that have generated the source data but we converge at least to parameters that minimize the Kullback–Leibler divergence between the observed distribution and the learned marginal likelihood. In other words, we learn the prior up to the symmetries of the problem.



(a) Two-moons. (b) SLCP.

Figure 17: Evolution during training of the learned parameters against the distribution parameters used to generate unseen data.

No Prior Knowledge

In this section we now assume minimal prior knowledge. We model the prior with a density estimator $q_\theta(\mathbf{x})$ and optimize its parameters θ. We model the prior over the SLCP and

two-moons simulator parameters with UMNN-MAFs. Tor the two-moons simulator, we use three layers where the embedding and integrand network both have three layers of 100 units. The prior on the SLCP problem is modeled with seven layers where the integrand and embedding network have 100 units each. For the other simulators, we use four NVP layer where the scale and translation networks are modelled by three-layer MLPs of 16 units with RELU between every two layers. We used 2^{10} MC integration steps for all simulators. We used the Adam optimizer with default parameters with a learning rate of 10^{-3} for the simple gaussian and two-moons simulators. We used a learning rate of 10^{-4} for the SLCP simulator and 10^{-5} for the inverse kinematics simulator.

The training objective defined in Section 5.3 (Equation 58) requires to be able to differentially sample from the density estimator. However, in the way that we have introduced UMNNs in Section 2.4, they do not allow it. Indeed, UMNN-MAFS model the forward process between noise to the variable of interest with a root finding algorithm. While implementing the latter in a differentiable way is technically possible, it may not be straightforward and efficient. Therefore, when we use them in this section, we swap the forward and backward pass. That is, we can model the forward process from noise ϵ to x as:

$$x = f(\epsilon) \tag{78a}$$
$$= \int_0^\epsilon g_\phi(t)dt + f(0). \tag{78b}$$

Then, to evaluate the density of a data point x:

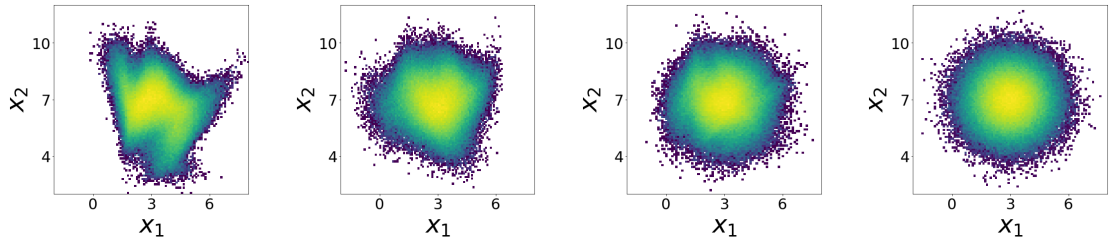
$$p(x) = p(\epsilon)\frac{d\epsilon}{dx} \tag{79a}$$
$$= p(\epsilon)|\det \frac{\partial f^{-1}(x)}{\partial x}| \tag{79b}$$
$$= p(\epsilon)|\det \frac{\partial f(\epsilon)}{\partial \epsilon}|^{-1} \tag{79c}$$
$$= p(\epsilon)\frac{1}{g_\phi(\epsilon)} \tag{79d}$$

where ϵ is obtained by a root-finding algorithm and $g_\phi(\cdot)$ is a neural network with positive outputs that can be quickly evaluated. The swapped model allows to learn a density that can be sampled from efficiently, which is interesting for rejection sampling for example. However, evaluating the density requires to solve a root-finding algorithm. This can be efficiently parallelizable but when this has to be computed multiple times for single points, for example in Markov Chain Monte Carlo (MCMC) samplers, this can be slow and other models would be more appropriated.

Here as well, we suppose that the assumptions required for the learned distribution to converge towards the unseen data distribution are met to some extent. Thus, we assess our method by comparing these two distributions.

Figure 18 shows the evolution of the learned prior against the expected prior as the number of observation increases. It can be seen that when the number of observations is large, the learned density learns well the prior defined in Section 6.1. At the same time, Figure 18a (among others) is not necessarily bad. No regularization have been used and it learns a distribution over $\mathbf{x}$ that explains well the observed data, and nothing more.

Figure 19 shows the learned prior distribution on the two-moons problem. Source data are uniformly distributed over $[-1, 1]^2$. It can be observed that we have remarkably well



(a) 100 observations. (b) 1,000 observations. (c) 10,000 observations. (d) Prior distribution.

Figure 18: **Simple gaussian simulator.** Prior distribution learned by maximum marginal likelihood given 100, 1,000 and 10,000 observations against the prior used to generate source data.

learned the bounds of the uniform prior over $[-1, 1]$ for each $\mathbf{x}_i$. However, the prior is not uniform as it should be. This is explained by the fact that in the simulator definition (Equation 69), $\mathbf{x}_1$ and $\mathbf{x}_2$ are only used through the intermediate variables $|\mathbf{x}_1 + \mathbf{x}_2|$ and $-\mathbf{x}_1 + \mathbf{x}_2$. As already observed in section 6.5, the prior has been learned up to the symmetries of the problem. Indeed, the sum of two uniform distributions follows a Irwin–Hall distribution, or a triangular distribution in 2-D. Figure 19 shows that we have clearly well learned the distribution of the intermediate variables.

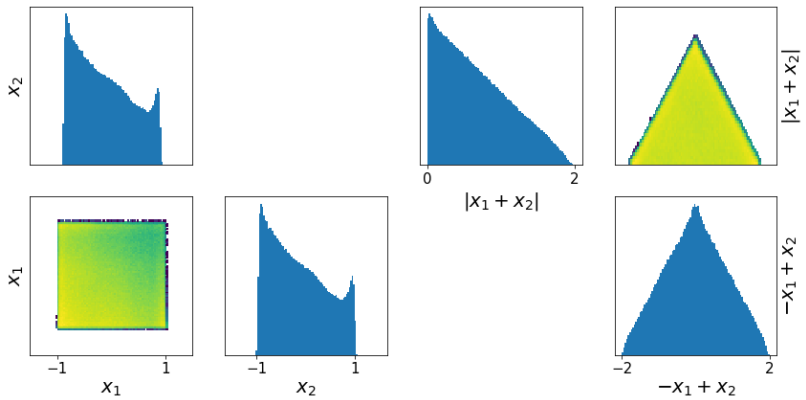


Figure 19: **Two-moons simulator.** Prior distribution learned by maximum marginal likelihood on 10,000 observations. Plots in the diagonals are 1-D histograms while the others are pairwise densities in log-scale. **a** Learned prior distribution over the input variables $\mathbf{x}_1$ and $\mathbf{x}_2$. The expected prior is uniform over $[-1, 1]$ for each $\mathbf{x}_i$. **b** Learned distribution of $|\mathbf{x}_1 + \mathbf{x}_2|$ and $-\mathbf{x}_1 + \mathbf{x}_2$ which is how $\mathbf{x}$ is used in the two-moons simulator. The learned distributions match the expected distributions (triangular distributions).

Figure 20 shows the learned distribution on the problem with simple likelihood and complex posterior (SLCP). Source data are uniformly distributed over $[-3, 3]^5$. For each variable, the learned prior is close to uniform and for the first four variables, the bounds have been nicely learned. For the last variables, the bounds are a bit tighter than expected. In the problem definition, a hyperbolic tangent is applied to $\mathbf{x}_5$. After squashing the variable, the learned bounds are close to the expected bounds and again, we see that without proper regularization we are limited by the symmetries of the problem.

Figure 21 shows the learned prior against the prior used to generate source data on the inverse kinematics problem. As opposed to Section 6.5, the prior converges to the expected prior rather than a prior limited by the symmetries of the problem. Even though the model here is more flexible and less likely to get stuck in a local extremum, the summary statistics over the learned marginal likelihood are not substantially better than the one presented in Table 3 — apart from the log variance over the second dimension that is slightly better. Therefore, both solutions are similar and the convergence to this solution here is more likely due to the network architecture and its initialization.

