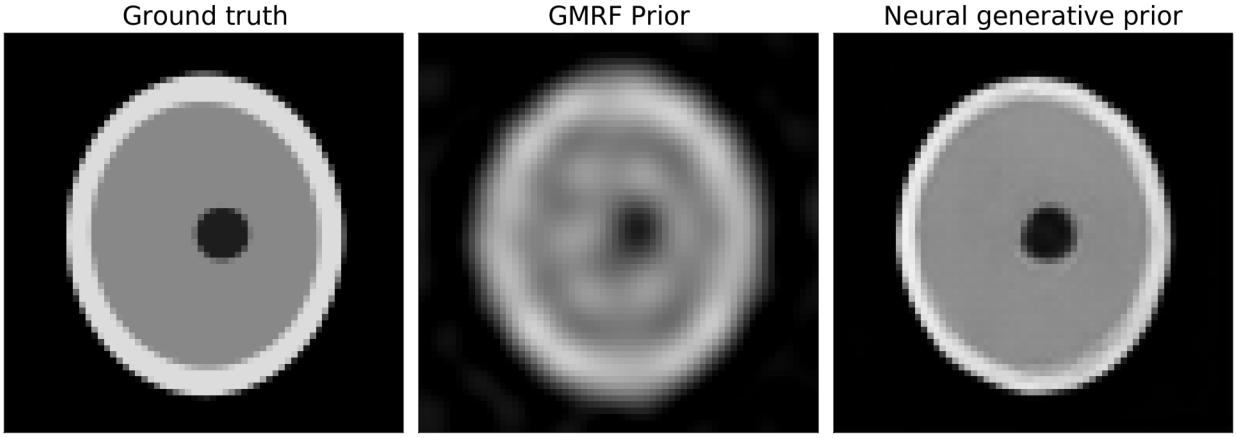


Bayesian Inverse Problems with Neural Generative Priors

Master's thesis in Engineering Mathematics and Computational Science

Vincent Molin

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Bayesian Inverse Problems with Neural Generative Priors

Vincent Molin

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2023

Bayesian Inverse Problems with Neural Generative Priors
Vincent Molin

Supervisors: Axel Ringh and Moritz Schauer, Department of Mathematical Sciences
Examiner: Professor Larisa Beilina, Department of Mathematical Sciences

Master's Thesis 2023
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Comparison of reconstruction using a Gaussian Markov random field prior (middle) and using a neural generative prior (right). The ground truth is shown to the left. The forward operator of the corresponding inverse problem observes the 10×10 low frequency coefficients in both directions of the two dimensional discrete Fourier transform.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Bayesian Inverse Problems with Neural Generative Priors
VINCENT MOLIN
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Inverse problems are ubiquitous in medical imaging and their solutions are essential for clinical decision making. To allow for quantification of uncertainty a Bayesian approach can be taken, but selecting an informed prior is highly non-trivial. We present a data-driven framework where a generative neural network in the form of a Wasserstein Generative Adversarial Network (WGAN) learns a realistic prior, decoupled from the inverse problem in consideration. The method is tested on a severely undersampled two dimensional MRI analogue. We use two different Markov chain Monte Carlo algorithms for approximating the resulting posterior expectations, one based on piecewise deterministic Markov processes, as well as the preconditioned Crank-Nicolson algorithm. The posterior mean and standard deviation is computed for the reconstructions. Our experiments demonstrate that this approach outperforms classical imaging priors both in the quality of reconstructions and also yielding a realistic posterior, allowing for sharp reconstructions with uncertainty using only a sparse observation.

Keywords: Inverse problems, Markov chain Monte Carlo, generative modelling, piecewise deterministic Markov processes

Acknowledgements

First and foremost I would like to express my deep gratitude to Axel and Moritz for their trust, support, guidance, and patience, during this project as well as during other academic pursuits.

Leading up to and during this project I have also tested the patience of others, and I would like to say thank you to the following people:

To my brother Douglas, for your friendship,

To Richard, Carl, and the Council, for keeping me sane,

To Ove, for your tireless efforts,

To Jacob, for your tireless ears,

To Jason, for piercing the void,

To Martin, Linda, Viggo and Elisabeth,

and to the rest of my friends. I do not know where I would be without your support, and I would very much prefer not to find out.

Vincent Molin, Gothenburg, June 2022

Contents

1

Introduction

Before anything else, it is a good idea to clarify what a *problem* is in this context, since there are two prominent uses of the word. Even though essentially similar, a problem hereafter refers to a question that one is interested in answering. One such question is: say that we have observed some effects, i.e., something has happened, what are the underlying properties? This sleuthy question is a so called *inverse problem*.

Inverse problems are a large class of common problems. In a sense, they are so ubiquitous that almost anything we humans do involves solving one. All of our sensory systems collects second hand information from the world and attempts to grasp the underlying object. For example, when light beams that bounced of a dandelion hits a seeing persons retina, it triggers a series of electrochemical impulses propagating through the optic nerve, that in the end leads to the *perception* of a dandelion. From the specific patterns of light waves, an idea of the shape, colour, position and size of an object in the physical world has been recreated.

Continuing with the flower example, the first time one sees a daffodil it would probably take a bit more visual inspection to fully understand the shape of this specific weed. If it is the first flower ever to cross ones path there is even more to unpack. But, crucially, if one is familiar with various plants, ferns and crops, it is fairly easy to understand the daffodil. In that case, it is not impossible that our brain is already equipped with a way to imagine different flowers and it could then only be a matter of tweaking this imagination to fit the new species.

In other words, an argument can be made that our perception of the world relies on *generative models*. For instance, we are able to dream up flowers we have never seen which at least supports the existence of such models. Seemingly, these create an extremely efficient inverse problem solver that allows us to quickly and accurately perceive our environment from limited measurements.

1.1 Inverse problems in technology

Inverse problems naturally arises in a wide range of scientific and engineering areas. A general inverse problem can be formulated as: given a set of observed effects, find

the factors that caused them. We refer to the other direction as the corresponding forward problem. Some important examples of inverse problems are different types of medical imaging problems, such as Magnetic Resonance Imaging (MRI), Computed Tomography (CT), and Positron Emission Tomography (PET).

Recently, purely data–driven methods have shown to be both promising and popular for solving inverse problems. In *Deep Bayesian Inversion* (Adler and Öktem 2018) the authors develop a new method for solving inverse problems by training a conditional generative network that can sample from the posterior distribution conditioned on the observations. This method shows good results, however it requires paired training data consisting of observed data and obtained reconstructions. In many cases the raw observations are however discarded after a reconstruction has been computed.

In this thesis we present a methodology that alleviates the requirement of paired training data by instead only relying on a dataset of reconstructions. With these, we create a generative model. This generative model then allows us to efficiently filter the space of possible solutions for an inverse problem. The hope is that this reduces the amount of data needed in order to obtain good-quality reconstructions. In CT, that would mean using lower dose for similar quality in reconstruction, and in MRI it would mean shorter scanning times for similar quality reconstructions.

1.2 Outline of the thesis

Inverse problems serve as the setting for the work presented in this thesis. However, perhaps somewhat counter–intuitively, the presented method is mostly applicable to inverse problems that are in some sense already well understood. We are concerned with a class of problems where the following assumptions hold:

1. the forward problem is well understood,
2. there is a reasonably accurate model for the observation noise, and,
3. given enough resources, high quality reconstructions can be made.

Many important problems satisfy these conditions. Still, the resources required may be prohibitive in practice. We seek to exploit both the fact that the problem is well understood and that it has been solved many times to create a method that is able to efficiently create a rich posterior, fusing data– and knowledge–driven approaches.

A survey of the building blocks required for the method we are proposing is presented in the first chapters of the thesis. We then lay out the procedure. With the help of a surrogate problem, a caricature of magnetic resonance imaging, we demonstrate that the method works, and also compare it to existing methods. In the closing chapter we discuss what could be pursued going forward and questions that we have encountered during this project.

2

Inverse problems

An inverse problem is the task of recovering an object x, an element of the *reconstruction space* $\mathcal{X}$, given some observed y in the *data space* $\mathcal{Y}$. We refer to the other direction, determining the observation given an object x as the *forward problem*.

In particular, we will consider linear inverse problems related to imaging. An illustrative example of an inverse problem in imaging is *deblurring*, where the corresponding forward problem is a convolution with a blurring kernel.

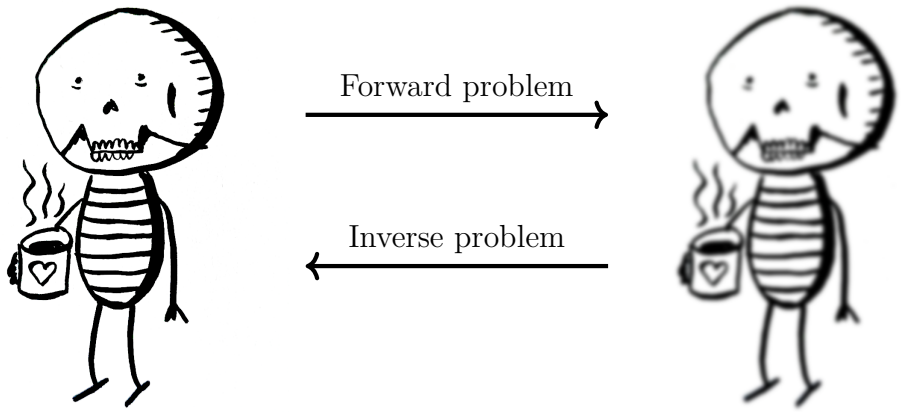


Figure 2.1: Illustration of a deblurring inverse problem. Here, the forward operator is a convolution operator, and the image on the right is obtained by applying this convolution operator to the image on the left.

The blurring operation attenuates high frequency features in the image, which can make the task of reconstructing a sharp image in a stable way non-trivial.

2.1 Linear inverse problems

Assume now that the solution of the forward problem induces a linear operator $\mathscr{A} : \mathcal{X} \to \mathcal{Y}$. We refer to the operator $\mathscr{A}$ as the *forward operator* and it models the observed effects caused by x under the absence of noise. Additionally, the measurement y is also assumed to be disturbed by random noise h, typically modelled by some known distribution.

The mathematical model of the problems we are considering is of the following form.

Problem 2.1.1 (Inverse problem)**.** Given some measurement $y \in \mathcal{Y}$, modelled by

$$y = \mathscr{A}x + h,$$

recover as much information about $x \in \mathcal{X}$ as possible.

The inverse problem is normally harder to solve the the corresponding forward problem. In particular, this is the case when the inverse problem is *ill-posed*.

Definition 2.1.2 (Ill-posed problem, Mueller and Siltanen 2012, Chapter 1)**.** A problem is ill-posed in the sense of Hadamard if at least one of the following conditions do *not* hold:

1. *Existence.* For all $y \in \mathcal{Y}$, there is at least one solution to the problem.
2. *Uniqueness.* For all $y \in \mathcal{Y}$, there is at most one solution to the problem.
3. *Stability.* The solution depends continuously on the data.

If all three conditions are met, the problem is instead called *well-posed*. If the first and the second condition both are met, the forward operator $\mathscr{A}$ is bijective and therefore has a well-defined inverse $\mathscr{A}^{-1}$. Condition 3 then states that $\mathscr{A}^{-1}$ needs to be continuous, i.e., a small change in y should lead to a small change in x.

Example 2.1.3. Let $\mathcal{X} = \mathbb{R}^2, \mathcal{Y} = \mathbb{R}$, and $\mathscr{A}((x_1, x_2)) = x_1 + x_2$. The forward operator $\mathscr{A}$ is linear, for instance, it has the following matrix representation

$$\mathscr{A}(x) = Ax = \begin{bmatrix} 1 & 1 \end{bmatrix} x.$$

For any $y \in \mathcal{Y}$ there exists at least one solution since the column space of A is the entire real line. However, this solution is not unique since the null space of A has dimension 1. Condition 2 is not satisfied and the corresponding inverse problem is therefore ill-posed.

2.2 Regularisation

As seen, ill-posed problems have mathematical properties which make their solutions behave badly, if a solution exists at all. To address these issues, additional structure is typically added to the problem in a process called *regularisation*. There are a wide range of possible techniques and strategies developed over a long period of time for regularising problems, but the core idea is to guide the solution by incorporating prior knowledge or experience about the problem.

Example 2.2.1. Consider the problem of fitting a quadratic curve $y = \beta_2 x^2 + \beta_1 x + \beta_0$ to a set of observed points $(x_i, y_i)_{i=1}^N$. This can be seen as a linear inverse problem, where

$$\beta = (\beta_0, \beta_1, \beta_2) \in \mathcal{X} = \mathbb{R}^3,$$

and

$$y = (y_1, ..., y_N) \in \mathcal{Y} = \mathbb{R}^N.$$

The forward operator $\mathscr{A} : \mathbb{R}^3 \to \mathbb{R}^N$ is the linear transformation given by the matrix vector product

$$\beta \mapsto \begin{bmatrix} 1 & x_1 & x_1^2 \\ & \vdots & \\ 1 & x_N & x_N^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = X\beta.$$

If $N > 3$ and all points $(x_i, y_i)_{i=1}^N$ do not lie exactly on a quadratic curve, $\mathscr{A}\beta = y$ has no solution. The problem can be remedied by assuming that the observations y_i *actually* are related to x_i by $y_i = \beta_2 x_i^2 + \beta_1 x_i + \beta_0 + \varepsilon_i$ where the ε_i are independent draws from a normal distribution $\mathcal{N}(0, \sigma^2)$. The parameters β can now for example be estimated by the maximum likelihood method, which coincides with least squares regression.

Here, the regression problem was regularised by, first, choosing a model for the formation of the observational noise, and then electing to reconstruct the parameters by selecting a suitable criterion to optimise.

2.3 Statistical inversion

In this work, we will take a probabilistic perspective on inverse problems. To this end, we again consider the data formation process

$$y = \mathscr{A}x + h.$$

Here, since h is the outcome of some random variable H, the data y is also the outcome of some $\mathcal{Y}$-valued random variable. Denote this random variable by Y. From a Bayesian point of view, x is also random and the pair (x, y) should be seen as the outcome of the random element (X, Y). Let $P_{X,Y}$ be the joint distribution of this random element over $X \times Y$ and let P_H denote the distribution of the noise H.