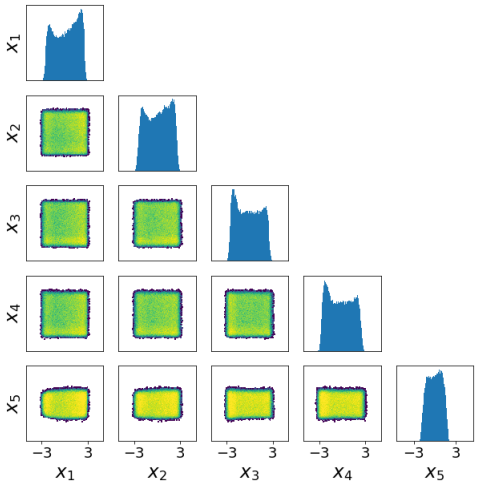


Figure 20: **SLCP simulator.** Prior distribution learned by maximum marginal likelihood on 25,000 observations. Plots in the diagonals are 1-D histograms while the others are pairwise densities in log-scale. The expected prior is uniform over $[-3, 3]$ for each $\mathbf{x}_i$.

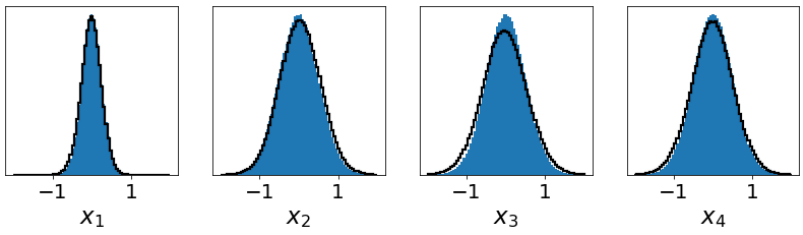
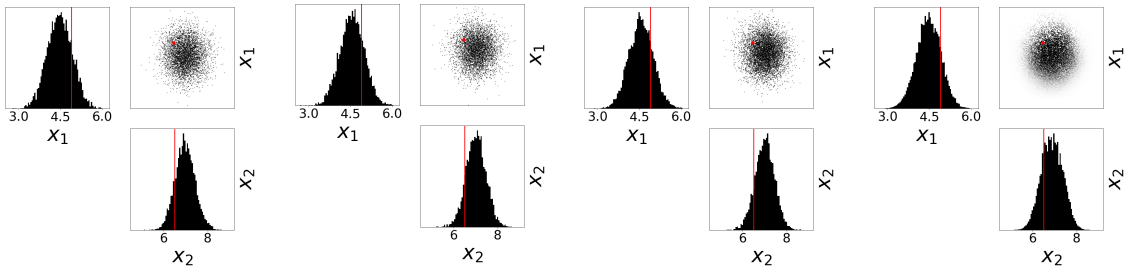


Figure 21: **Inverse Kinematics simulator.** Prior distribution learned by maximum marginal likelihood on 25,000 observations. The learned prior (black) closely match the prior used to generate unseen data (blue).

6.6 Amortized Inference

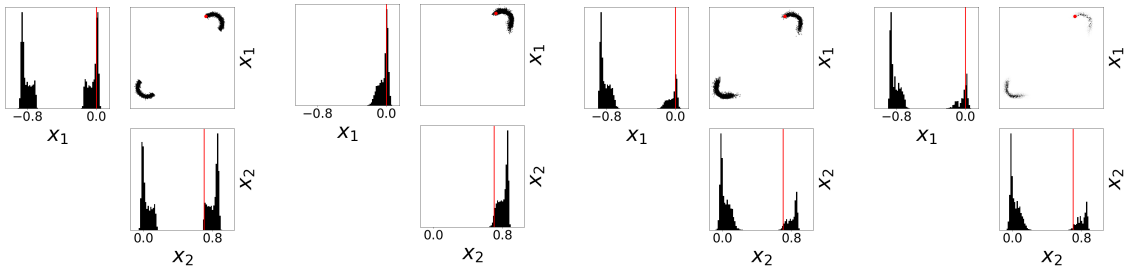
In this section we use the prior learned in the previous section and the surrogate likelihood in order to learn a posterior distribution. Starting with minimal assumptions, we have therefore build tools that now allow us to do Bayesian inference. In this section, we compare Markov Chain Monte Carlo, rejection sampling and importance sampling as defined in Section 5.4. For MCMC, we used Slice Sampling (Neal, 2003) as in Papamakarios et al.

(2019). Variational inference could have been used as well in order to directly learn a model of the posterior. However, this requires to solve an optimization algorithm while the other methods proposed here are not based on optimization, even if MCMC requires careful checks and tuning. Therefore, variational inference has not been used in this section but depending on the end-goal, could be used.



(a) Exact posterior. (b) MCMC. (c) Rejection sampling. (d) Importance sampling.

Figure 22: **Simple gaussian simulator.** Approximated posterior with different methods against the exact posterior. Samples from the exact posterior are sampled from MCMC with the true simulator likelihood and the exact prior distribution over source data. Other methods use the surrogate likelihood and a prior learned with empirical Bayes to do inference. Plots in the diagonals are 1-D histograms while the others are pairwise scatter plots. Red data are the ground truth parameters.



(a) Exact posterior. (b) MCMC. (c) Rejection sampling. (d) Importance sampling.

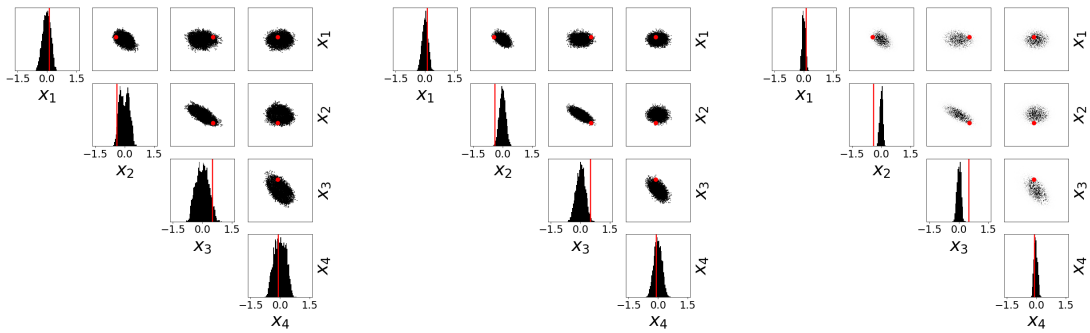
Figure 23: **Two-moons simulator.** Approximated posterior with different methods against the exact posterior which can be directly sampled from. Other methods use the surrogate likelihood and a prior learned with empirical Bayes to do inference. Plots in the diagonals are 1-D histograms while the others are pairwise scatter plots. Red data are the ground truth parameters.

The exact posterior defined by the simulator likelihood and the prior used to generate source data against the posteriors approximated with the different methods are shown for each toy problem in Figures 22, 23, 24 and 25. For the inverse kinematics problem, the simulator likelihood is not known in close-form and therefore, the exact posterior cannot be computed. For the simple gaussian and SLCP simulators, the plotted distribution for each methods are almost indistinguishable between each others and between the true posterior. For the two-moons simulator, the MCMC sampler often fails to capture one of the two modes and similar observations were made in Greenberg et al. (2019). A better tuning of

the sampler or another sampler may solve this issue. For the inverse kinematics simulator, the distribution learned by importance sampling is tighter than the others.

Importance sampling is without a doubt the fastest method used here. It only needs to sample points from the prior and weight them by their likelihood. Even for millions of data points, this is done in a few seconds on GPU. In Table 5 we show the approximated posterior expectation against the true empirical expectation. For the simple gaussian problem, the expectation is approximated with high accuracy. For the two-moons and SLCP simulators, the accuracy is not as good. Nonetheless, for the SLCP toy problem, the estimated values are clearly an approximation of the true expectation.

Importance sampling allows to get properties about the target distribution but is not designed to sample from the distribution. In order to sample from the posterior, we therefore use rejection sampling or MCMC. Rejection sampling performs particularly well in this context. As explained in section 5.4, the process can be parallelizable and in order to generate the data points plotted in Figures 22, 23, 24 and 25 (5000 points per plot), it takes between a few seconds to a few minutes for the higher dimensional simulator. On the other hand, Markov chain Monte Carlo is much slower and takes dozens of minutes to several hours. Moreover, rejection sampling only needs to sample from the prior distribution while MCMC needs to evaluate its density. Table 4 shows the Maximum Mean Discrepancy (MMD) Gretton et al. (2012) between the exact and approximated posteriors. Results are good and similar between rejection sampling and MCMC apart on the two-moons simulator where MCMC is much worse than rejection sampling due to the fact that MCMC fails to consistently get the two modes.



(a) MCMC. (b) Rejection sampling. (c) Importance sampling.

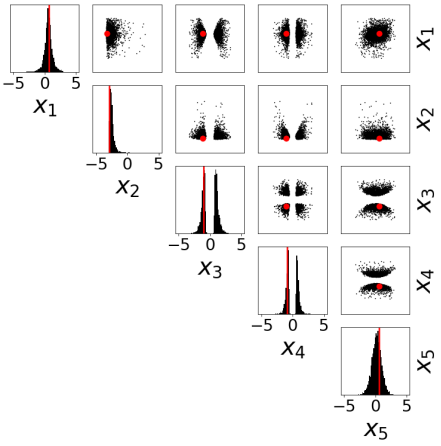
Figure 24: **Inverse kinematics simulator.** Approximated posterior with different methods that use the surrogate likelihood and a prior learned with empirical Bayes to do inference. Plots in the diagonals are 1-D histograms while the others are pairwise scatter plots. Red data are the ground truth parameters.

	MCMC	Rejection sampling
Simple gaussian	0.005 ± 0.001	0.005
Two-moons	0.380 ± 0.001	0.090 ± 0.010
SLCP	0.025 ± 0.001	0.022 ± 0.002

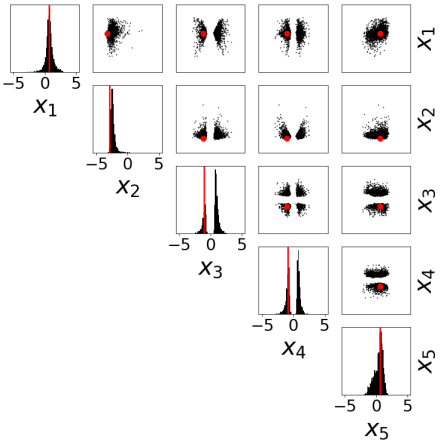
Table 4: Comparison of the Maximum Mean Discrepancy between the true posterior and the posterior approximated by Markov Chain Monte Carlo and rejection sampling. The Maximum Mean Discrepancy is computed with a gaussian kernel.

	Posterior mean	Approximated mean
Simple gaussian	$[4.48, 6.98]^\top$	$[4.48, 6.88]^\top$
Two-moons	$[-0.42, 0.43]^\top$	$[-0.61, 0.23]^\top$
SLCP	$[0.61, -2.46, 0.01, -0.07, 0.22]^\top$	$[0.75, -2.33, 0.30, -0.01, 0.40]^\top$

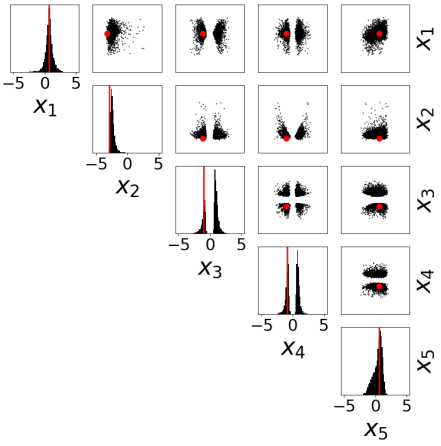
Table 5: Empirical posterior expectation against its approximation by importance sampling. When it is not possible to sample directly from the posterior, we generate samples from it with Markov Chain Monte Carlo with the true simulator likelihood and the exact distribution over source data.



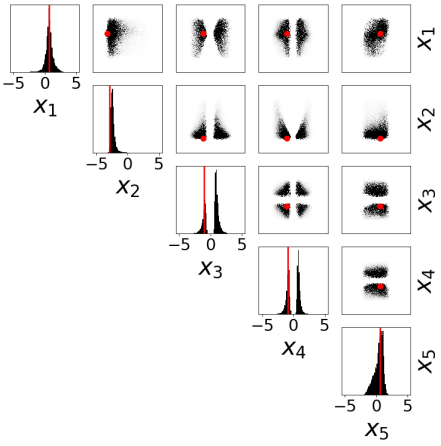
(a) Exact posterior.



(b) MCMC.



(c) Rejection sampling.



(d) Importance sampling.

Figure 25: **SLCP simulator.** Approximated posterior with different methods against the exact posterior. Samples from the exact posterior are sampled from MCMC with the true simulator likelihood and the exact prior distribution over source data. Other methods use the surrogate likelihood and a prior learned with empirical Bayes to do inference. Plots in the diagonals are 1-D histograms while the others are pairwise scatter plots. Red data are the ground truth parameters.

7. Detector Effects Correction in High Energy Physics

7.1 Problem Statement

In experimental particle physics, the measurement process follows a hierarchical process similar to those discussed in previous sections. The theory of fundamental particles and their interactions, the Standard Model (SM), can predict with high accuracy the outcomes of particle interactions, but relies on 19 free parameters. At a collider like the Large Hadron Collider (LHC), high energy particle collisions produce an emanating spray of particles which are measured, with finite precision, in detector devices. Thus, the Standard Model can predict a prior particle distribution but it is only measured by a noisy detector device (the detector response can be considered independent of the parameters of the Standard Model). A major goal in such experiments is therefore to deconvolve the effects of the detector. From individual particle measurements, the aim is to access the "true" particle energy to use in downstream analysis and to measure the distribution of particle properties for comparison with the Standard Model predictions.

The LHC produces proton-proton collisions at approximately 40 MHz. Protons and neutrons are the most famous hadrons, they are made of quarks held together by strong force interactions. According to the Standard Model (SM) of particle physics, quarks are one of the building blocks of our universe. These building blocks are elementary particles classified in different categories (Figure 26).

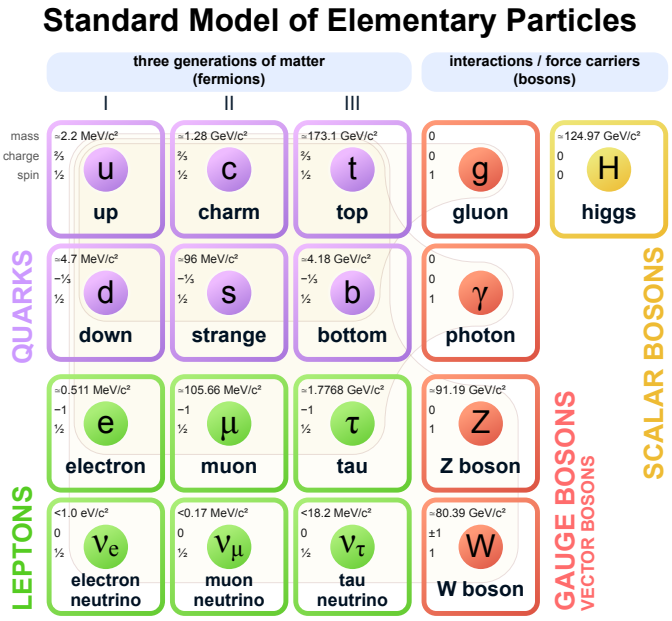


Figure 26: Elementary particles defined by the Standard Model (Commons, 2020).