To ease the presentation we now restrict ourselves to the case where $\mathcal{X}$ and $\mathcal{Y}$ are subsets of $\mathbb{R}^n$ or $\mathbb{C}^m$. Assuming that the distribution of the additive noise P_H has a probability density π_H, we find that there also exists a conditional density of Y given $x \in \mathcal{X}$, namely

$$\pi(y \mid x) = \pi_H(y - \mathscr{A}x).$$

Previously we added additional information to the problem in order to improve the structure of it. In the probabilistic setting, this regularisation is in essence captured by the definition of a *prior* distribution P_X on $\mathcal{X}$. Assuming that this permits a density π_X we can find the *posterior* density of x given the observation y by applying Bayes' theorem.

Theorem 2.3.1 (Kaipio and Somersalo 2006, Theorem 3.1)**.** *Assume that $X \in \mathbb{R}^n$ has a known prior probability density $\pi_X(x)$ and that the observation y is the outcome of the random variable $Y \in \mathbb{R}^m$ satisfying $\int \pi(y \mid x)\pi_X(x)\mathrm{d}x \neq 0$. Then the posterior density of X given y exists and is given by*

$$\pi(x) = \pi(x \mid y) = \frac{\pi(y \mid x)\pi_X(x)}{\int \pi(y \mid x)\pi_X(x)\mathrm{d}x}.$$

The denominator $\int \pi(y \mid x)\pi_X(x)\mathrm{d}x$ serves as a normalising constant and is called the *model evidence*. The posterior distribution of x is the precise quantification of our knowledge of x. Equipped with the posterior we can answer all manners of statistical questions about x. For instance, we can determine the probability that x has some property P given what we have observed by computing the integral

$$\int_A \pi(x)\mathrm{d}x, \quad A = \{x \in \mathcal{X} \colon P(x)\}. \tag{2.3.1}$$

This requires computing two integrals, and both these can be intractable in practice. In Chapter 4 we will see how this issue can be circumvented by generating samples from the posterior, with methods that only require knowing the posterior up to a multiplicative constant not depending on x. We write this as

$$\pi(x) \propto_x \pi(y \mid x)\pi_X(x).$$

For instance, with a sample $(x_1, ..., x_N)$ from the posterior we can approximate (2.3.1) by

$$\int_A \pi(x)\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} P(x_i).$$

In a Bayesian setting, one way to reformulate Problem 2.1.1 more precisely is as follows.

Problem 2.3.2 (Bayesian inverse problem)**.** Given some measurement $y \in \mathcal{Y}$, modelled by

$$y = \mathscr{A}x + h,$$

recover the posterior density $\pi(x \mid y)$.

Example 2.3.3. In Example 2.2.1 we chose to reconstruct the parameters β using the maximum likelihood method. If we instead assign a prior density to the scale σ of the observation error h, and to the coefficients β_i, we can find a posterior distribution of the parameters. The following computations summarises the steps in O'Hagan 1994, Chapter 9. To find a closed form for the posterior in terms of well-known distributions, it can be shown that choosing the following form of the prior

$$\pi(\beta, \sigma^2) = \pi(\beta \mid \sigma^2)\pi(\sigma^2),$$

where

$$\sigma^2 \sim \text{Inverse-Gamma}(a_0, b_0),$$

and

$$\beta \mid \sigma^2 \sim \text{Normal}(\mu_0, \sigma^2 \Lambda_0^{-1}).$$

yields the following expression for the posterior

$$\begin{aligned}\pi(\beta, \sigma^2 \mid y) \propto (\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2}(y - X\beta)^T(y - X\beta)\right) \\ \cdot (\sigma^2)^{-3/2} \exp\left(-\frac{1}{2\sigma^2}(\beta - \mu_0)^T \Lambda_0 (\beta - \mu_0)\right) (\sigma^2)^{-(a_0+1)} \exp\left(-\frac{b_0}{\sigma^2}\right).\end{aligned}$$

Further, this can be factorised as $\pi(\beta, \sigma^2 \mid y) \propto \pi(\beta \mid \sigma^2, y)\pi(\sigma^2 \mid y)$, where

$$\begin{aligned}\beta \mid \sigma^2, y &\sim \text{Normal}(\mu_n, \sigma^2 \Lambda_0^{-1}), \\ \Lambda_n &= X^T X + \Lambda_0, \\ \mu_n &= (\Lambda_n)^{-1}(X^T X \hat{\beta} + \Lambda_0 \mu_0),\end{aligned}$$

and

$$\begin{aligned}\sigma^2 \mid y &\sim \text{Inverse-Gamma}(a_n, b_n), \\ a_n &= a_0 + \frac{n}{2}, \\ v_n &= b_0 + \frac{1}{2}(y^T y + \mu_0^T \Lambda_0 \mu_0 - \mu_n^T \Lambda_n \mu_n).\end{aligned}$$

In the previous example, the prior densities were chosen so that the posterior had the same functional form. This is a so-called conjugate prior.

In practice there are however two major hurdles that one needs to overcome in order to make the Bayesian approach viable. Firstly, writing down a prior distribution, that actually contains most of our knowledge of the problem, by hand might be impossible. For example, this is the case in almost all inverse problems in imaging, and it is therefore common to resort to using structural priors on the reconstruction space.

Secondly, while Theorem 2.3.1 guarantees the existence of a posterior distribution, it might still be numerically intractable to compute the integrals one is interested in. When faced with complicated posteriors it is commonplace to perform Monte Carlo inference by drawing samples from the posterior.

In the following two chapters the tools needed to tackle these two problems will be presented. Specifically, Chapter 3 tackles the problem of difficult priors by introducing generative models and Chapter 4 presents a modern Markov chain Monte Carlo framework for handling the computationally complicated posteriors that arise.

3

Generative neural networks

The aim of this chapter is to present a class of modern generative models based on neural networks, and some familiarity with the latter is assumed. For a more thorough and formal introduction to neural networks, the reader is referred to the textbook by Goodfellow, Bengio, and Courville 2016[1], or to the myriad of package documentations, blog posts, tutorials and videos available everywhere.

The goal of generative modelling is to generate new data from a distribution of interest. Typically, one starts from a dataset $\{x_i\} \subset \mathcal{X}$, viewing it as a sample from some underlying distribution of interest. As to not bury the lead, the distribution we are interested is the prior on $\mathcal{X}$, and we denote this by P_X. A generative model should then allow one to draw new samples from P_X.

There are a few major groups of generative neural networks, and while their training, capabilities and properties vary, on a high level most work in a similar way. More specifically, after training, the generative network is a function that maps samples from a very simple distribution, i.e., a multivariate standard normal, to samples that should resemble the true distribution. In this work, we will only consider a class of models known as *Generative Adversarial Networks* (GANs). But first, a short primer on neural networks is in place.

3.1 Neural networks

The current widespread use of neural networks is in large part due to the flexibility that this class of models permits. In theory, a neural network can approximate any function up to arbitrary precision, see for instance Section 6.4.1 in Goodfellow, Bengio, and Courville 2016.

We consider a dense feed forward neural network f_θ, consisting of n layers and parametrised by the weights $\theta \in \mathbb{R}^N$. The network f_θ is a function that maps an input vector $z \in \mathbb{R}^d$ to some $f_\theta(z) \in D \subseteq \mathbb{R}^o$.

Let $L^i_{\theta_i}$ be the ith layer of this network. Then f_θ is the composition

$$f_\theta = L^n_{\theta_n} \circ ... \circ L^1_{\theta_1}.$$

[1] Freely accessible at `https://www.deeplearningbook.org`.

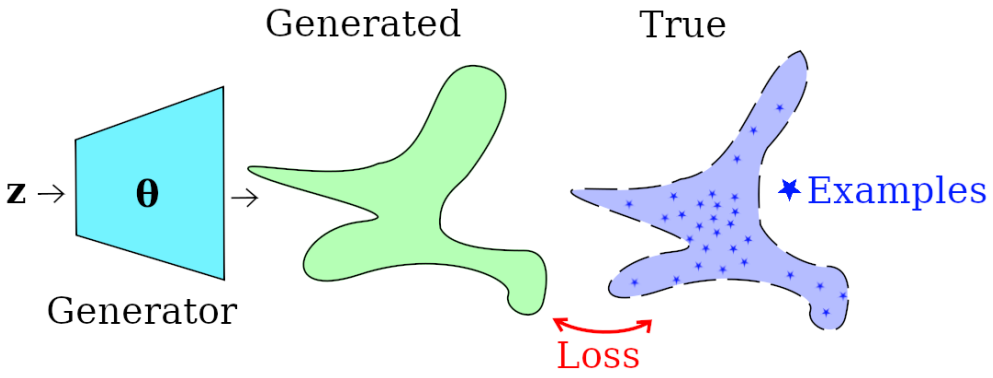


Figure 3.1: Illustration of a generative neural network. Here, z is some simple latent distribution that gets mapped through the potentially complicated generator network g_θ, which is a function parametrised by the vector θ. With the help of examples from the true distribution one is interested in approximating, a loss for the generated distribution can be computed and θ is tuned by the minimisation of this loss.

Further, each layer of this network can be written as

$$L^i_{\theta_i}(x) = \sigma.(W_i x + b_i),$$

where $W_i x$ is a matrix vector product, b_i is the additive bias, and $\sigma.$ is the *activation function* of the layer, which typically is applied element–wise. If the activation functions are linear the network collapses to an affine function $Wx + b$, so typically all intermediate activations are non-linear.

3.1.1 Smooth neural networks

While still in their infancy, it was common to use saturating activation functions such as the *Sigmoid*, $x \mapsto (1 + e^{-x})^{-1}$, or tanh in neural networks. These where the natural choice due to their similarity with the observed activation patterns of real, biological neurons. These functions do however have numerical properties that can cause difficulties during training. A less biologically inspired approach has been observed to perform better, and for quite some time a common choice of non-linearities in the intermediate layers of the network has been the rectified linear unit (ReLU), given by $x \mapsto \max(0, x)$. However, this activation function does not have a continuous derivative at $x = 0$.

In our setting, we will later see that it is important that the gradient of networks we are interested in is well-behaved. Specifically, we would like to have a gradient ∇f that is Lipschitz continuous, and, preferably, we would like this Lipschitz constant to be somewhat small.

Definition 3.1.1. The Sigmoid-weighted Linear Unit (SiL), presented in Elfwing, Uchibe, and Doya 2018, is an activation function given by

$$x \mapsto x \cdot \text{Sigmoid}(x) = x \cdot \frac{1}{1 + e^{-x}}.$$

This activation function coincides with the *Swish* activation from Ramachandran, Zoph, and Le 2017,

$$x \mapsto x \cdot \frac{1}{1 + e^{-\beta x}},$$

with the trainable parameter in the latter fixed to $\beta = 1$.

In practice, it is common to refer to both $x \cdot \text{Sigmoid}(x)$ and $x \cdot \text{Sigmoid}(\beta x)$ as Swish, and going forward Swish will refer to the former.

Remark 3.1.2. Both Sigmoid and Swish are smooth activation functions. This means that any finite network with only these two activation functions is smooth, and therefore at least has a locally Lipschitz gradient.

3.1.2 Convolutional neural networks

The fully connected networks presented so far have been around for a long time, heralding back to the computational model of a neuron suggested in Mcculloch and Pitts 1943, the perceptron (Rosenblatt 1957), and the advent of computers in general. Faith in artificial neural networks has fluctuated since their introduction, enjoying years of unbridled enthusiasm as well as more pessimistic periods, typically nicknamed AI winters.

While there is no clear starting point for the latest wave of optimism, the early 2010s saw the rise of deep learning with convolutional neural networks starting to outpace other methods in computer vision. Convolutional architectures, inspired by the organisation of neurons in the visual cortices of mammals, had already been around for around 20 years but this is where the first implementations running on graphical processing units (GPUs) saw the light of day.

Compared to the fully connected architectures, convolutional architectures can be seen as imposing a lot of additional structure and sparseness on the weight matrices of the networks. Specifically, this is due to the fact that many of the layers represent discrete convolutions with kernels with relatively small support. This has lead to great performance on various tasks involving image data and convolutional architectures are today ubiquitous when dealing with images in machine learning.

3.1.3 Training of neural networks

Approximate optimisation of some scalar loss function with respect to the weights of a neural network is called training the network. Neural networks are typically trained with stochastic gradient descent, and the gradients are computed using some form of reverse mode automatic differentiation, where back-propagation is a classic special case (Section 6.5, Goodfellow, Bengio, and Courville 2016).

To describe training in a bit more detail, we consider a supervised learning task, consisting of a model f_θ, a dataset $D = \{(x_i, y_i)\}_{i=1}^N$ of label and observation example tuples and some scalar loss $l(y, \widetilde{y})$ quantifying the severity of the prediction error.

One is interested in minimising the expected prediction error

$$\mathscr{L}(\theta) := \mathbb{E}_{(x,y)\sim P_{X\times Y}}[l(y, f_\theta(x))],$$

where $P_{X\times Y}$ is the underlying joint distribution of labels and observations. In practice, $P_{X\times Y}$ is typically unknown. However, one normally has access to a finite sample of examples and the expectation is replaced with its empirical counterpart

$$\mathscr{L}_{emp}(\theta) := \frac{1}{N}\sum_{i=1}^{N} l(y_i, f_\theta(x_i)).$$

When N is large it can be computationally infeasible to take gradients of the losses of every training example at each step. Instead, a finite subsample is considered at each step, and in place of the gradient of $\mathscr{L}_{emp}$ the random estimate resulting from this subsample is used instead. Performing gradient descent using these estimates is known as *stochastic gradient descent*, and it forms the basis for training most neural networks.

3.2 Generative Adversarial Networks

We now turn our attention to a scheme for training generative neural networks, famously invented one late night at a bar by Goodfellow et al. 2014. The idea is to train a generative neural network to model some complicated distribution by having it compete with a second critiquing network. This second network, called the *discriminator* or the *critic* f_ϕ is trained to discern between real samples from P_X and fake samples created by the *generator* g_θ, which in turn is optimised to fool the discriminator.

More precisely, the generator is a function from some low-dimensional latent space $\mathcal{Z}$ with a simple and predetermined noise distribution P_Z, typically a multivariate standard normal distribution. In its most general form, the competition between the discriminator and the generator is formulated as the minimax problem

$$\theta^* = \operatorname*{argmin}_\theta \max_\phi \; \mathbb{E}_{x\sim P_X, z\sim P_Z} V[f_\phi, g_\theta](x, z), \tag{3.2.1}$$

where $V[f_\phi, g_\theta]$ is some scalar function. This minimax problem is referred to as *adversarial training*. In the original article by Goodfellow et al. the value function

$$V[f_\phi, g_\theta](x, z) = \log f_\phi(x) + \log(1 - f_\phi(g_\theta(z))) \tag{3.2.2}$$

was used.

3.2.1 Theoretical motivation of GANs

The following section serves as a theoretical motivation as to why training GANs could be a good idea. However, note that the assumptions made do not hold in practice most of the time. Nevertheless, training GANs have become a powerful

method to obtain generative neural networks. The presentation closely follows the one of Goodfellow et al. 2014.[2]

First let us convince ourselves that the generators transformation of the latent noise actually induces a probability measure on the image space. This is typically expressed in a language requiring a few measure-theoretic technicalities. If the reader is unfamiliar, the gist is that it is pretty clear that one gets a probability distribution on the image space but this might not permit a probability density as one is used to.

Let $\mathscr{M}_+(\mathcal{X})$ denote the space of non-negative measures on $\mathcal{X}$, equipped with the usual Borel σ-algebra Σ, and let $\mathscr{M}_+^1(\mathcal{X})$ be the subset of these with total measure 1, i.e., the probability measures on $\mathcal{X}$.

Definition 3.2.1. Let $(\mathcal{X}, \Sigma)$ and $(\mathcal{Y}, T)$ be two measurable spaces. A function $f\colon \mathcal{X} \to \mathcal{Y}$ is said to be measurable if the pre-image of every E in T lies in Σ, that is, f is measurable if

$$\forall E \in T\colon f^{-1}(E) := \{x \in \mathcal{X} : f(x) \in E\} \in \Sigma.$$

This condition is easily satisfied and we can safely assume that the generator $g_\theta\colon \mathcal{Z} \to \mathcal{X}$ is a measurable function with respect to the measurable spaces $\mathcal{Z}$ and $\mathcal{X}$. Now, given a probability measure on the latent space $\mathcal{Z}$ and some measurable function, we can push this measure to $\mathcal{X}$ in the following way.

Definition 3.2.2. Let $f\colon \mathcal{X} \to \mathcal{Y}$ be measurable, and let μ be a measure on $\mathcal{X}$. The *push-forward* of μ under f, denoted $f_*(\mu)$, is a measure on $\mathcal{Y}$ given by

$$f_*(\mu)(B) = \mu(f^{-1}(B)).$$

Now, we can be sure that it makes sense to talk about a learned distribution Q_θ on the image space, by letting Q_θ be the push-forward of the latent noise distribution P_Z under the generator g_θ, i.e.,

$$Q_\theta := (g_\theta)_*(P_Z).$$

We set out to show that solving (3.2.1), with V selected as (3.2.2), minimises a distance between the learned distribution Q_θ and the data distribution P_X.

Definition 3.2.3. Let μ and ν be two measures defined on the same measurable space $(\mathcal{X}, \Sigma)$. Then μ is said to be *absolutely continuous* with respect to ν if $\mu(A) = 0$ for all $A \in \Sigma$ such that $\nu(A) = 0$.

Absolute continuity permits the existence of *densities* by the Radon-Nikodym Theorem (see Billingsley 1995, Section 32). We state this result informally. If μ is

[2]Colin Raffel has written a nice exposition on the topic, available at `https://colinraffel.com/blog/gans-and-divergence-minimization.html`.

absolutely continuous with respect to ν, there exists a measurable function p such that

$$\int_A \mathrm{d}\nu = \int_A p \,\mathrm{d}\mu,$$

for all measurable sets A.

Definition 3.2.4. Let $P, Q \in \mathscr{M}_+^1(\mathcal{X})$ be two probability measures, where P is absolutely continuous with respect to Q. Let μ be a measure such that both P and Q are absolutely continuous with respect to μ. The *Kullback-Leibler divergence* D_{KL} from P to Q is given by

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{\mathcal{X}} p \log\left(\frac{p}{q}\right) \mathrm{d}\mu,$$

where p and q are densities of P respectively Q with respect to μ.

Note that the existence of a reference measure μ in the definition above can be assumed without loss of generality, for instance, the choice of $\mu = \frac{1}{2}(P+Q)$ satisfies the condition.

While it quantifies distance between measures in some sense, the Kullback-Leibler divergence does not satisfy the axioms of a metric. For example, it is not symmetric in the arguments. For two probability measures P, Q it does however hold that it is non-negative, and that $D_{\mathrm{KL}}(P \parallel Q) = 0$ implies that $P = Q$ (Bishop 2006, Section 1.6.1). We can easily construct a symmetric divergence in the following way.

Definition 3.2.5. The *Jensen-Shannon divergence* D_{JS} between P and Q is given by

$$D_{\mathrm{JS}}(P, Q) = \frac{1}{2} D_{\mathrm{KL}}(P \parallel \mu) + \frac{1}{2} D_{\mathrm{KL}}(Q \parallel \mu),$$

where

$$\mu = \frac{1}{2}(P + Q).$$

While the Jensen-Shannon divergence is still not a metric (it does not satisfy the triangle inequality), the square root of the divergence is actually a metric (Endres and Schindelin 2003).

Proposition 3.2.6. *Assume that the discriminator $f_\phi : \mathcal{X} \to [0,1]$ can represent any function. The minimax problem*

$$\theta^* = \operatorname*{argmin}_\theta \max_\phi \; \mathbb{E}_{x \sim P_X, z \sim P_Z}[\log f_\phi(x) + \log(1 - f_\phi(g_\theta(z)))]$$

reduces to

$$\theta^* = \operatorname*{argmin}_\theta \; D_{JS}(P_X, Q_\theta).$$

Proof. Let μ be the probability measure on $\mathcal{X}$ given by $\mu = \frac{1}{2}(Q_\theta + P_X)$, and let q_θ and p_X be densities of Q_θ and P_X with regard to μ. For brevity, denote

$$\mathbb{E}\, V[f_\phi, g_\theta] := \mathbb{E}_{x \sim P_X, z \sim P_Z} V[f_\phi, g_\theta](x, z).$$

Then

$$\begin{aligned}\mathbb{E}\, V[f_\phi, g_\theta] &= \mathbb{E}_{x\sim P_X, z\sim P_Z}[\log f_\phi(x) + \log(1 - f_\phi(g_\theta(z))] \\ &= \int_{\mathcal{X}} (p_X \log f_\phi + q_\theta \log(1 - f_\phi))\, \mathrm{d}\mu.\end{aligned}$$

For fixed θ, the integrand is maximised by

$$f_\phi^* = \operatorname*{argmax}_y \; p_X \log y + q_\theta \log(1 - y) = \frac{p_X}{p_X + q_\theta}. \tag{3.2.3}$$

Then, since this choice of f_ϕ^* maximises the expectation, it follows that

$$\begin{aligned}\max_\phi \; \mathbb{E}\, V[f_\phi, g_\theta] &= \int_{\mathcal{X}} p_X \log\left(\frac{p_X}{p_X + q_\theta}\right) \mathrm{d}\mu + \int_{\mathcal{X}} q_\theta \log\left(\frac{q_\theta}{p_X + q_\theta}\right) \mathrm{d}\mu \\ &= \int_{\mathcal{X}} p_X \log\left(\frac{1}{2} \cdot \frac{p_X}{\left(\frac{p_X+q_\theta}{2}\right)}\right) \mathrm{d}\mu + \int_{\mathcal{X}} q_\theta \log\left(\frac{1}{2} \cdot \frac{q_\theta}{\left(\frac{p_X+q_\theta}{2}\right)}\right) \mathrm{d}\mu \\ &= \int_{\mathcal{X}} p_X \log\left(\frac{p_X}{\left(\frac{p_X+q_\theta}{2}\right)}\right) \mathrm{d}\mu + \int_{\mathcal{X}} q_\theta \log\left(\frac{q_\theta}{\left(\frac{p_X+q_\theta}{2}\right)}\right) \mathrm{d}\mu - 2\log 2 \\ &= D_{\mathrm{KL}}\left(P_X \,\middle\|\, \frac{P_X + Q_\theta}{2}\right) + D_{\mathrm{KL}}\left(Q_\theta \,\middle\|\, \frac{P_X + Q_\theta}{2}\right) - 2\log 2 \\ &= 2 \cdot D_{\mathrm{JS}}(P_X, Q_\theta) - 2\log 2,\end{aligned}$$

so

$$\theta^* = \operatorname*{argmin}_\theta \; D_{\mathrm{JS}}(P_X, Q_\theta), \tag{3.2.4}$$

which completes the proof. □

Corollary 3.2.7. *In the ideal setting, the optimal generator models the data distribution exactly, i.e., $P_X = Q_\theta$.*

In practice, the discriminator f_ϕ is a neural network that does not have infinite capacity. In addition it is also optimised with stochastic gradient ascent and there is no guarantee that it converges to the global optimum. This means that it at best represents some approximation of the Jensen-Shannon divergence.

Alongside the difficulties of finding the optimal discriminator one is faced with another dilemma. If the discriminator is near optimal, and the generator is not, one would have

$$f_\phi(x) \approx 1, \quad f_\phi(g_\theta(z)) \approx 0, \quad \text{for } x \sim P_X, z \sim P_Z.$$

Numerically, this would then lead to vanishing gradients for the generator which can cause the optimisation of the generator to grind to halt, hindering convergence. On the other hand, if the discriminator is not able to discern well, the gradients are instead meaningless. For a more detailed discussion of these difficulties, we refer the reader to the article *Towards Principled Methods for Training Generative Adversarial Networks* by Arjovsky and Bottou 2017.

In practice, successful training of GANs requires a delicate balance between the strengths of the generator and the discriminator in the intermediate steps as well as a bit of luck.

3.2.2 Wasserstein GANs

Introduced in Arjovsky, Chintala, and Bottou 2017, *Wasserstein Generative Adversarial Networks* (WGANs) seeks to improve the training of GANs by, loosely speaking, replacing the somewhat problematic Jensen-Shannon divergence with a more well-behaved distance. To possibly little surprise, the new candidate metric is the *Wasserstein-1* metric, also called the *Earth mover* distance.

Definition 3.2.8. The Wasserstein-1 distance between two probability measures $\mu, \nu \in \mathscr{M}_+^1(\mathcal{X})$ is the optimal solution to the linear program

$$\begin{aligned} W_1(\mu,\nu) := \min_{\pi \in \mathscr{M}_+^1(\mathcal{X}\times\mathcal{X})} \quad & \int_{\mathcal{X}\times\mathcal{X}} \|x_0 - x_1\|_2 \pi(\mathrm{d}x_0, \mathrm{d}x_1), \\ \text{subject to} \quad & \int_{\mathcal{X}} \pi(\cdot, \mathrm{d}x_1) = \mu(\mathrm{d}x_0), \\ & \int_{\mathcal{X}} \pi(\mathrm{d}x_0, \cdot) = \nu(\mathrm{d}x_1). \end{aligned}$$

Intuitively, this is the minimal amount of work one would have to carry out to move all probability mass in μ to ν if the cost of moving a mass m from x to y is $m\|x-y\|_2$.

Defining $\mathscr{U}(\mu,\nu)$ to be the set of feasible solutions to the linear program in Definition 3.2.8, that is, the joint probability distributions over $\mathcal{X}\times\mathcal{X}$ with marginals μ and ν, the W_1 distance can be rewritten as

$$W_1(\mu,\nu) = \inf_{\pi\in\mathscr{U}(\mu,\nu)} \mathbb{E}_{(X,Y)\sim\pi}(\|X-Y\|_2).$$

Let $\mathrm{Lip}_k(\mathcal{X})$ denote the set of scalar valued Lipschitz functions $f\colon \mathcal{X}\to\mathbb{R}$ with Lipschitz constant less than k. By the Kantorovich-Rubinstein duality (see Peyré, Cuturi, et al. 2019, page 97-98) it holds that[3]

$$W_1(\mu,\nu) = \sup_{f\in\mathrm{Lip}_1(\mathcal{X})} \mathbb{E}_{x\sim\mu} f(x) - \mathbb{E}_{y\sim\nu} f(y). \tag{3.2.5}$$

To reformulate the training criterion in terms of the Wasserstein distance, first note that we want to select θ^* by

$$\theta^* \in \operatorname*{argmin}_\theta W_1(P_X, Q_\theta),$$

which we can expand as

$$\theta^* \in \operatorname*{argmin}_\theta \sup_{f\in\mathrm{Lip}_1(\mathcal{X})} \mathbb{E}_{x\sim P_X} f(x) - \mathbb{E}_{z\sim P_Z} f(g_\theta(z)).$$

[3]for any μ,ν such that $\int_{\mathcal{X}} \|x_0-x\|_2\mu(\mathrm{d}x) < +\infty$ and $\int_{\mathcal{X}} \|x_0-x\|_2\mu(\mathrm{d}x) < +\infty$, for an arbitrary $x_0\in\mathcal{X}$.

The set of 1-Lipschitz is intractable to optimise over, so we approximate the supremum over all possible 1-Lipschitz functions by replacing it with a neural network f_ϕ. To ensure that the Lipschitz condition still holds additional steps needs to be taken. Considering again the formulation of adversarial training in the beginning of this section, (3.2.1), this is seen to correspond to the value function

$$V[f_\phi, g_\theta](x, z) = f_\phi(x) - f_\phi(g_\theta(z)).$$

In the original article by Arjovsky, Chintala, and Bottou 2017, the authors use weight-clipping to ensure the critic meets the Lipschitz condition. This is done by projecting all network weights to lie within the interval $[-c, c]$ at every iteration. A less crude approach that empirically has been shown to improve training further instead adds the Lipschitz condition as a penalised term to the objective function. The resulting training scheme, introduced in Gulrajani et al. 2017, is known as *WGAN with gradient penalty* (WGAN-GP for short), and the training problem is given by

$$\theta^* \in \operatorname*{argmin}_\theta \max_\phi \; \mathbb{E}_{x\sim P_X} f_\phi(x) - \mathbb{E}_{z\sim P_Z} f_\phi(g_\theta(z)) + \lambda \mathbb{E}_{\hat{x}\sim \mathbb{P}_{\hat{x}}} \|\nabla f_\theta(\hat{x}) - 1\|_2^2. \quad (3.2.6)$$

The distribution $\mathbb{P}_{\hat{x}}$ is defined implicitly, arising by uniformly sampling points on lines between pairs of points (x, y) sampled from (P_X, Q_θ). In Gulrajani et al. 2017 it is shown that the gradient of the optimal f in (3.2.5) has unit norm on these lines, which motivates the two-sided penalty on the gradient norm in (3.2.6) above.