In order to test different theories, search for new physics and study their properties, elementary particles are created in particle accelerators. For example, collisions at the LHC have sufficient energy to create heavy short lived particles, including those defined by the Standard Model such as the Higgs Boson and potentially new heavy particle beyond the Standard Model (BSM) theory. Once created, such heavy particles are unstable and decay to other elementary particles on extremely short time scales, for example the Higgs boson lifetime is $\mathcal{O}(10^{-22}s)$. Therefore, these particles cannot be observed directly but only a

by-product of their formation is observed. For example, the Higgs boson was discovered in 2012 (Collaboration, 2012) by individual searches into its decay into Z, photon and W pairs that are elementary particles as well. The intermediate produced particles may also have short lifetimes. Thus, they will themselves decay and the newly produced particles will decay again. This cascade process will continue until a stable particle that cannot decay (at least on the timescale of it traversing the detector) is produced. In the case of quarks, they cannot exist as isolated particles and when produced at high energy will radiate energy and form hadrons, thus producing a collimated spray of particles known as a jet. The particles of this jet then emanate outward and their energy and direction are measured by calorimeter detectors. In order to identify a jet in a calorimeter, a clustering algorithm is used to cluster the energy depositions.

Let us illustrate these concepts with a typical proton-proton collision. After a collision at high energy, a proton-proton collision may produce elementary particles after a process called scattering. Among others, the collision may produce a Z hand Higgs Boson:

$$pp \to ZH. \tag{80}$$

Z and Higgs Boson have respectively mean lifetimes of about 3×10^{-25}s and 1.56×10^{-22}s. Thus, they will themselves decay into other particles. Z boson may decay into charged lepton-antilepton (leptons are elementary particles that are not subjected to the strong force) pairs such as an electron and a positron. The H boson may decay for example, into two bottom quarks. This is represented in Figure 27.

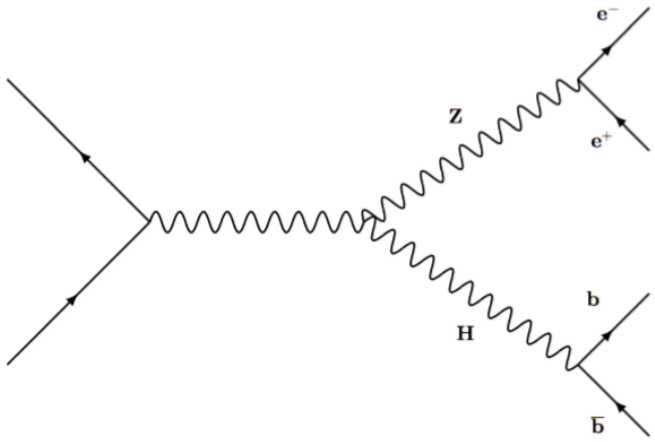


Figure 27: Feynman diagram where a Z and Higgs boson are produced. The former decays into a positron and an electron. The later decays into bottom quarks.

The bottom quarks will radiate energy and form hadrons. The produced hadrons will deposit energy in the calorimeters of particle detectors and a clustering algorithm will produce jets associated to the original bottom quarks. This process is represented in Figure 28. Then, these jets have different attributes such as mass, momentum and rapidity that can be studied.

A standard way to simulate the collision process is to use Monte Carlo simulators such as Pythia (Sjöstrand et al., 2006; ATL, 2014; Sjöstrand et al., 2015) or Herwig (Bähr et al., 2008). Then, the smearing through the detector should be simulated as well. GEANT Brun et al. (1994) is used to simulate realistic particle interactions with material. As this

is a highly computationally expensive process, sometimes a simplified model of particle interactions with material is used, for example the Delphes software (de Favereau et al., 2014).

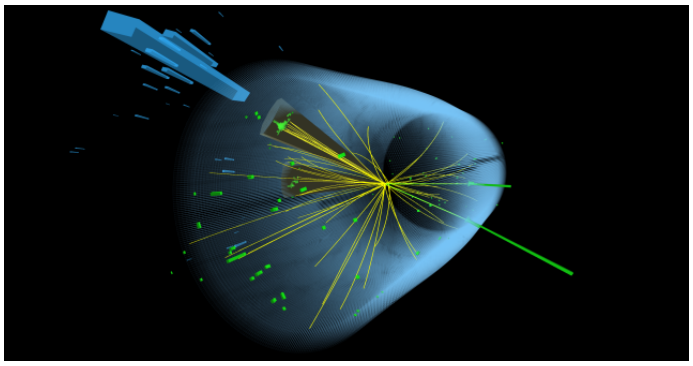


Figure 28: Display of an event observed in the CMS detector in which a Higgs boson decays to bottom quarks (Mc Cauley, 2018). The event produces a Z and Higgs boson. The former decays into a positron and an electron represented by the two green lines. The later decays into bottom quarks that form jets through radiating energy and then hadronizing. These jets are represented by the two cones while the green towers represent energy deposits in the electromagnetic calorimeter.

Here, given measured jet properties (hence corrupted), we aim at retrieving source properties. We study four of them. The first one is the jet mass m. In relativity, the particle with energy E and momentum $\mathbf{p} = m\gamma\mathbf{v}$ (where γ is the Lorentz factor and v is the velocity of the particle) is represented as the energy-momentum four-vector:

$$\mathbf{P} = [\frac{E}{c}, p_x, p_y, p_z]^\top.$$

The jet mass is computed as:

$$m = \sqrt{(\frac{E}{c}\frac{E}{c} - p_x p_x - p_y p_y - p_z p_z)}. \tag{81}$$

where the jet four-vector is obtained by summing the four-vectors of all the particles that belong to the jet:

$$\mathbf{P} = [\frac{E}{c} f, p_x, p_y, p_z]^\top = \sum_{i \in jet} \mathbf{P}_i. \tag{82}$$

Equation 81 is obtained by the energy-momentum relation:

$$E^2 = (mc)^2 + (pc)^2 \tag{83}$$

where c is the speed of light and p, the momentum magnitude.

We also study the groomed mass after Soft Drop (Larkoski et al., 2014a) which is a jet grooming technique. Jet grooming are post-processing methods that aim at cleaning a jet from constituents that are likely to come from contamination (Marzani et al., 2017). For

example, at the LHC, multiple collisions happen at the same time[11] (pile-up) and thus, in a given jet, some of the observed data might not come from the collision of interest.

We also study the jet width w that belongs to a family that defines jet shapes called generalized angularity. For example, generalized angularities are effective in discriminating quarks and gluons because they probe the energetic and angular structure of the jet (Larkoski et al., 2014b). Finally, we also study the 2-subjettiness value with $\beta = 1$. Similarly to the jet width, the N-subjettiness is a jet shape. It is designed to identify boosted hadronic objects such as top quarks (Thaler and Van Tilburg, 2012).

We use the dataset (Andreassen et al., 2019a) of pairs of source and corrupted data made publicly available in Andreassen et al. (2019b). Proton-proton collisions are studied at $\sqrt{s} =$ 14 Tev. Among the different datasets made available, we use source data generated with the Monte Carlo simulator Pythia 8.243 (Sjöstrand et al., 2015) with Tune 26 (ATL, 2014). Data are corrupted with a fast simulation of the CMS detector, Delphes 3.4.2 (de Favereau et al., 2014). We use 60 % of the dataset to train the surrogate, 20% as validation set to tune the inference algorithm and 20% as test set to report inference performance. We also assess performance on the dataset where source data are generated by Herwig 7.1.5 (Bähr et al., 2008; Bahr et al., 1999) with default tune and corrupted with Delphes detector simulator. All the dataset is used to only report inference performance. This is a challenging setting where the distribution used to train the surrogate is substantially different than the distribution on which inference is done. Both datasets contain approximately 1.6 million events.

7.2 Related Work

Since D'Agostini (1995), Iterative Bayesian Unfolding (IBU) has been a widely used technique to correct detector effects. Source and corrupted data are usually binned into 1-D histograms with respectively N_E and N_C bins. Therefore, equation:

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\mathbf{dx} \tag{84}$$

is rewritten for each dimension as:

$$P(y_j) = \sum_{l=1}^{N_C} P(y_j|x_l)P(x_l) \tag{85}$$

where $P(y_j)$ is the probability assigned to the j^{th} bins. IBU requires an approximation of the likelihood $P(y_j|x_l)$, approximated with pairs of source and corrupted data. This quantity is approximated by computing the 2-D joint histogram $\frac{P(y,x)}{P(x)}$ for 1-D variables y and x. Therefore, as opposed to us, IBU only treat one variable at a time and correlation between variables are lost. Given an approximation of the likelihood, IBU iteratively learns

11. In order to collide particles, two beams circulating in opposite directions are concentrated in a small section. Due to the size of protons, and the limited size of the section, the probability that two protons collide is small. Therefore, each beam contains a large number of protons and thus, several collisions happen at the same time.

a 1-D binned prior distribution and update the probability assigned to each bin as:

$$P_{n+1}(x_i) = \sum_{j=1}^{N_E} P(x_i|y_j)P(y_j) \tag{86a}$$

$$= \sum_{j=1}^{N_E} \frac{P(y_j|x_i)P_n(x_i)}{\sum_{l=1}^{N_C} P(y_j|x_l)P_n(x_l)} P(y_j) \tag{86b}$$

where the first equation is obtained by marginalizing x and the second line is obtained using the Bayes rule. The initial values $P_0(x_i)$ are initialized given prior knowledge and uniformly in case of complete ignorance.