We conclude the chapter by a slight reformulation of Theorem 1 in Arjovsky, Chintala, and Bottou 2017.

Theorem 3.2.9 (Theorem 1, Arjovsky, Chintala, and Bottou 2017)**.** *Let P_X be a fixed distribution over $\mathcal{X}$. Let P_Z be a known distribution (e.g Gaussian) over another space $\mathcal{Z}$. Let $g : \mathcal{Z} \times \mathbb{R}^d \to \mathcal{X}$ be a function, that will be denoted $g_\theta(z)$ with z the first coordinate and θ the second. Let Q_θ denote the push-forward $(g_\theta)_*(P_Z)$. Then,*

1. *If g is continuous in θ, so is $W_1(P_X, Q_\theta)$.*
2. *If g is locally Lipschitz and satisfies a regularity assumption, then $W_1(P_X, Q_\theta)$ is continuous everywhere, and differentiable almost everywhere.*
3. *Statements 1-2 are false for the Jensen-Shannon divergence D_{JS}.*

While we have not directly touched upon the hypothetical low-dimensional support of P_X, this is one major reason that we expect that the low-dimensional latent parametrisation is sufficient. The key takeaway from Theorem 3.2.9 is that, in this case, the Wasserstein distance is differentiable whereas the Jensen-Shannon divergence is not necessarily.

4

Markov chain Monte Carlo and piecewise deterministic Markov processes

In this chapter we will consider two methods for tractable Bayesian inference. These methods rely on generating samples from the *target* distribution one is interested in. Both rely on creating a stochastic process which is both relatively easy to simulate and has the target distribution one is interested in as an invariant distribution. Simulation of stochastic processes become much simpler if it is sufficient to only know the current state of the process, i.e., if they possess the Markov property. As the avid reader might have noticed from the title of this chapter, this is something both methods have in common.

4.1 Making Bayesian inference tractable

Recall the posterior density given by Theorem 2.3.1

$$\pi(x) = \pi(x \mid y) = \frac{\pi(y \mid x)\pi_X(x)}{\int \pi(y \mid x)\pi_X(x)\mathrm{d}x},$$

and denote the corresponding probability measure by Π. Expectations with respect to this measure then reduces to integrals, i.e.,

$$\mathbb{E}_{x\sim\Pi} f(x) = \int_{\mathcal{X}} f(x)\pi(x)\mathrm{d}x, \tag{4.1.1}$$

where f is some measurable function. The evaluation of such expectations is the goal of statistical computing. For instance, being able to compute these would allow us to find the posterior mean and the posterior variance.

In Example 2.3.3 we saw how one could carefully select the functional form of prior densities to guarantee that the posterior is of the same form. This is known as *conjugacy* and allows for closed-form expressions of the posterior. Slight variations of the prior can however easily break the conjugacy, leading to complicated posterior densities not described by the named parametric distributions available. To be able

to handle such more general forms of posterior distributions one needs additional tools.

If we were able to generate a sample $(x_1, ..., x_N)$ from Π, we could approximate the expectation in (4.1.1) with the Monte Carlo approximation

$$\mathbb{E}_{x\sim\Pi} f(x) \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i).$$

Developed during the latter half of the twentieth century and popularised by the growing availability of computational resources near the turn of the millennium, Markov chain Monte Carlo (MCMC) methods allow generating samples from a posterior of interest. These sampling methods have made Bayesian inference practical at a much larger scale. Still, when the dimensionality of the space on which the target distribution is defined increases, standard MCMC algorithms typically become arbitrarily slow. We first turn our attention to one modification of the classical Metropolis-Hastings algorithm (see Bishop 2006, Section 11.2.2).

4.2 The preconditioned Crank-Nicolson algorithm

The preconditioned Crank-Nicolson (pCN) algorithm, named in Cotter et al. 2013 but first invented in 2008, is a Markov chain Monte Carlo method that can sample densities of the form

$$\pi(x) \propto \exp(-\Phi(x))\mu(x),$$

where $\mu(x)$ is a Gaussian prior density[1]. The chain of samples is generated by proposing a new point and either accepting or rejecting it as in the Metropolis-Hastings algorithm. For comparison, let us first consider the standard random walk Metropolis-Hastings algorithm for sampling from Π. Let $\psi(x) = -\log \pi(x)$.

- Select an initial point $x^{(0)}$ and set $k = 0$.
- Repeat N times:
  - Propose $x^o = x^{(k)} + \beta \cdot u$, where $u \sim \mathcal{N}(0, \Sigma)$.
  - Set $x^{(k+1)} = x^o$ with probability $a(x^o, x^{(k)}) = \min\{1, \exp(\psi(x^o) - \psi(x^{(k)})\}$.
  - Set $x^{(k+1)} = x^{(k)}$ otherwise.

Importantly, we have that the proposal $x^o \mid x^{(k)} \sim \mathcal{N}(x^{(k)}, \beta\Sigma)$. For fixed β, this causes the acceptance probability to tend to zero as the dimensionality of x tends to infinity. The key idea in the preconditioned Crank-Nicolson algorithm is to modify the proposal in a way so that the acceptance probability does not approach zero.

The proposal is modified in the following way. An increment u is sampled from the prior μ, and a *Crank-Nicolson step*, with parameter $\rho \in (0, 1)$, is performed to

[1]Note that this is a slight abuse of notation compared to previous chapters.

procure the proposal, i.e.,

$$x^o = \sqrt{1-\rho^2} \cdot x^{(k)} + \rho \cdot u, \quad u \sim \mu.$$

Further, with the acceptance probability

$$a\left(x^o, x^{(k)}\right) = \min\{1, \exp\left(\Phi\left(x^{(k)}\right) - \Phi\left(x^o\right)\right)\},$$

it can be shown that this algorithm has an acceptance probability independent of the dimension of x (Hairer, Stuart, and Vollmer 2014).

Both the random walk Metropolis-Hastings and the preconditioned Crank-Nicolson algorithms define Markov chains with the target distribution as invariant distributions. For a proof of this statement for the former, we refer the reader to Section 11.2.2 in Bishop 2006, and for the latter, see Hairer, Stuart, and Vollmer 2014. Algorithm 1 gives a full description of the implementation of this method.

Algorithm 1 The preconditioned Crank-Nicolson algorithm

Require: $\rho \in (0,1), x_0, N$, a negative log likelihood Φ, a Gaussian prior $\mu = \mathcal{N}(0, \Sigma)$.
  $x \leftarrow x_0$
  $S \leftarrow \{x\}$
  for $1 \dots N-1$ **do**
    $x^o \leftarrow \sqrt{1-\rho^2} \cdot x + \rho \cdot u, \quad u \sim \mu$
    $R \sim \text{Unif}([0,1])$
    if $R < \exp(\Phi(x) - \Phi(x^o))$ **then**
      $x \leftarrow x^o$
    end if
    $S \leftarrow \{x, S\}$
  end for
  return S

4.3 Piecewise deterministic Markov processes

We now turn our attention towards a novel class of MCMC related methods. Compared to traditional MCMC methods based on the Metropolis-Hastings algorithm, *piecewise deterministic Markov process Monte Carlo* (PDMP-MC) are a class of non-reversible and rejection free algorithms for exploring posteriors.

First introduced in Davis 1984, a *piecewise deterministic Markov process* (PDMP) is a continuous time stochastic process with *càdlàg*[2] paths in a locally Euclidean space E. The process evolves deterministically between random jumps that happen at random event times. A PDMP $(Z_t)_{t\geq 0}$ is characterised by

1. an ordinary differential equation with differentiable drift $\xi : E \to E$, that is
$$\frac{\mathrm{d}z_t}{\mathrm{d}t} = \xi(z_t),$$
inducing the deterministic dynamics $\varphi : \mathbb{R}_{\geq 0} \times E \to E$,
2. an event rate $\lambda : E \to \mathbb{R}_{\geq 0}$, with $\lambda(z_t)\varepsilon + o(\varepsilon)$ being the probability of an event in the time interval $[t, t+\varepsilon]$,
3. a Markov transition kernel $\mathcal{Q}(z, \mathrm{d}z')$, where at an event time τ a new state z'_τ is sampled from the probability distribution $\mathcal{Q}(z_\tau, \cdot)$.

The analysis of properties of a given PDMP is eased by the *generator* of the process, of which we give an informal definition.

Definition 4.3.1. The *generator* $\mathfrak{A}$ of a PDMP is a function defined on a well-behaved space of measurable functions, defined by
$$\mathfrak{A}f(z) := \lim_{t\to 0} \frac{\mathbb{E}[f(Z_t) \mid Z_0 = z] - f(z)}{t}.$$

Under suitable conditions (see Davis 1993, Theorem 26.14) the generator is given by
$$\mathfrak{A}f(z) = \langle \xi(z), \nabla f(z)\rangle + \lambda(z) \int (f(z') - f(z))\, \mathcal{Q}(z, \mathrm{d}z'). \tag{4.3.1}$$

In other words, the generator of a PDMP applied to a function, $\mathfrak{A}f$, associates to each point x the infinitesimal expected change if the process would run from x for a small time $\mathrm{d}t$. As such it can be helpful to think of it as some type of derivative. Indeed, for a specific case we can explicitly recover this connection.

Example 4.3.2. Let us consider a simple PDMP where a particle in $\mathbb{R}^n$ travels with some constant velocity, unbothered by random events. To this end, define $E = \mathbb{R}^n \times \mathbb{R}^n$ and write $z = (x, v) \in E$ where $x, v \in \mathbb{R}^n$. We set the drift to

[2] *continue à droite, limite à gauche*, i.e., right-continuous with limits on the left, and French.

$\xi(z) = (v, 0)$, which induces the linear flow $\varphi(t, z) = (x + tv, v)$. With the event rate $\lambda \equiv 0$ we have precisely the prescribed scenario, showing that it actually is a PDMP. Now, let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable measurable function, and let $\widetilde{f}(z) = (f(x), 0)$. Then, by (4.3.1), we have that

$$\mathfrak{A}\widetilde{f}(z) = \left\langle \xi(z), \nabla \widetilde{f}(z) \right\rangle \tag{4.3.2}$$

$$= \langle (v, 0), (\nabla f(x), 0) \rangle \tag{4.3.3}$$

$$= v^T \nabla f(x), \tag{4.3.4}$$

which is the directional derivative of f along v evaluated at x.

4.3.1 Sampling from a target measure

In discrete time MCMC, a Markov chain is constructed in such a way that the stationary distribution of the process is the distribution of interest. Samples are then generated simply by simulating the Markov chain and collecting the visited states. Analogously, we now show how one constructs a PDMP which preserves the target distribution.

Let $\Pi(\mathrm{d}x)$ be a target probability measure on $\mathbb{R}^d$, permitting a density $\pi(x)$ with respect to the Lebesgue measure $\mathrm{d}x$, that we are interested in sampling from. Let

$$\psi(x) := -\log \pi(x)$$

be the negative logarithm of the density π. In what follows we consider PDMPs on state spaces $E = \mathbb{R}^d \times V$, where either $V = \mathbb{R}^d$, or $V = S^{d-1}$, the d-dimensional unit sphere. For a state z in E it is convenient to write $z = (x, v)$ and call $x \in \mathbb{R}^d$ the position and $v \in V$ the velocity.

A necessary condition for the PDMP to have the measure $\mu(\mathrm{d}z)$ as a unique stationary measure is to admit it as a stationary measure at all. Proposition 34.7 in Davis 1993 states that invariance with respect to μ will be satisfied if

$$\int \mu(\mathrm{d}z) \mathfrak{A} f(z) = 0$$

for all functions f in the domain of the generator $\mathfrak{A}$.

We give a definition of the *Bouncy Particle Sampler*, discussed in detail in Bouchard-Côté, Vollmer, and Doucet 2018.

Definition 4.3.3. The bouncy particle sampler is a PDMP typically on the product space $E = \mathbb{R}^d \times \mathbb{R}^d$. The deterministic dynamics are given by

$$\varphi(t, (x, v)) = (x + tv, v),$$

i.e., linear trajectories with drift $\xi((x, v)) = (v, 0)$. The events happen at a rate

$$\lambda(x, v) = \lambda_B(x, v) + \lambda_R$$

where

$$\lambda_B(x, v) = \max(0, \langle v, \nabla\psi(x)\rangle)$$

is the bounce rate and λ_R is the constant refreshment rate. For the bounce events we have

$$\mathcal{Q}_B((x, v), \cdot) = \delta_{(x, B_{\nabla\psi} v)}$$

where

$$B_{\nabla\psi} v = v - 2\frac{\langle v, \nabla\psi(x)\rangle}{\|\nabla\psi(x)\|_2^2}\nabla\psi(x),$$

and for the refreshments we have

$$\mathcal{Q}_R((x, v), (\mathrm{d}x, \mathrm{d}v)) = (\delta_x(\mathrm{d}x), \mathcal{N}(\mathrm{d}v; 0, I_d)),$$

i.e., the speeds are randomised according to a standard normal distribution. We define the full transition kernel $\mathcal{Q}$ implicitly; we can simulate two independent stopping times, one for each kernel with their corresponding rates. If the refreshment process stops first, the new state is simulated from $\mathcal{Q}_R$ and vice versa for $\mathcal{Q}_B$.

Proposition 4.3.4. *The product measure $\mu(\mathrm{d}z) = \Pi(\mathrm{d}x) \otimes \mathcal{N}(\mathrm{d}v; 0, I_d)$ is an invariant measure for the bouncy particle sampler.*

The proof of this statement is left out in favor of a similar result later. For a reference, see Vanetti et al. 2017.