Recently, Andreassen et al. (2019b) used machine learning to generalize Equation 86 by replacing the sum by a full phase-space integral:

$$p_{n+1}(\mathbf{x}) = \int \frac{p(\mathbf{y}|\mathbf{x})p_n(\mathbf{x})}{\int p(\mathbf{y}|\mathbf{x})p_n(\mathbf{x})\mathbf{dx}} p(\mathbf{y})\mathbf{dy}. \tag{87}$$

The idea is to use a dataset of corrupted and uncorrupted data in the same way that IBU uses pairs of corrupted and uncorrupted data to approximate the likelihood or in the same way that we need a dataset to train a surrogate model. Then, they learn a function $\nu(\mathbf{x})$ such that the prior distribution $p(\mathbf{x})$ can be approximated by reweighing the dataset distribution of source data $p_{data}(\mathbf{x})$:

$$p(\mathbf{x}) = \nu(\mathbf{x})p_{data}(\mathbf{x}). \tag{88}$$

Thus, Equation 88 can be plugged in Equation 87 which provides an iterative strategy to update the function $\nu(\cdot)$. Yet, they do not need to approximate the likelihood function nor the integral function defined in Equation 87. They use the likelihood ratio in order to iteratively learn a function that reweights the corrupted data from the dataset to the observed corrupted data. Then, they use this function to learn the function $\nu(\cdot)$. As opposed to them, we do not learn a weighting function. We learn a prior distribution that can be sampled from, whose density can be evaluated and differentiated.

7.3 Surrogate Model

In order to define the surrogate model, a grid search was performed over hyperparameters. This grid search showed similar performance between different architectures. Therefore, we used the same architecture than the one used on toy problems defined in Section 6.2. Training was done with Adam optimizer with a learning rate of 10^{-4} and no weight decay regularization for 300 epochs over the dataset. As the 1-D marginal distributions over the variables of interest have long tails, we preprocessed the variables by taking their logarithm. No further preprocessing was done.

Figure 29 shows the (empirical) learned corrupted distribution against the target dataset distribution on the test set.

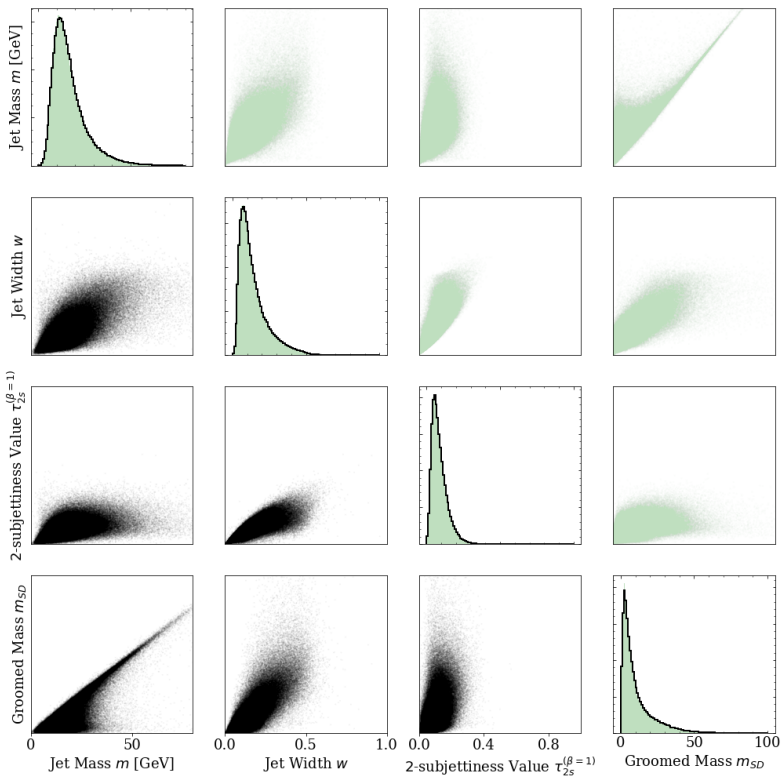


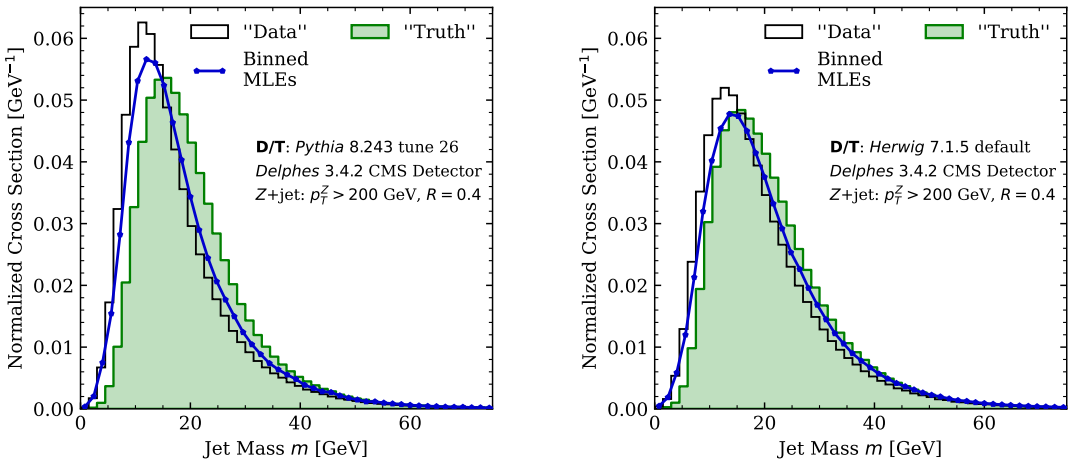
Figure 29: Empirical distributions of corrupted data from the test set (in green) against their associated source data corrupted by the trained surrogate (in black). The x and y-axes correspond to different jet properties. The diagonals show the 1-D marginal distributions while others are pairwise scatter plots. The surrogate model approximates well the detector corruption process.

7.4 Empirical Prior from Point Estimates

In Section 6.3, we showed that given a large number of observations $\{\mathbf{y}_1, ..., \mathbf{y}_N\}$ we can learn the most likely estimate associated to each observation and bin them to approximate the source data distribution. Here, we show that despite simple, this approach can be truly efficient even on real, complex problems.

The experiment setup is the same as the one introduced in Section 6.3. As defined by Equation 73, we aim at maximizing the log likelihood of each observation $\mathbf{y}_i$ by optimizing individual unseen parameters $\mathbf{x}_i$. As in Section 6.3, we use the Adam optimizer for its learning rate adaptability and to parallelize the optimization algorithm. We used the default parameters of Adam optimizer because it performed well on the Pythia validation set. For this proof of concept, few other hyperparameter settings were tested. We initialize each $\mathbf{x}_i$ to the corrupted data $\mathbf{y}_i$, because we assume that the source data should not be too far away from the corrupted data and then, solve a maximum likelihood estimation problem for each $\mathbf{y}_i$.

In Figure 30, we report the learned empirical prior on Pythia and Herwig test sets that contain respectively about 300,000 and 1,6 millions observations. Binning is done in 1-D for each jet property by defining 100 equally-likely bins in the range of values taken by the data. We insist on the fact that binning is done after the optimization process. Therefore, we could bin in multiple dimensions as well. It can be observed that on both datasets, a first correction step has been nicely done. Given observed data represented by the black curve, the learned blue curve is a better approximation of the source distribution. While results are interesting for this jet property, the correction step is not as big on other jet properties. Therefore, this simple technique can be embedded in a more complex one, for example to initialize the weights of the prior distribution in IBU or in empirical Bayes.