In practice, the refreshment rate λ_R needs to be tuned to achieve good performance. If λ_R is too large, the process evolves more like a random walk, which is only expected to cover a distance $\sqrt{\mathrm{d}t}$, and therefore explores the posterior slowly[3]. On the other hand, setting it too small can lead to poor exploitation.

We present a modified version of the Bouncy Particle Sampler, where one can consider the need for refreshments are alleviated via the introduction of a randomised bounce kernel $\mathcal{Q}_B$. In Vanetti et al. 2017 a few closely related examples of randomised kernels are discussed. Specifically, at a bounce event we perform an autoregressive random direction change in the linear subspace orthogonal to the gradient of the energy at that point.

Definition 4.3.5 (Orthogonal Subspace Crank-Nicolson)**.** The Orthogonal Subspace Crank-Nicolson (OSCN) is a PDMP with linear flow, with an event rate

$$\lambda(x, v) = \max(0, \langle v, \nabla\psi(x)\rangle).$$

At a bounce event, a random speed change $v'_\perp$ is sampled in the linear subspace orthogonal to the gradient $\nabla\psi(x)$ and perform a Crank-Nicolson step with parameter $\rho \in (0, 1)$, and flip the component of v parallel to $\nabla\psi(x)$.

[3]Considering for instance a Wiener process W_t, it then holds that $W_{t+u} - W_t \sim \mathcal{N}(0, u)$, i.e., this increment has standard deviation $\sqrt{u}$.

The random reflection is done by orthogonally decomposing the current velocity $v = v_\perp + v_p$, where v_p is parallel to the gradient at the current position x and $v_\perp$ is orthogonal to v_p. One then samples a random increment u from the velocity distribution $\mathcal{N}(0, I_n)$, projects this increment to the orthogonal subspace, and updates the orthogonal component by performing a Crank-Nicolson step, i.e., by setting

$$v'_\perp = \rho \cdot v_\perp + \sqrt{1-\rho^2} \cdot u_\perp.$$

The speed after the event is then set to

$$v' = -v_p + v'_\perp.$$

Flipping the sign of the component parallel to $\nabla\psi$ ensures that the new state $(X', V') \sim \mathcal{Q}((x, v), \cdot)$ satisfies

$$\langle v', \nabla\psi(x)\rangle = -\langle V', \nabla\psi(X')\rangle\,, \tag{4.3.5}$$

almost surely (that is, with probability 1). The method is also illustrated in Algorithm 2.

Proposition 4.3.6. *The product measure $\mu(\mathrm{d}z) = \Pi(\mathrm{d}x) \otimes \mathcal{N}(\mathrm{d}v; 0, I_d)$ is an invariant measure for the Orthogonal Subspace Crank-Nicolson.*

Proof sketch. The Markov kernel $\mathcal{Q}$ leaves x unchanged and we can factorise

$$\mathcal{Q}((x, v), (\mathrm{d}x', \mathrm{d}v')) = \delta_x(\mathrm{d}x')\kappa_x(v, \mathrm{d}v').$$

From (4.3.1) we have that the generator is given by

$$\mathfrak{A}f(x, v) = \langle v, \nabla_x f(x, v)\rangle + \lambda(x, v)\int (f(x, v') - f(x, v))\,\kappa_x(v, \mathrm{d}v').$$

Note that $\mu(\mathrm{d}z) = \rho(\mathrm{d}v)\pi(x)\mathrm{d}x$, where $\rho(\mathrm{d}v) := \mathcal{N}(\mathrm{d}v; 0, I_d)$.

Now, we have that

$$\begin{aligned}\int \mathfrak{A}f(z)\mu(\mathrm{d}z) = &\int\int \langle v, \nabla_x f(x, v)\rangle \rho(\mathrm{d}v)\pi(x)\mathrm{d}x \\ &+ \int\int \lambda(x, v)\int (f(x, v') - f(x, v))\,\kappa_x(v, \mathrm{d}v')\rho(\mathrm{d}v)\pi(x)\mathrm{d}x.\end{aligned}$$

Using integration by parts and $\pi(x) = \exp(-\psi(x))$ yields

$$\begin{aligned}\int \mathfrak{A}f(z)\mu(\mathrm{d}z) = &\int\int \langle v, \nabla\psi(x)\rangle f(x, v)\rho(\mathrm{d}v)\pi(x)\mathrm{d}x \\ &+ \int\int\int \lambda(x, v)f(x', v')\mathcal{Q}((x, v), \mathrm{d}(x', v'))\rho(\mathrm{d}v)\pi(x)\mathrm{d}x \\ &- \int\int \lambda(x, v)f(x, v)\rho(\mathrm{d}v)\pi(x)\mathrm{d}x.\end{aligned}$$

By (4.3.5) for the second term, and by the definition of the rate λ we now have that

$$\begin{aligned}\int \mathfrak{A}f(z)\mu(\mathrm{d}z) &= \int\int \langle v, \nabla\psi(x)\rangle f(x,v)\rho(\mathrm{d}v)\pi(x)\mathrm{d}x \\ &\quad + \int\int \max(0, -\langle v', \nabla\psi(x)\rangle) f(x,v')\rho(\mathrm{d}v')\pi(x)\mathrm{d}x \\ &\quad - \int\int \max(0, \langle v, \nabla\psi(x)\rangle) f(x,v)\rho(\mathrm{d}v)\pi(x)\mathrm{d}x \\ &= 0,\end{aligned}$$

since $\max(0,-a) - \max(0,a) = -a$.

To complete the proof, one needs to show that the process (Z_t) is *Feller* (see Davis 1993, page 76), and that the set of functions f for which the above identity holds is large enough (Davis 1993, Proposition 34.7). This is out of the scope of this presentation. □

4.3.2 Simulating a PDMP

The deterministic evolution of the state between events allows generating a continuous trajectory from a discrete set of times, making the exact simulation of a PDMP tractable. Most of the computational complexity arises from the sampling of these event times, since they are the first event times of an inhomogeneous Poisson process. This can be done by *thinning* a homogeneous Poisson process if the rate λ is bounded, locally or globally. Loosely, one simulates events happening at faster rate, and then randomly reject these according to a rejection probability. For an in-depth description, see Bouchard-Côté, Vollmer, and Doucet 2018, Section 2.3.2.

Algorithm 2 OSCN-BPS

Require: $\nabla\psi, \rho \in [0,1], T, (x^{(0)}, v^{(0)}) \in \mathbb{R}^n \times \mathbb{R}^n$
$t \leftarrow 0$
for $i = 1, 2, ..$ **do**

1. Simulate the event time $\tau_i \in (0, \infty)$ as the first event time of an inhomogeneous Poisson process with rate
$$\lambda(t) = \left\langle \nabla\psi\left(x^{(i-1)} + tv^{(i-1)}\right), v^{(i-1)}\right\rangle.$$
2. Set $x^{(i)} = x^{(i-1)} + \tau v^{(i-1)}$.
3. Decompose $v^{(i-1)} = v_\perp + v_p$, where $\left\langle v_\perp, \nabla\psi\left(x^{(i)}\right)\right\rangle = 0$.
4. Set $v'_\perp \leftarrow \rho \cdot v_\perp + \sqrt{1-\rho^2} \cdot \left(u - \frac{\langle \nabla\psi(x^{(i)}), u\rangle}{\|\nabla\psi(x^{(i)})\|_2^2} \cdot u\right)$, where $u \sim \mathcal{N}(0, I_n)$.
5. Set $v^{(i)} \leftarrow -v_p + v'_\perp$.
6. If $\sum_{j=1}^{i} \tau_i \geq T$, break.

end for

5

Bayesian inversion with neural generative priors

With the required background in place, we now turn our attention to a linear inverse problem with additive noise

$$y = \mathscr{A}x + h,$$

where h is the outcome of the $\mathcal{Y}$-valued random variable H with probability density π_H. As we have seen in Chapter 2, π_H defines for fixed $x \in \mathcal{X}$ a likelihood $\pi(y \mid x)$ by

$$y - \mathscr{A}x = h, \quad \pi(y \mid x) = \pi_H(y - \mathscr{A}x).$$

Still, it is in many cases difficult to define a realistic prior in the sense that samples $x \sim P_X$ resemble possible solutions to the problem at hand. Moreover, it is often assumed that the true prior is only supported on a low-dimensional manifold of the reconstruction space $\mathcal{X}$ (see Section 5.11.3, Goodfellow, Bengio, and Courville 2016). This motivates the use of a generative model to learn a low-dimensional parametrisation of a prior.

5.1 Fitting a neural generative prior

Assume that we have access to a dataset $\mathcal{D} = \{x_i\}$ of samples from the true prior P_X. A generative network $g_\theta : \mathcal{Z} \to \mathcal{X}$ is trained on $\mathcal{D}$, yielding a transformation of the latent random variable $\mathscr{Z}$, taking values in $\mathcal{Z}$, such that $g_\theta(\mathscr{Z}) \sim Q_\theta \approx P_X$. For this to be tractable, $\mathcal{X}$ needs to be finite dimensional, i.e., in some cases this means that one needs to choose a discretisation of the reconstruction space.

Specifically, in this work we propose using a generative model from the GAN family, and out of these, the WGAN-GP algorithm has been empirically shown to perform well. For instance, all the prominent StyleGAN models, see e.g. StyleGAN-3 Karras et al. 2021, are based on the WGAN-GP algorithm.

The distribution and domain $\mathcal{Z}$ of the latent variable $\mathscr{Z}$ is a hyperparameter, and is typically chosen as a standard normal on $\mathbb{R}^l$, where $l \ll \dim(\mathcal{X})$. This choice yields a simple prior density on $\mathcal{Z}$ given by $\pi(z) = \exp(-\frac{1}{2}\|z\|_2^2)$.

Other choices of prior distributions have been discussed in the GAN literature. In the paper accompanying BigGAN, Brock, Donahue, and Simonyan 2019, the authors chooses to use a truncated normal instead, and lists many other possible choices.

To reiterate, the latent variable together with the trained generator g_θ serves as an approximate parametrisation of P_X. To generate a new sample x' from Q_θ, one samples a $z' \sim \mathcal{N}(0, I_l)$ and maps this to $x' = g_\theta(z')$.

5.2 The posterior on the latent space

The key step now is essentially to compose the likelihood $\pi(y \mid \cdot) : \mathcal{X} \to \mathbb{R}$, given by the forward operator $\mathscr{A}$ and a density for the noise, with the generator $g_\theta : \mathcal{Z} \to \mathcal{X}$, which yields a likelihood for the latent variable z by

$$\begin{aligned} \pi_z(y \mid \cdot) := \pi(y \mid \cdot) \circ g_\theta : \mathcal{Z} &\to \mathbb{R} \\ z &\mapsto \pi_h(y - \mathscr{A}(g_\theta(z)). \end{aligned}$$

The posterior for z is then

$$\pi(z \mid y) \propto \pi(z)\pi(y \mid g_\theta(z)) = \pi(z)\pi_h(\mathscr{A}(g_\theta(z)) - y).$$

Further, assuming that the chosen noise model $\mathscr{H}$ is a zero-mean Gaussian with a specified covariance matrix Σ, and with $\mathscr{Z}$ a standard normal, we find that

$$\pi(z \mid y) \propto \exp\left\{-\frac{1}{2}\left(\|z\|_2^2 + \|\mathscr{A}(g_\theta(z)) - y\|_{\Sigma^{-1}}^2\right)\right\}.$$

The negative logarithm of the posterior then becomes

$$\begin{aligned} \psi(z) = -\log \pi(z \mid y) &= \frac{1}{2}\left(\|z\|_2^2 + \|\mathscr{A}(g_\theta(z)) - y\|_{\Sigma^{-1}}^2\right) + C \\ &= \frac{1}{2}\|z\|_2^2 + \frac{1}{2}\left(\mathscr{A}(g_\theta(z)) - y\right)^T \Sigma^{-1}(\mathscr{A}(g_\theta(z)) - y) + C, \end{aligned}$$

where C is some additive constant, and the gradient is then[1]

$$\nabla\psi(z) = z + J_{g_\theta}^T(z)\mathscr{A}^T\Sigma^{-1}(\mathscr{A}(g_\theta(z)) - y),$$

where J_{g_θ} is the Jacobian of the generator g_θ with respect to z. Writing

$$v := \mathscr{A}^T\Sigma^{-1}(\mathscr{A}(g_\theta(z)) - y),$$

we have that

$$\nabla\psi(z) = z + J_{g_\theta}^T(z)v.$$

This is useful since vector Jacobian products $J_f^T v$ is precisely what reverse mode automatic differentiation efficiently computes. To sample this posterior using OSCN-BPS, one needs to evaluate the gradient ψ which should be tractable with the available automatic differentiation software. Finally, applying the generator to the posterior sample of the latent vector yields a posterior sample in the reconstruction space $\mathcal{X}$.

[1] found with the help of `matrixcalculus.org`, Laue, Mitterreiter, and Giesen 2020, and the chain rule.

6

Experimental setup and results

To test the method we define a synthetic problem. This consists of two parts, an imagined prior as well as a difficult inverse problem. On this problem, we compare both the quality of reconstructions, as well as the informativeness of the posterior resulting from our learned prior to carefully tuned classical reconstruction methods.

In this problem, we want to reconstruct an image from a noised subsample of its Fourier coefficients. This can be seen as a simplified version of magnetic resonance imaging. Specifically we will consider the case where only low frequency coefficients are measured. This naturally leads to a loss of information about finer detail in the image. The level of ill-posedness is proportional to the degree of the subsampling.

6.1 Defining the inverse problem

For simplicity, let $\mathcal{X} = \mathbb{R}^{64\times 64}$ be the reconstruction space and let $\mathcal{Y} = \mathbb{C}^{64\times 64}$ be the data space. We define the forward operator $\mathscr{A} : \mathcal{X} \to \mathcal{Y}$ to be given by

$$\mathscr{A}(x) = \mathcal{M} \circ \mathcal{F}(x),$$

where $\mathcal{M}$ is the masking operator, and $\mathcal{F}$ is the two-dimensional discrete Fourier transform (DFT). Since it affects the scale of the observational noise, note that the unnormalised DFT is used. The masking operator is given by the element-wise product of a Boolean matrix $M \in \{0,1\}^{64\times 64}$, and is therefore self-adjoint.