(a) Pythia dataset.

(b) Herwig dataset.

Figure 30: Empirical prior approximated by binning most likely estimates (MLEs) of unseen jet properties (truth) where MLEs are obtained by solving a maximum likelihood problem for each measurement (data).

7.5 Detector Effects Correction with Empirical Bayes

In this section, we learn an approximation of the source data distribution, that we model by a density estimator. As defined in Section 5.3, we maximize the log marginal likelihood. In order to do so, we evaluate the log marginal likelihood and optimize it on stochastic minibatches of size 256. In order to approximate the marginal likelihood integral, we use Monte Carlo integration with 2^{10} integration steps. We use Adam optimizer with default parameters and a learning rate of 10^{-4}. We train for 10 epochs over the whole Herwig dataset. Finally, the density estimator modelling the prior distribution is made of 6 NVP layers where the functions $s(\cdot)$ and $t(\cdot)$ are MLPs with three layers of 32 units each and RELU between every two layers. These parameters performed best after a grid search over hyperparameters on the Pythia validation set.

In Figure 31, we compare our results to IBU and Multifold. Multifold is a version of omnifold where jet properties are used rather that the raw particles at detector level. Results for these methods as well as the layout used to make Figures 30 and 31 have been obtained with the public implementation[12] of Andreassen et al. (2019b). For IBU and

12. https://github.com/ericmetodiev/OmniFold

Multifold, we used the same hyperparameters than them. We also report the two-sample Kolmogorov–Smirnov test for each jet property in Table 6.

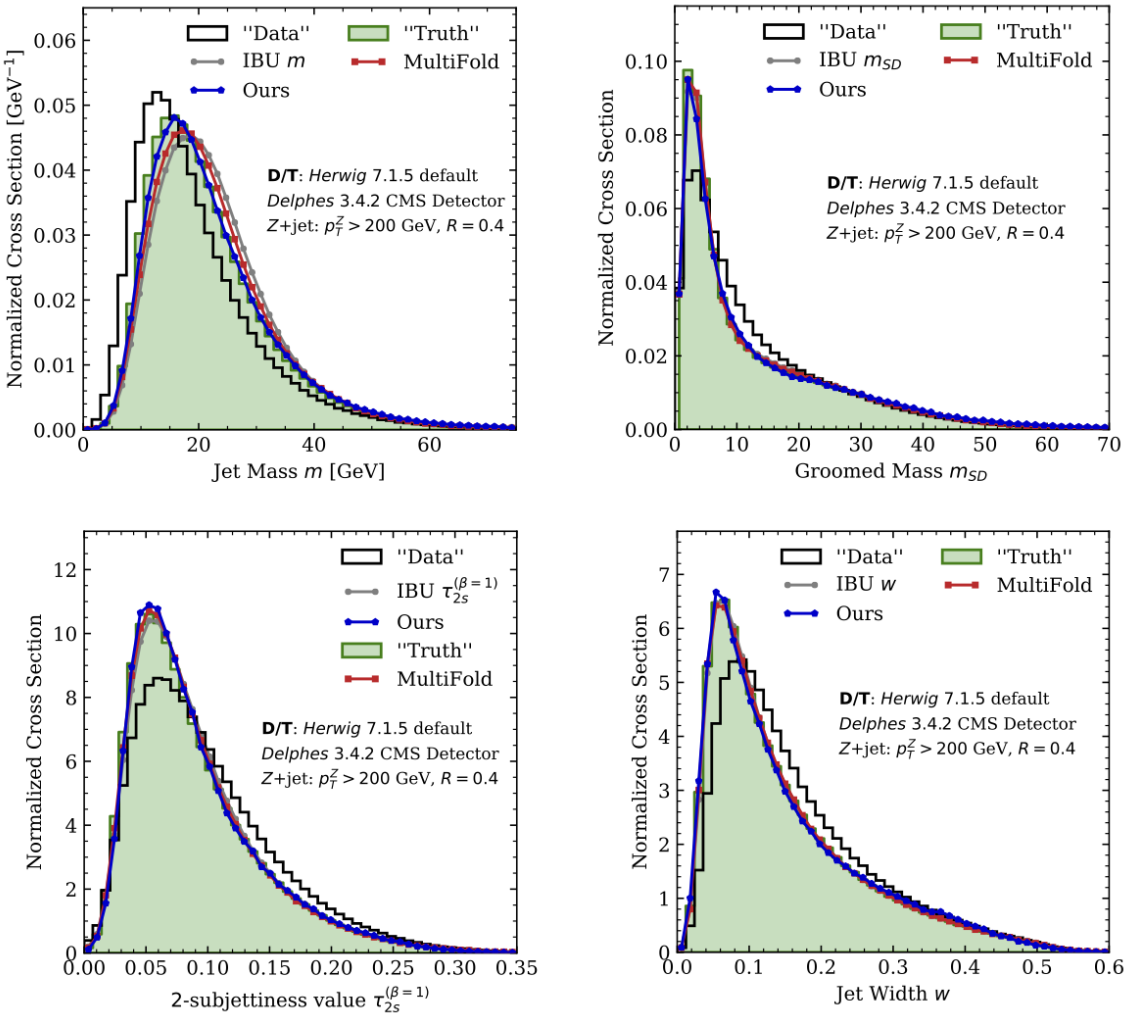


Figure 31: Detector effects correction on four jet substructure observables. Source data (truth) have been generated with Herwig simulator and then corrupted (data) with Delphes simulator. We compare our results against IBU and Multifold. All methods perform well with our method being slightly better in correcting the mass property.

The mass is challenging to learn and the likelihood-free empirical Bayes method consistently does better than IBU and Multifold. For the other properties, it incredibly well learns the source distribution even if it performs a bit worse than IBU and Multifold quantitatively. As opposed to IBU and Multifold, the likelihood-free empirical Bayes method learns a prior distribution that can be sampled from, whose density can be evaluated and differentiated. This should open new perspectives in downstream analysis.

	m	m_{SD}	$\tau_{2s}^{\beta=1}$	w
IBU	0.0839	0.004	0.022	0.021
Multifold	0.054 ± 0.001	0.003 ± 0.001	**0.009 ± 0.001**	**0.016 ± 0.001**
Ours	**0.038 ± 0.012**	0.010 ± 0.003	0.028 ± 0.012	0.032 ± 0.005

Table 6: Detector effects correction performance for IBU, Multifold and our method. Performance are quantified using the two-sample Kolmogorov–Smirnov test: $D_{p,q} = \sup_x |F_p(x) - F_q(x)|$ where F_p and F_q are the binned empirical distribution of the truth-level and corrected histograms. We show the mean and standard deviation over 5 runs for Multifold and our method while IBU algorithm has no stochasticity. All methods perform well.

8. Conclusion

This works leverages the recent advances in generative modelling and especially normalizing flows to approximate the otherwise intractable likelihood of simulators. Being able to approximate and differentiate the likelihood function allows to use traditional methods such as maximum likelihood and posterior inference. We show how, with minimal assumptions, we can learn a prior over the model parameters. Then, the learned prior can be used with the approximated likelihood to do traditional Bayesian inference. Therefore, after an upfront data acquisition phase and surrogate training, inference is later amortized.

Hence, we develop a pipeline than learns a prior and then use it to do inference with minimal assumptions and we show its applicability on a real high energy physics problem. This opens new perspectives for methods that require less and less domain knowledge.

In this work, the learned prior distributions have been consistently modeled by normalizing flows, which act as regularizers by themselves. A strength of our method is that the prior can be modeled by any generative model. Therefore, for future work it is interesting to investigate stronger regularizations that do not specifically require more domain knowledge. For example, convolutional neural network are strong regularizers that constrain data in the space of natural-like images.