We model the noise as independent and identically distributed complex Normal noise on the observed coefficients, that is

$$y_{i,j} = \begin{cases} \mathscr{A}(x)_{i,j} + h_{i,j}, & \text{if } M_{i,j} = 1, \text{where } h_{i,j} \sim \mathcal{CN}(0, \frac{\sigma^2}{2} I_2) \\ 0, & \text{otherwise,} \end{cases}$$

which leads to the conditional negative log density for the observation y given the object of interest $x \in \mathcal{X}$

$$-\log \pi(y \mid x) = \frac{1}{\sigma^2} \|\mathscr{A}(x) - y\|_2^2.$$

For the problem, we also implicitly define a true underlying prior P_X by constructing a random image generator, creating variations of greyscale images of ellipses of

varying shades in a random but fairly predictable pattern. Figure 6.1 shows 25 samples from this prior. The images are 64-by-64 pixels large, and are as such elements of $[0,1]^{64\times 64}$. A detailed description of the procedure is presented in Appendix A.

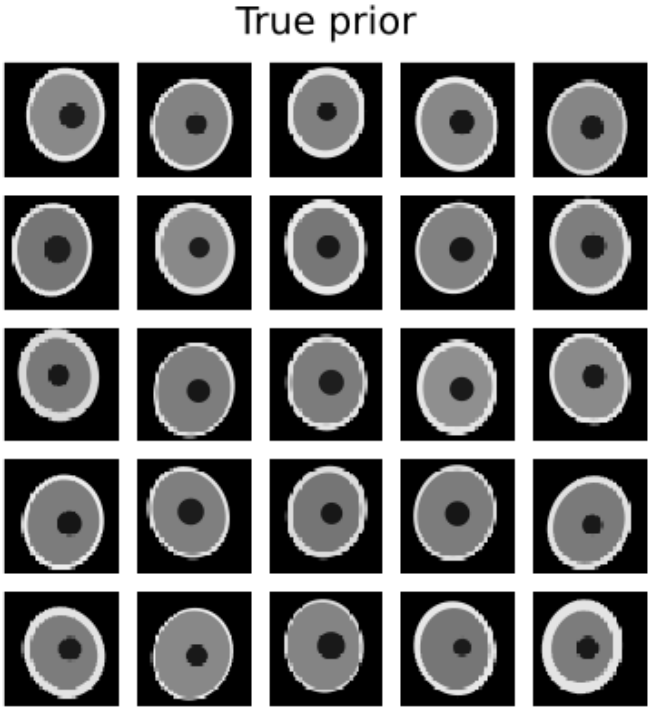


Figure 6.1: 25 images sampled from the true prior, implicitly defined by the random image generator.

6.2 Solving the inverse problem using classical priors

To test the method, we sample an image x_{true} from the true prior P_X. We choose the mask M to select the 10×10 lowest frequency components in both directions and discard the rest. To form the observation, we also corrupt the observed Fourier coefficients with complex Normal noise with $\sigma = 10$, leading to a slight corruption of the image in pixel space. As shown in Figure 6.2, this is a low noise example and the difficulty mostly arises from the low rank of $\mathscr{A}$.

We first consider the performance of two classical imaging priors on this problem. Since we have access to the ground truth, we can tune these methods by considering different metrics. We have chosen to use the mean squared error (MSE) and the *structural similarity index* (SSIM), Wang et al. 2004. The question is two fold: does the regularisation method produce a good point estimate? And is the posterior informative?

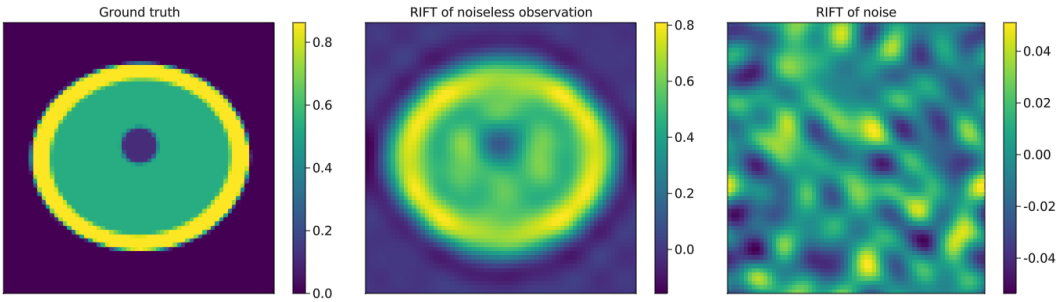


Figure 6.2: True image we want to reconstruct (left), real part of the inverse Fourier transform, abbreviated RIFT, of the masked coefficients (middle), and real part of the additive observational noise in image space (right).

6.2.1 Tikhonov regularisation

A typical choice to regularise the inverse problem is to penalise the L_2-norm of the solution. This is the simplest case of Tikhonov regularisation, and it is most commonly used to find a point estimate of the solution. The optimisation problem is then

$$\hat{x} = \underset{x}{\operatorname{argmin}} \|\mathscr{A}(x) - y\|_2^2 + \lambda \|x\|_2^2, \tag{6.2.1}$$

where $\lambda \in \mathbb{R}_+$ is a regularisation that can be tuned. This corresponds to selecting a zero mean independent Gaussian prior on the pixels, where the prior scale σ_p is inversely related to the regularisation parameter λ, and then computing the resulting *maximum a posteriori* estimate (MAP).

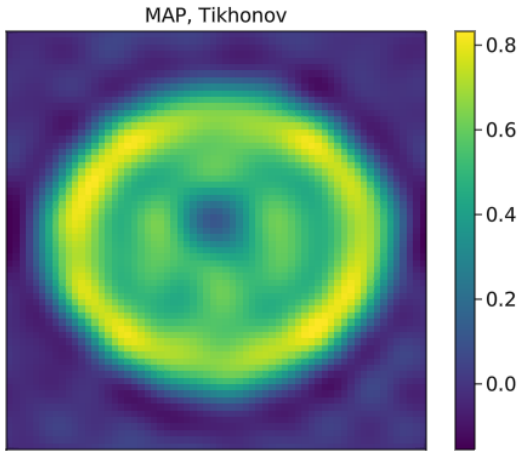
We found that the choice of $\sigma_p = 1/2$ achieved the best point estimates using this method, scoring a SSIM of 0.46 and an MSE of 0.012.

The posterior corresponding to this choice of prior scale is then sampled using the preconditioned Crank-Nicolson (pCN) MCMC method. The parameter $\rho = 0.045$ was tuned to achieve an acceptance rate of approximately 25%. Figure 6.3 shows the posterior mean and standard deviations after sampling the posterior 1,000,000 times starting from the posterior mode, keeping only every thousandth step. Figure 6.4 shows 20 of these samples, as well as the result of applying $\mathscr{A}^T\mathscr{A}$ to these samples. The latter visualises what components of the image affect the likelihood term in the posterior.

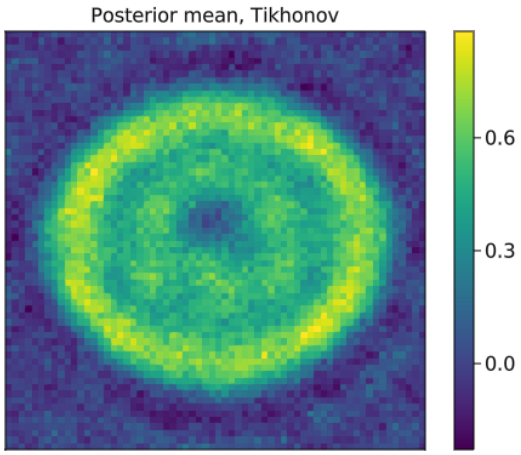
6.2.2 Structural prior

A more informed prior than the previous independent zero mean Gaussian one would be to use a Gaussian Markov random field (GMRF) prior. This captures the notion that pixel values should be positively correlated with the neighbouring values. For instance, such priors can be used for image in-painting and de-noising.

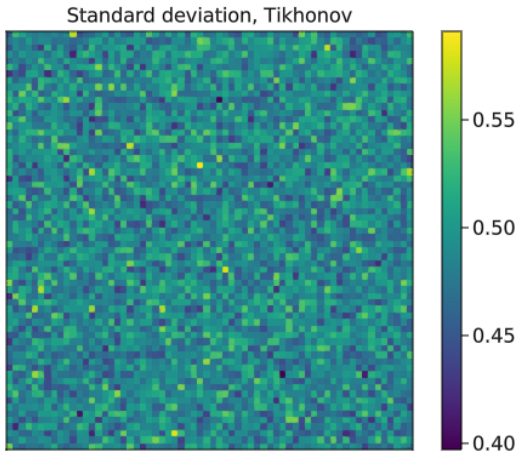
For the GMRF we define a sparse precision matrix Λ by computing the 2 dimensional



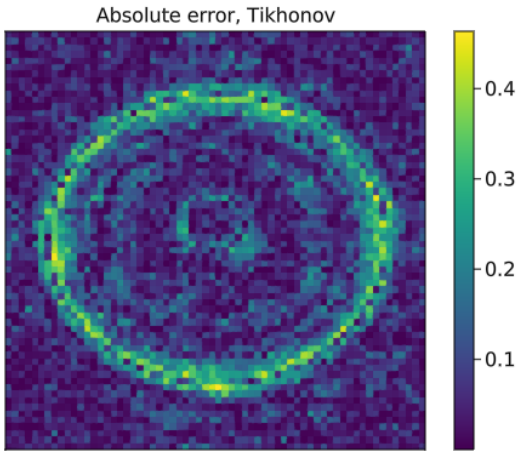
(a) MAP



(b) Posterior mean



(c) Standard deviation



(d) Absolute error of the posterior mean

Figure 6.3: Summary statistics from inversion using the Tikhonov prior computed from a pCN chain of 1,000,000 samples starting at the MAP estimate shown in (a). Subfigure (b) shows the posterior mean, (c) shows the pixelwise standard deviations, and (d) shows the absolute error of the posterior mean compared to the ground truth.

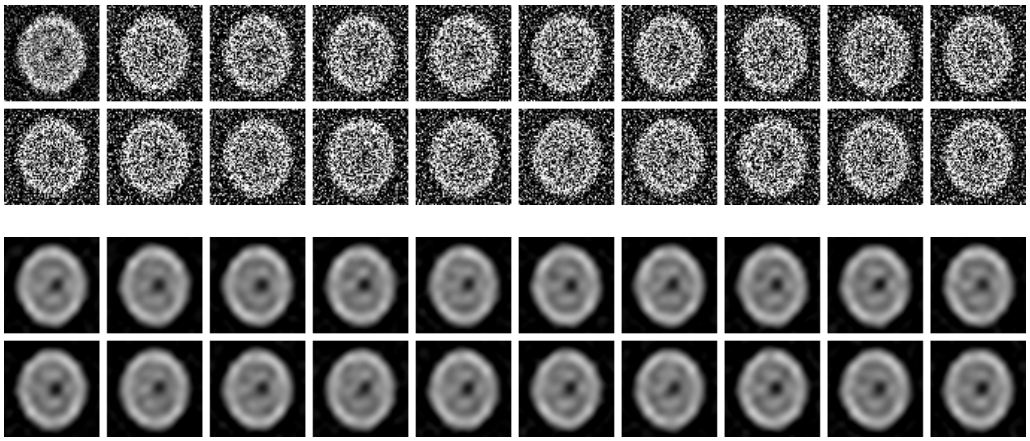


Figure 6.4: Twenty posterior samples corresponding to the Tikhonov prior in the top two rows. The bottom two rows shows a visualisation of the observed low frequency components in image space of the same samples affecting the likelihood term in the posterior.

grid Laplacian Λ_0 of the 64×64 pixel grid, and perturb this slightly with a scaled identity matrix to ensure positive definiteness, setting

$$\Lambda = \Lambda_0 + cI,$$

with $c = 10^{-6}$. First Cholesky factorising $LL^T = \Lambda$, the negative log posterior then can be written as

$$\psi(x) = \frac{1}{2\sigma_e^2}\|\mathcal{A}x - y\|_2^2 + \lambda\|L^T x\|_2^2,$$

up to some additive constant, where the parameter λ determines the strength of the prior. As before, we tune λ by assessing the MSE and SSIM between the MAP and the ground truth.

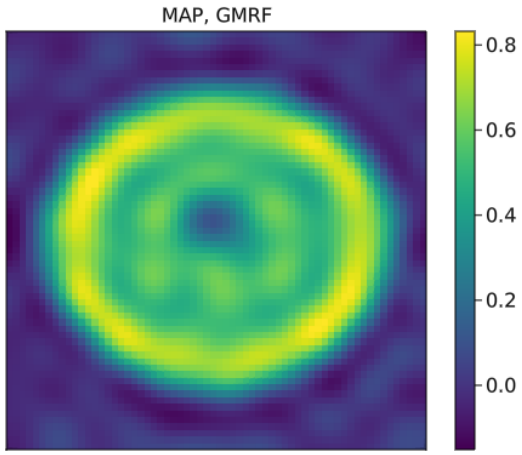
To sample from the posterior, we use the pCN algorithm with $\rho = 0.005$, tuned to achieve an approximate acceptance rate of 20-25%, initialising the chain at the MAP estimate. Figure 6.5 shows the resulting MAP, conditional mean and standard deviations after 10,000 steps.

6.3 Solving the inverse problem using a neural generative prior

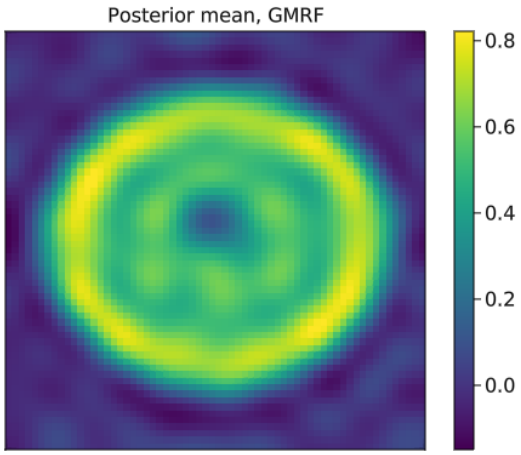
Having tried two typical imaging priors, we now use our proposed method. For brevity, we call this the neural generative prior method (NGP). First, we approximate an informed prior by fitting a generative model and then we produce samples from the resulting posterior.

6.3.1 Learning the prior

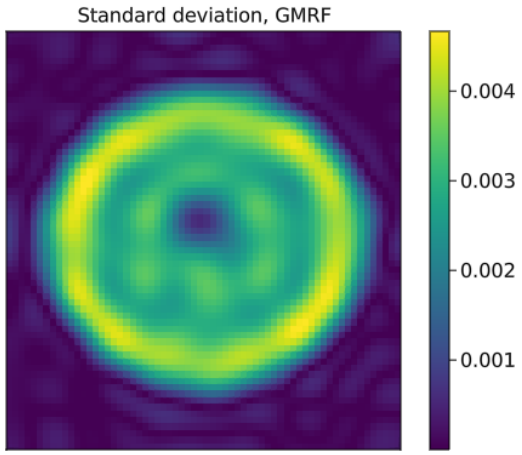
A generative network g_θ is trained to approximate the implicitly defined true prior using the WGAN-GP algorithm. We use a DCGAN architecture for both the critic



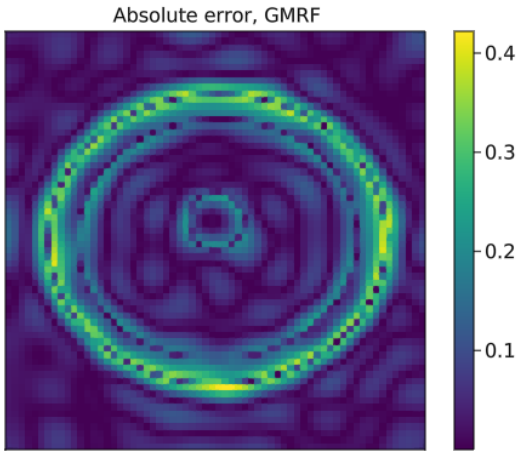
(a) MAP



(b) Posterior mean



(c) Standard deviation



(d) Absolute error of the posterior mean

Figure 6.5: Summary statistics from inversion using the GMRF prior computed from a pCN chain of 10,000 samples starting at the MAP estimate shown in (a). Subfigure (b) shows the posterior mean, (c) shows the pixelwise standard deviations, and (d) shows the absolute error of the posterior mean compared to the ground truth.

and the generator. More importantly, to guarantee that the gradient ∇g_θ of the generator is locally Lipschitz continuous, we use Swish activations for each intermediate layer. Additionally, we also add a small weight penalty to the loss of the generator since the weights imposes an upper bound on the Lipschitz constant of the gradient.

The WGAN is trained with the standard choice of gradient penalty $\lambda = 10$, using minibatches of 16 real and 16 generated images. We use ADAM for both the generator and the critic, with parameters $\eta = 0.05$ and $(\beta_0, \beta_1) = (0.9, 0.99)$. We use a latent dimension of 64 with a multivariate standard normal prior.

After training for 10,000 minibatches, the generated image quality has reached a satisfactory level and training is stopped. Figure 6.6 shows a sample of 25 generated images from the trained WGAN.

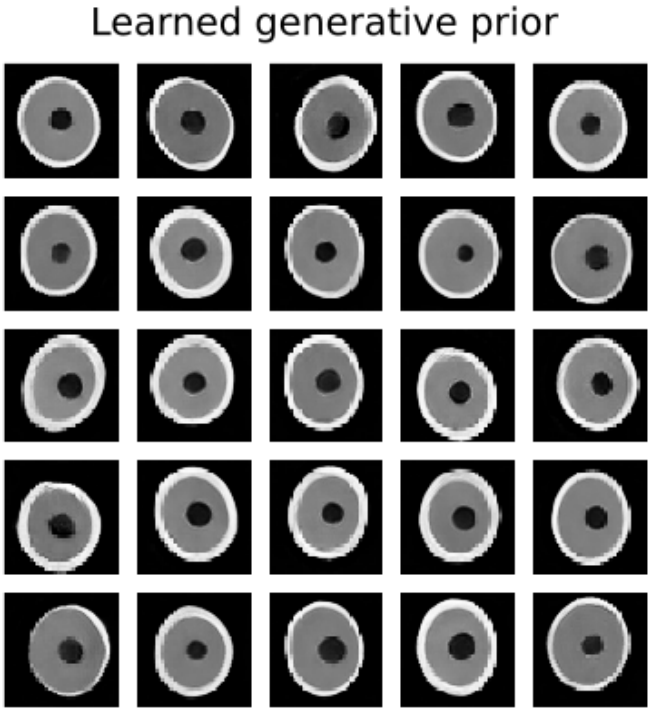


Figure 6.6: Images sampled from the learned neural generative prior.

6.3.2 Solving the inverse problem

As we saw in the previous chapter, with the neural prior Q_θ, the negative log posterior ψ of the latent vector z given y is

$$\frac{1}{2\sigma^2} \|\mathscr{A}(g_\theta(z)) - y\|_2^2 + \|z\|_2^2,$$

up to some additive constant. The gradient is then

$$\nabla\psi(z) = z + J_{g_\theta}^T(z)v,$$

where

$$\begin{aligned} v &= \frac{1}{\sigma^2}\mathscr{A}^T(\mathscr{A}(g_\theta(z)) - y) \\ &= \frac{1}{\sigma^2}\Re\left(\mathcal{F}^T\mathcal{M}^T(\mathcal{M}\mathcal{F}g_\theta(z) - y)\right) \\ &= \frac{1}{\sigma^2}\Re\left(\mathcal{F}^T(\mathcal{M}\mathcal{F}g_\theta(z) - y)\right), \end{aligned}$$

with $\Re$ being the real part of each coordinate.

To sample from this posterior, we first find a mode of the posterior using gradient descent with a fixed small step size, characterising it as a mode by approximating the Hessian using finite differencing and checking that it is positive definite. We then sample the posterior starting from this mode using OSCN-BPS for $T = 100$ time units, with $\rho = 0.98$ and a local rate bound $\bar{\lambda}_t = \lambda_t + c\|v\|_2^2$ valid for one time unit forward. For the upper Lipschitz bound we found that $c = 20,000$ worked. Sampling takes an hour of wall clock time.

Figure 6.7 shows summary statistics after discretising the trajectory to a sample of size 100. Figure 6.8 shows 20 samples from the resulting trajectory and. Illustrating the exploration of the posterior by OSCN-BPS, Figure 6.9 shows the trace of the first 8 latent variables, including the optimisation steps. We also try sampling the NGP posterior using 1,000,000 pCN steps. The trace of the first 8 latent variables is shown in 6.10.

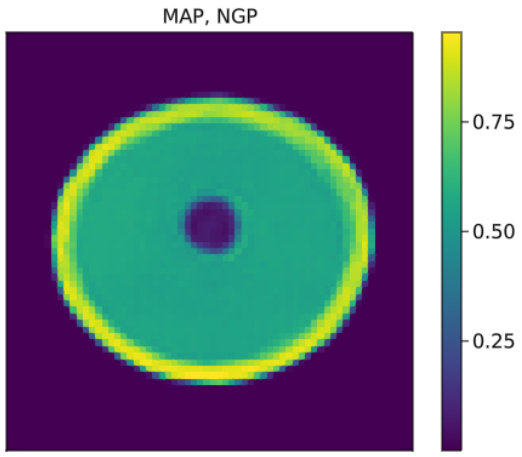
6.4 Evaluation and discussion

To wrap up this chapter, we discuss the findings and also consider some additional examples and metrics.

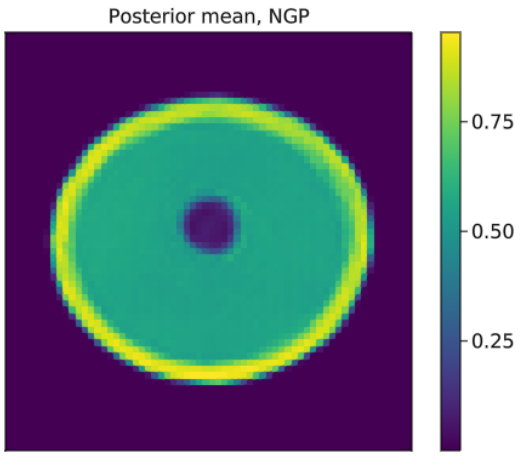
6.4.1 Performance of point estimators

For both the Tikhonov prior and the GMRF prior, the best point estimates do not differ much from just applying the adjoint of the forward operator, $\mathscr{A}^T$, to the observation. Clearly, in this case, these priors do not promote a blocky reconstruction. This is in stark contrast to the posterior mode found by using the learned generative prior, which visually contains more features such as near constant regions and sharp edges.

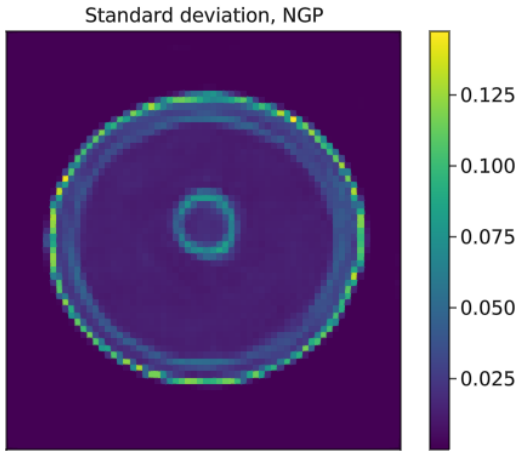
We can quantify this by computing the SSIM, where the NGP estimate scores 0.92 out of 1, and the GMRF estimate scores 0.38. The mean squared pixel error for the NGP reconstruction is 0.005 whereas the GMRF estimate has an MSE of 0.013. Notably, the errors of the NGP lie on the edges of the object. The forward operator effectively filters out sharpness in the image so this is not unexpected. The other methods also perform the worst on the edges, but they also have errors in other regions of the image.



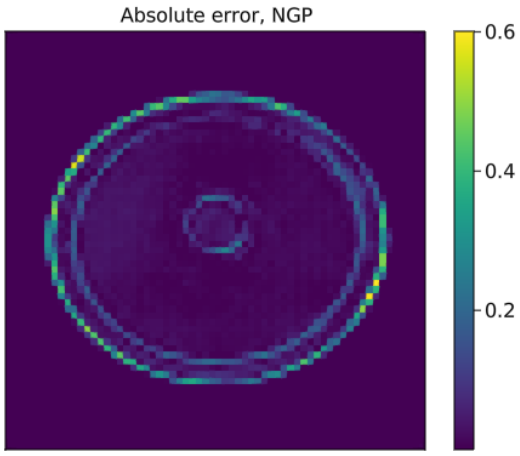
(a) MAP



(b) Posterior mean



(c) Standard deviation



(d) Absolute error of the posterior mean

Figure 6.7: Summary statistics from inversion using the NGP computed from a discretised OSCN-BPS trajectory starting at the MAP estimate (a), showing the posterior mean (b), pixelwise standard deviations (c) and the absolute error of the posterior mean compared to the ground truth (d).

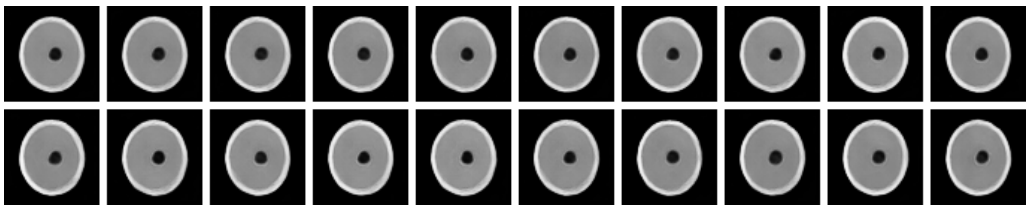


Figure 6.8: Twenty posterior OSCN-BPS generated samples from the NGP posterior, discretised equally spaced over the entire trajectory.

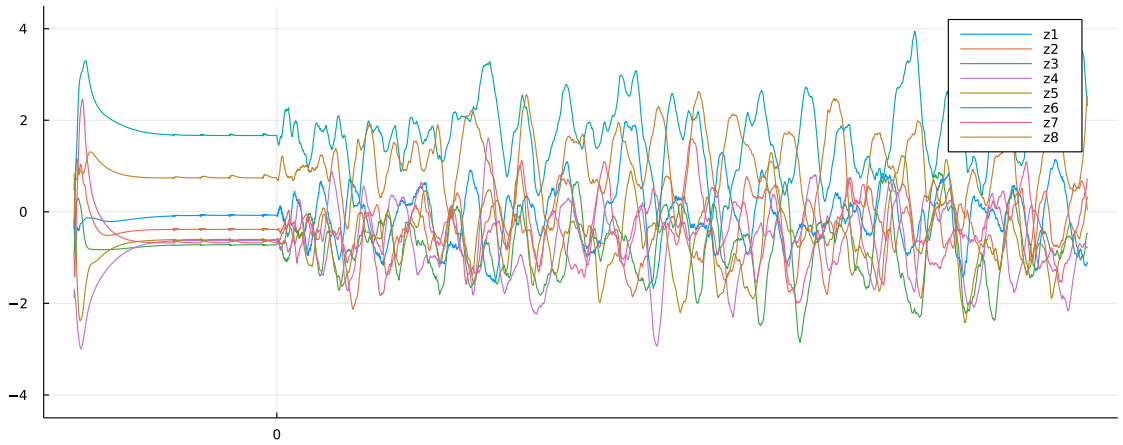


Figure 6.9: Trace of the first 8 latent variables from a OSCN-BPS trajectory running for $T = 100$ time units on the NGP posterior. The computation runs for a wall clock time of 62 minutes. The negative $x-$axis shows the gradient descent steps approaching the mode of the posterior.