Appendix A. Maximum Likelihood as a Generalization of Minimum Squared Error

In this appendix we show that anyone who is solving a constrain problem of the type:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} ||\mathbf{A}\mathbf{x} - \mathbf{y}||^2, \quad s.t. \quad \mathbf{x} \in \mathcal{S} \tag{89}$$

is implicitly solving a maximum likelihood problem. This generalizes to unconstrained problems.

Equation 89 can rewritten as:

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} ||\mathbf{A}h(\mathbf{z}) - \mathbf{y}||^2, \quad \mathbf{x}^* = h(\mathbf{z}^*) \tag{90}$$

where $h(.)$ enforce $\mathbf{x}$ to be in the set $\mathcal{S}$.

Solving a maximum likelihood problems aims at solving:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}), \quad s.t. \quad \mathbf{x} \in \mathcal{S} \tag{91}$$

or equivalently:

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} \log p(\mathbf{y}|h(\mathbf{z})), \quad \mathbf{x}^* = h(\mathbf{z}^*). \tag{92}$$

Assuming that $p(\mathbf{y}|\mathbf{x}) \sim \mathcal{N}(\mathbf{A}\mathbf{x}, \mathbf{\Sigma}^2)$ where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the matrix defined in Equation 89 and $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is the identity matrix multiplied by an arbitrary constant, Equation 92 can be rewritten as:

$$\begin{aligned}
\mathbf{z}^* &= \arg\max_{\mathbf{z}} \ln \frac{1}{\sqrt{(2\pi)^n |\mathbf{\Sigma}|}} \exp(-\frac{1}{2}(\mathbf{y} - \mathbf{A}h(\mathbf{z}))^T \mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{A}h(\mathbf{z}))) && \text{(93a)} \\
&= \arg\max_{\mathbf{z}} \ln \exp(-\frac{1}{2}(\mathbf{y} - \mathbf{A}h(\mathbf{z}))^T \mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{A}h(\mathbf{z}))) + \cancel{cst} && \text{(93b)} \\
&= \arg\max_{\mathbf{z}} \ln \exp(-\frac{1}{2}\mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{A}h(\mathbf{z}))^T (\mathbf{y} - \mathbf{A}h(\mathbf{z}))) && \text{(93c)} \\
&= \arg\max_{\mathbf{z}} -\frac{1}{2}\mathbf{\Sigma}^{-1}(\mathbf{y} - \mathbf{A}h(\mathbf{z}))^2 && \text{(93d)} \\
&= \arg\max_{\mathbf{z}} ||\mathbf{A}h(\mathbf{z}) - \mathbf{y})||_2^2 && \text{(93e)}
\end{aligned}$$

In the first line, we use the definition of the multivariate gaussian distribution. Then, in the second line, we use the hypothesis that the covariance matrix is independent on $\mathbf{z}$ and therefore, we remove it from the optimization problem. In the third line, we perform a commutativity that may be performed because $\mathbf{\Sigma}$ is diagonal. In (93d) we remove the identity operator and finally, in the last line we remove terms that are independent on $\mathbf{z}$. The norm in the last line derives from the fact that $\mathbf{\Sigma}$ is a constant diagonal matrix.

Equation 93e is the same to Equation 90 proving that maximum likelihood is a generalization of solving a minimum mean squared error. Note that this generalizes to nonlinear inverse problems where $\mathbf{A}h(\mathbf{z})$ should be replaced by $s(h(\mathbf{z}))$ where $s(.)$ is a deterministic nonlinear corruption process.

Before concluding let us dive into the assumption made about the likelihood function, what it implies and therefore what assumptions are made when solving a problem with a least-squared error loss.

Assuming that conditioned on $\mathbf{x}$, the likelihood follows a normal distribution assume that in stochastic processes, given a fixed $\mathbf{x}$, the produces $\mathbf{y}'s$ are distributed given a multivariate gaussian with an arbitrary complex mean $A\mathbf{x}$ or $s(\mathbf{x})$, a fixed variance along each dimension and with no correlations between dimensions. Using a weighted squared error is equivalent to Equation 93d where $\mathbf{\Sigma}$ is an arbitrary diagonal matrix and therefore, relax the assumption that the variance along each dimension is the same.

References

ATLAS Pythia 8 tunes to 7 TeV datas. Technical Report ATL-PHYS-PUB-2014-021, CERN, Geneva, Nov 2014. URL https://cds.cern.ch/record/1966419.

Anders Andreassen, Patrick Komiske, Eric Metodiev, Benjamin Nachman, and Jesse Thaler. Pythia/Herwig + Delphes Jet Datasets for OmniFold Unfolding, November 2019a. URL https://doi.org/10.5281/zenodo.3548091.

Anders Johan Andreassen, Patrick T. Komiske, Eric M. Metodiev, Benjamin Nachman, and Jesse Thaler. Omnifold: A method to simultaneously unfold all observables. *arXiv: High Energy Physics - Phenomenology*, 2019b.

Lynton Ardizzone, Jakob Kruse, Sebastian J. Wirkert, Daniel Rahner, Eric W. Pellegrini, Ralf S. Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *CoRR*, abs/1808.04730, 2018. URL http://arxiv.org/abs/1808.04730.

Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *CoRR*, abs/1907.02392, 2019. URL http://arxiv.org/abs/1907.02392.

Manuel Bahr, S. Gieseke, M. A. Gigg, David Grellscheid, K. Hamilton, O. Latunde-Dada, Simon Platzer, P. Richardson, Mike H. Seymour, A. Sherstnev, and Bryan Webber. Herwig++ 2.1 release note. *IEEE Software*, 1999.

R. Bellman and Karreman Mathematics Research Collection. *Adaptive Control Processes: A Guided Tour*. Princeton Legacy Library. Princeton University Press, 1961. URL https://books.google.be/books?id=POAmAAAAMAAJ.

C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006. ISBN 9780387310732. URL https://books.google.be/books?id=qWPwnQEACAAJ.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G. Dimakis. Compressed sensing using generative models, 2017.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004. ISBN 0521833787.

Johann Brehmer, Kyle Cranmer, Gilles Louppe, and Juan Pavez. A guide to constraining effective field theories with machine learning. *Physical Review D*, 98(5), Sep 2018. ISSN 2470-0029. doi: 10.1103/physrevd.98.052004. URL http://dx.doi.org/10.1103/PhysRevD.98.052004.

Rene Brun, László Urbán, Michel le Maire, Federico Carminati, Simone Moretti Giani, Flavienne Bruyant, Garda Patrick, and A. C. McPherson. Geant : detector description and simulation tool. 1994.

Manuel Bähr, Stefan Gieseke, Martyn A. Gigg, David Grellscheid, Keith Hamilton, Oluseyi Latunde-Dada, Simon Plätzer, Peter Richardson, Michael H. Seymour, Alexander Sherstnev, and et al. Herwig++ physics and manual. *The European Physical Journal C*, 58

(4):639–707, Nov 2008. ISSN 1434-6052. doi: 10.1140/epjc/s10052-008-0798-9. URL http://dx.doi.org/10.1140/epjc/s10052-008-0798-9.

George Casella. An introduction to empirical bayes data analysis. *The American Statistician*, 39(2):83–87, 1985. ISSN 00031305. URL http://www.jstor.org/stable/2682801.

Kiran Chakravarthula. *Study of Jet Transverse Momentum and Jet Rapidity Dependence on Dijet Azimuthal Decorrelations*. PhD thesis, Louisiana Tech. U., 2012.

The Atlas Collaboration. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. 2012.

Wikimedia Commons. File:standard model of elementary particles.svg — wikimedia commons, the free media repository, 2020. URL https://commons.wikimedia.org/w/index.php?title=File:Standard_Model_of_Elementary_Particles.svg&oldid=422112175. [Online; accessed 31-May-2020].

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences of the United States of America*, 2020.

Nicholas Dagalakis. *Industrial Robotics Standards*, pages 447 – 459. 11 2007. ISBN 9780470172506. doi: 10.1002/9780470172506.ch24.

G. D'Agostini. A multidimensional unfolding method based on bayes' theorem. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 362(2):487 – 498, 1995. ISSN 0168-9002. doi: https://doi.org/10.1016/0168-9002(95)00274-X. URL http://www.sciencedirect.com/science/article/pii/016890029500274X.

J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. Delphes 3: a modular framework for fast simulation of a generic collider experiment. *Journal of High Energy Physics*, 2014(2), Feb 2014. ISSN 1029-8479. doi: 10.1007/jhep02(2014)057. URL http://dx.doi.org/10.1007/JHEP02(2014)057.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. doi: 10.1111/j.2517-6161.1977.tb01600.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1977.tb01600.x.

Peter J. Diggle and Richard J. Gratton. Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46 (2):193–227, 1984. ISSN 00359246. URL http://www.jstor.org/stable/2345504.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ArXiv*, abs/1605.08803, 2017.

Conor Durkan, Iain Murray, and George Papamakarios. On contrastive learning for likelihood-free inference. *ArXiv*, abs/2002.03712, 2020.

Bradley Efron. Bayes, oracle bayes and empirical bayes. *Statist. Sci.*, 34(2):177–201, 05 2019. doi: 10.1214/18-STS674. URL https://doi.org/10.1214/18-STS674.

Mario J. Gonzalez, Andrés Almansa, Mauricio Delbracio, Pablo Musé, and Pauline Tan. Solving inverse problems by joint posterior maximization with a vae prior. *ArXiv*, abs/1911.06379, 2019.

David S. Greenberg, Marcel Nonnenmacher, and Jakob H. Macke. Automatic posterior transformation for likelihood-free inference. *CoRR*, abs/1905.07488, 2019. URL http://arxiv.org/abs/1905.07488.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25): 723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free mcmc with amortized approximate ratio estimators, 2019.

Maxwell Hong and Ying Cheng. Robust maximum marginal likelihood (rmml) estimation for item response theory models. *Behavior Research Methods*, 51, 10 2018. doi: 10.3758/s13428-018-1150-4.

Ranganath Krishnan, Mahesh Subedar, and Omesh Tickoo. MOPED: efficient priors for scalable variational inference in bayesian deep neural networks. *CoRR*, abs/1906.05323, 2019. URL http://arxiv.org/abs/1906.05323.

Jakob Kruse, Lynton Ardizzone, Carsten Rother, and Ullrich Köthe. Benchmarking invertible architectures on inverse problems. 2019.

Andrew J. Larkoski, Simone Marzani, Gregory Soyez, and Jesse Thaler. Soft drop. *Journal of High Energy Physics*, 2014(5), May 2014a. ISSN 1029-8479. doi: 10.1007/jhep05(2014)146. URL http://dx.doi.org/10.1007/JHEP05(2014)146.

Andrew J. Larkoski, Jesse Thaler, and Wouter J. Waalewijn. Gaining (mutual) information about quark/gluon discrimination. *Journal of High Energy Physics*, 2014(11), Nov 2014b. ISSN 1029-8479. doi: 10.1007/jhep11(2014)129. URL http://dx.doi.org/10.1007/JHEP11(2014)129.

Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *MATHEMATICAL PROGRAMMING*, 45:503–528, 1989.

Gilles Louppe. Likelihood-free inference. 1st Terascale School of Machine Learning, DESY, Hamburg, Germany, Oct 2018.

Gilles Louppe and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators. *ArXiv*, abs/1707.07113, 2019.

Jan-Matthis Lueckmann, Pedro J. Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H. Macke. Flexible statistical inference for mechanistic models of neural dynamics. In *NIPS*, 2017.

Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H. Macke. Likelihood-free inference with emulator networks. In *AABI*, 2018.

Simone Marzani, Lais Schunk, and Gregory Soyez. A study of jet mass distributions with grooming. *Journal of High Energy Physics*, 2017:1–38, 2017.

Simone Marzani, Gregory Soyez, and Michael Spannowsky. Looking inside jets. *Lecture Notes in Physics*, 2019. ISSN 1616-6361. doi: 10.1007/978-3-030-15709-8. URL http://dx.doi.org/10.1007/978-3-030-15709-8.

Thomas Mc Cauley. Display of a event observed in the CMS detector in which a Higgs boson decays to bottom quarks. CMS Collection., Oct 2018. URL https://cds.cern.ch/record/2642472.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *ArXiv*, abs/1906.10652, 2019.

Radford Neal and Geoffrey Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 89, 11 2000. doi: 10.1007/978-94-011-5014-9_12.

Radford M. Neal. Slice sampling. *Ann. Statist.*, 31(3):705–767, 06 2003. doi: 10.1214/aos/1056562461. URL https://doi.org/10.1214/aos/1056562461.

George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with bayesian conditional density estimation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1028–1036. Curran Associates, Inc., 2016.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation, 2017.

George Papamakarios, David C. Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *AISTATS*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Patrick Putzky and Max Welling. Recurrent inference machines for solving inverse problems. *CoRR*, abs/1706.04008, 2017. URL http://arxiv.org/abs/1706.04008.

Françoise Claude Foy Renaud Foy. Introduction to image reconstruction and inverse problems. In *Optics in Astrophysics*, 2002.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *ArXiv*, abs/1505.05770, 2015.

Herbert Robbins. An empirical bayes approach to statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 157–163, Berkeley, Calif., 1956. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200501653.

Donald B. Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *Ann. Statist.*, 12(4):1151–1172, 12 1984. doi: 10.1214/aos/1176346785. URL https://doi.org/10.1214/aos/1176346785.

Viraj Shah and Chinmay Hegde. Solving linear inverse problems using gan priors: An algorithm with provable guarantees. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4609–4613, 2018.

S. N. Shirobokov, Vladislav Belavin, Michael Kagan, Andrey Ustyuzhanin, and Atilim Gunecs Baydin. Differentiating the black-box: Optimization with local generative surrogates. *ArXiv*, abs/2002.04632, 2020.

Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. Pythia 6.4 physics and manual. *Journal of High Energy Physics*, 2006(05):026–026, May 2006. ISSN 1029-8479. doi: 10.1088/1126-6708/2006/05/026. URL http://dx.doi.org/10.1088/1126-6708/2006/05/026.

Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands. An introduction to pythia 8.2. *Computer Physics Communications*, 191:159 – 177, 2015. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2015.01.024. URL http://www.sciencedirect.com/science/article/pii/S0010465515000442.

Jesse Thaler and Ken Van Tilburg. Maximizing boosted top identification by minimizing n-subjettiness. *Journal of High Energy Physics*, 2012(2), Feb 2012. ISSN 1029-8479. doi: 10.1007/jhep02(2012)093. URL http://dx.doi.org/10.1007/JHEP02(2012)093.

Dustin Tran, Rajesh Ranganath, and David M. Blei. Deep and hierarchical implicit models. *ArXiv*, abs/1702.08896, 2017a.

Dustin Tran, Rajesh Ranganath, and David M. Blei. Hierarchical implicit models and likelihood-free variational inference, 2017b.

Karen Ullrich, Rianne van den Berg, Marcus A. Brubaker, David J. Fleet, and Max Welling. Differentiable probabilistic models of scientific imaging with the fourier slice theorem. *CoRR*, abs/1906.07582, 2019. URL http://arxiv.org/abs/1906.07582.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. *CoRR*, abs/1711.10925, 2017. URL http://arxiv.org/abs/1711.10925.

Don van Ravenzwaaij, Peter Cassey, and Scott Brown. A simple introduction to markov chain monte–carlo sampling. *Psychonomic Bulletin Review*, 25, 03 2016. doi: 10.3758/s13423-016-1015-8.

Edgar Walker, Fabian Sinz, Erick Cobos, Taliah Muhammad, Emmanouil Froudarakis, Paul Fahey, Alexander Ecker, Jacob Reimer, Xaq Pitkow, and Andreas Tolias. Inception loops discover what excites neurons most using deep predictive models. *Nature Neuroscience*, 22:1–6, 12 2019. doi: 10.1038/s41593-019-0517-x.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. In *BNAIC/BENELEARN*, 2019.

Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E. Turner, José Miguel Hernández-Lobato, and Alexander L. Gaunt. Fixing variational bayes: Deterministic variational inference for bayesian neural networks. *CoRR*, abs/1810.03958, 2018. URL http://arxiv.org/abs/1810.03958.