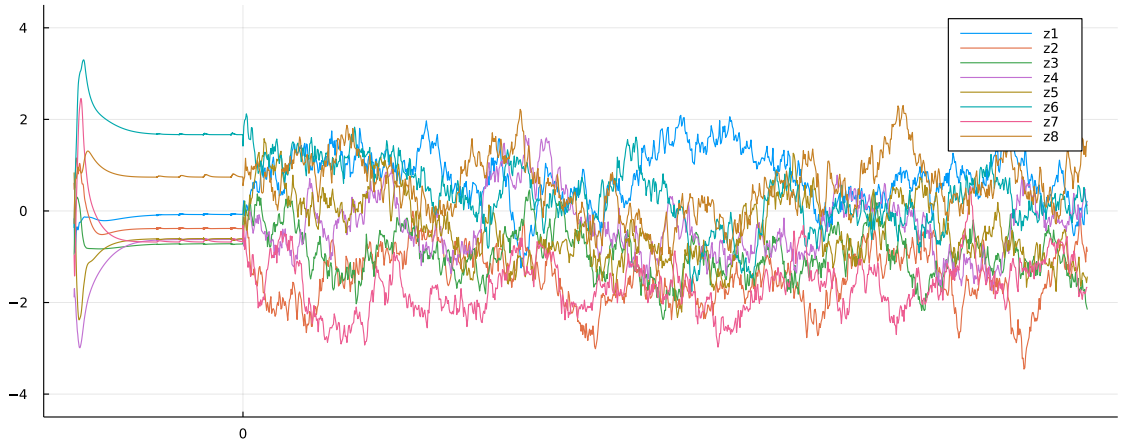


Figure 6.10: Trace of the first 8 latent variables from a pCN chain on the NGP posterior, with $\rho = 0.012$ to achieve an acceptance rate of 25% starting at the mode of the posterior. The chain is run for 1,000,000 steps, running for a wall clock time of 15 minutes. The negative $x-$axis shows the gradient descent steps approaching the mode of the posterior.

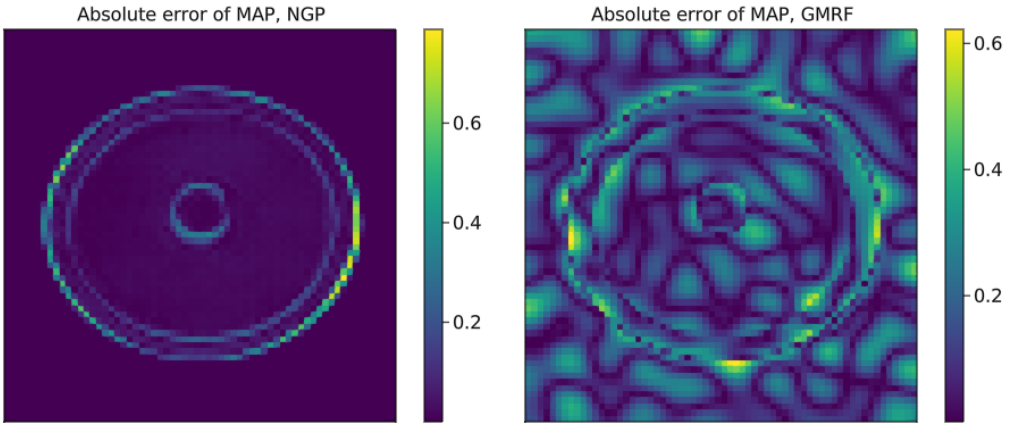


Figure 6.11: Absolute error of point estimates of NGP (left) and GMRF (right) after increasing the scale of the noise by a factor of 5.

The example we tried was fairly noiseless. Figure 6.11 shows the effect of increasing the scale of the noise by a factor of 5, where the mean squared error of the GMRF estimate has increased to 0.032 whereas the estimate produced by the NGP increased to 0.007, showing a high stability to noise.

6.4.2 Informativeness of the posteriors

So far we have only discussed the point estimates, and not the information gained from exploring the posterior. From the experiments it is fairly clear that it is not interesting to sample the Tikhonov posterior. To a human eye, the samples look corrupted by high frequency noise. This noise pattern is typically referred to as *salt-and-pepper* noise. We note that this is not surprising in this case. This is due to the fact that the likelihood term only is affected by low frequencies in the image, so states with high frequency noise are, as long as the scale is not too large, likely. This noise drowns out any information one could have gained by for instance considering the posterior standard deviation, which is roughly equal to 0.5 over the entire image.

The local dependencies of the Gaussian Markov random field prior does hinder this effect. From the posterior samples of this method, one can make out a more interpretable pattern of the pixel-wise standard deviation. While one first might be optimistic about the uncertainty around the edges of the shape, we note that the standard deviation seems very highly correlated with the posterior mean. That is, there is more variation in points where more mass is predicted. Still, this is a nice result, this seems to have quantified uncertainty in the pixel values concentrated at the shape.

The NGP posterior does however tell us more. For instance, just considering the black regions in the original image, the NGP posterior shows both confidence and accuracy in the prediction of these values. Instead, the uncertainty is focused at the edges in the image, which as we noted previously is precisely the region that is most obscured by the forward operator of the problem. This tells us that, for this

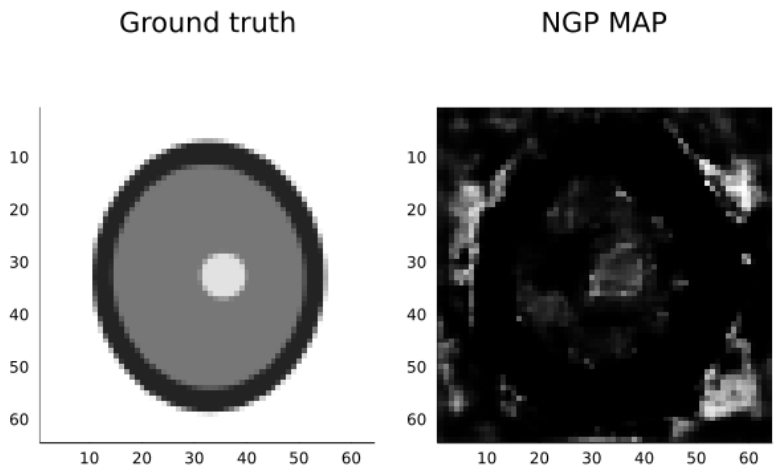


Figure 6.12: Failed NGP reconstruction of an out of distribution example.

example at least, the prior is rich enough to contain both a close estimate of the ground truth, as well as a neighbourhood of similar shapes that could also match the observation well.

Finally, we stress that while this learned prior is in some sense very specialised, it is not in any way coupled with the actual inverse problem. The forward operator and noise model are completely replaceable with other implementable methods.

6.4.3 Performance of the sampling algorithms

We have tried two different methods for sampling the posterior, but trace plots and corresponding measures of wall clock time does not really permit confidence in any conclusions. For instance, there is no guarantee that we explored the posterior well. During experiments, the OSCN-BPS required increasing the tuning parameter a lot related to the local rate bound. Since we are working with neural networks, one is essentially more or less constrained to single precision floating point numbers and round off can become a pronounced problem in this setting. The need for a high bound parameter can be due to using a way too long validity interval for the bound.

With the caveats in place, it certainly looks like the BPS travels much faster and evolves less like a random walk compared to pCN. This is one promising aspect of these methods.

6.4.4 Failed reconstruction for out-of-distribution sample

It is easy to break the NGP reconstruction by feeding it an out of distribution example. Consider for instance exactly the same setup as before, but inverting the pixel intensities of the ground truth. This image is not in the range of the generator, and the reconstruction is not satisfactory as seen in Figure 6.12. This example highlights the need for relevant training examples.

7

Conclusion and outlook

Overall, we have demonstrated that the method works on a simple but not trivial example. Before touching on the the technical challenges ahead, it is a good idea to remind ourselves of the potential of this approach. An illuminating example of the potential benefits of developing these methods further comes from medicine.

For many medical predicaments, MRI scans are the gold standard test. However, the long acquisition times puts heavy constraints on its widespread usage in everyday medical practice. This potentially leads to a misdiagnosed conditions, which carry risks ranging from lower quality of life and unnecessary stress for the patient up to untimely death.

If, and it is a big if, one could effectively learn a highly realistic prior for, for instance, brain scans, we have shown that this allows us to drastically under sample the data, which could reduce acquisition times drastically. In addition, it could also allow for quantifying uncertainty in the reconstructions which for instance allows one to measure the confidence in predictions. Scaling the method up to real life inverse problems is not an easy challenge. We have considered relatively small 2D images in our experiments, but MRI scans are most often large three dimensional volumes which increases the complexity by a staggering amount. Machine learning research can at times be resource intensive, and while this needs to be taken in consideration when attempting larger problems, there are clear potential gains for the society at whole. In closing, we still think that this could be a fruitful research direction.

Bibliography

[1] J. Adler and O. Öktem. *Deep Bayesian Inversion*. 2018. arXiv: 1811.05910 [stat.ML].

[2] M. Arjovsky and L. Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *ArXiv* abs/1701.04862 (2017).

[3] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[4] P. Billingsley. *Probability and Measure*. Third. John Wiley and Sons, 1995.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] A. Bouchard-Côté, S. J. Vollmer, and A. Doucet. "The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method". In: *Journal of the American Statistical Association* 113.522 (2018), pp. 855–867.

[7] A. Brock, J. Donahue, and K. Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *ArXiv* abs/1809.11096 (2019).

[8] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. "MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster". In: *Statistical Science* 28.3 (2013), pp. 424 –446. DOI: 10.1214/13-STS421.

[9] M. H. A. Davis. "Piecewise-Deterministic Markov Processes: A General Class of Non-Diffusion Stochastic Models". In: *Journal of the royal statistical society series b-methodological* 46 (1984), pp. 353–376.

[10] M. Davis. *Markov Models & Optimization*. CRC Press, 1993.

[11] S. Elfwing, E. Uchibe, and K. Doya. "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning". In: *Neural Networks* 107 (2018), pp. 3–11.

[12] D. Endres and J. Schindelin. "A new metric for probability distributions". In: *IEEE Transactions on Information Theory* 49.7 (2003), pp. 1858–1860. DOI: 10.1109/TIT.2003.813506.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[14] I. Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[15] I. Gulrajani et al. “Improved training of Wasserstein GANs”. In: *Advances in neural information processing systems* 30 (2017).

[16] M. Hairer, A. M. Stuart, and S. J. Vollmer. “Spectral gaps for a Metropolis–Hastings algorithm in infinite dimensions”. In: *Annals of Applied Probability* 24 (2014).

[17] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Applied Mathematical Sciences. Springer New York, 2006. ISBN: 9780387271323.

[18] T. Karras et al. “Alias-Free Generative Adversarial Networks”. In: *NeurIPS*. 2021.

[19] S. Laue, M. Mitterreiter, and J. Giesen. “A Simple and Efficient Tensor Calculus”. In: *AAAI Conference on Artificial Intelligence, (AAAI)*. 2020.

[20] W. Mcculloch and W. Pitts. “A Logical Calculus of Ideas Immanent in Nervous Activity”. In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 127–147.

[21] J. L. Mueller and S. Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2012. DOI: 10.1137/1.9781611972344.

[22] A. O’Hagan. *Kendall’s Advanced Theory of Statistics: Volume 2B: Bayesian inference*. Kendall’s Advanced Theory of Statistics. Arnold, a member of the Hodder Headline Group, 1994. ISBN: 9780340529225.

[23] G. Peyré, M. Cuturi, et al. “Computational optimal transport: With applications to data science”. In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607.

[24] P. Ramachandran, B. Zoph, and Q. V. Le. “Swish: a self-gated activation function”. In: *arXiv preprint arXiv:1710.05941* 7.1 (2017), p. 5.

[25] F. Rosenblatt. *The perceptron - A perceiving and recognizing automaton*. Tech. rep. 85-460-1. Ithaca, New York: Cornell Aeronautical Laboratory, 1957.

[26] P. Vanetti, A. Bouchard-Côté, G. Deligiannidis, and A. Doucet. “Piecewise-deterministic Markov chain Monte Carlo”. In: *arXiv preprint arXiv:1707.05296* (2017).

[27] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

A

Appendix: Image generator

To generate a random image from the implicit true prior in Chapter 6, we start with a black image matrix $M = 0^{d\times d} \in [0,1]^{d\times d}$ and then draw three filled ellipses on M. Let $h = d/2$. If not mentioned explicitly, any (pseudo)-random draw is (pseudo)-independent. The shapes are added to the matrix with anti-aliasing.

We first draw the *skull* ellipse, which is with very high probability the largest of the three. First, we sample an intensity c_0 by

$$c_0 = \max\left(0, \min\left(1, \widetilde{c}_0\right)\right), \quad \widetilde{c}_0 \sim \mathcal{N}\left(0.9, 0.025^2\right).$$

The skull is drawn inside a random bounding rectangle, centered at the point

$$p_0 \sim \mathcal{N}\left((h,h), \left(\frac{h}{15}\right)^2 I_2\right),$$

rotated $r_0 \sim \mathcal{N}\left(0, 0.3^2\right)$ radians with width w_0 and height h_0 sampled by

$$w_0 \sim \mathcal{N}\left(1.4h, \left(\frac{h}{50}\right)^2\right), \qquad h_0 \mid w_0 \sim \mathcal{N}\left(w_0 + 0.25h, \left(\frac{h}{50}\right)^2\right).$$

Pixels inside the ellipse that best fills this rectangle is then set to c_0.

For the *brain* ellipse, we first sample an intensity

$$c_1 = \max\left(0, \min\left(1, \widetilde{c}_1\right)\right), \quad \widetilde{c}_1 \sim \mathcal{N}\left(0.5, 0.025^2\right).$$

The previous rectangle is then shrunk by a factor $s \sim \mathcal{N}(0.85, 0.05^2)$, rotated around its center p_0 by $r_1 \sim \mathcal{N}(0, 0.05^2)$ radians. Pixels inside the ellipse that best fills this perturbed rectangle is then set to c_1. That is, the brain ellipse is concentric with the skull, it is with very high probability slightly smaller and it is randomly rotated a little bit.

Finally, one small filled circle with intensity $c_2 = \max\left(0, \min\left(1, \widetilde{c}_2\right)\right), \quad \widetilde{c}_2 \sim \mathcal{N}\left(0.1, 0.01^2\right)$ and radius $R \sim \mathcal{N}(0.2h, h^2 0.025^2)$ is drawn at a distance $d \sim \mathcal{N}(0.1, 0.05^2)$ in a direction $r_2 \sim \mathcal{N}(0, 0.25^2)$ (radians) from p_0.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY